IntroCS POGIL

Activities for CS1 in Python

Student Workbook

Tricia Shepherd, Chris Mayfield, Helen Hu

With contributions from Michael Stewart, Theresa Wilson, and Barbara Wahl

Fall 2019

Contents

Panic Attack, A Friend Indeed, Oops!, Too Close for Comfort,
Let’s Make a Deal, A Friendly Assist, A Team Effort

Introduction to Python

In this course, you will work in teams of 3–4 students to learn new concepts. This activity will introduce you to the process. We'll take a first look at variables, assignment, and input/output.

Content Learning Objectives

After completing this activity, students should be able to:

- Describe differences between program and output text.
- Identify and execute Python functions for input/output.
- Write assignment statements and use assigned variables.

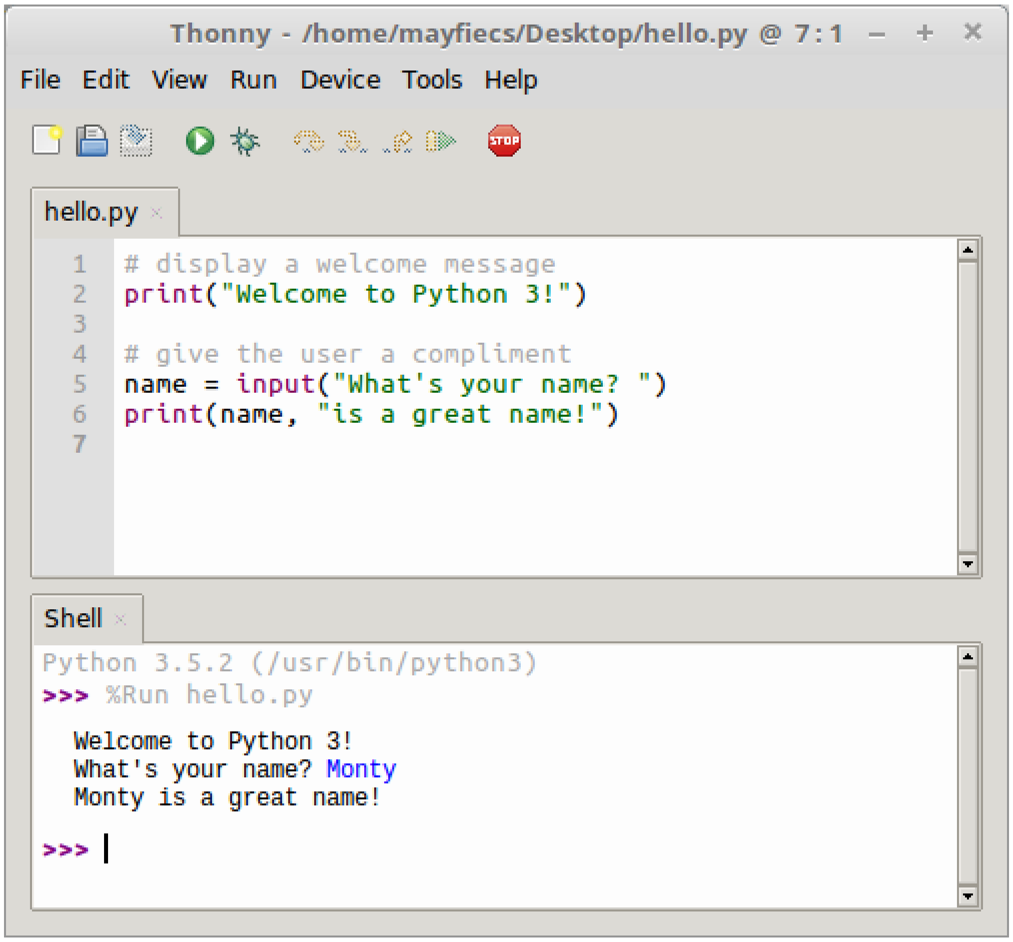
Process Skill Goals

During the activity, students should make progress toward:

- Leveraging prior knowledge and experience of other students. (Teamwork)

Model 1 Getting Started with Thonny

Thonny is an integrated development environment (IDE) for Python designed for learning and teaching programming. It is freely available at https://thonny.org/.


Thonny - /home/mayfiecs/Desktop/hello.py @ 7 : 1
File Edit View Run Device Tools Help
STOP
hello.py
Shell

```
# display a welcome message
print("Welcome to Python 3!")

# give the user a compliment
name = input("What's your name? ")
print(name, "is a great name!")
```

```
Python 3.5.2 (/usr/bin/python3)
>>> %Run hello.py

  Welcome to Python 3!
  What's your name? Monty
  Monty is a great name!

>>> 
```

Do not run Thonny yet! Answer the questions first!

Questions (15 min) **Start time: __________**

1. Based on the screenshot in Model 1:

   a) where is the Shell window?

   b) where is the Editor window?

   c) what is the name of the file in the Editor?

   d) what is the directory where this file is located?

2. Identify the number of lines corresponding to:

a) the program (in the Editor)

b) the output of the program

3. What is the symbol at the start of a line of program text not displayed as output?

4. Consider the three program lines (in the Editor) that are not displayed as output. Describe what might be the purpose of:

a) a ***comment*** line (starts with a pound sign: #)

b) a blank line

Now open Thonny on your computer, type the code shown in Model 1, save the file as hello.py, and run the program. Ask for help if you get stuck!

5. What was required before the third line of the program output was displayed?

6. In the Shell window, what is the color of:

a) the program's output?

b) the user's input?

7. Based on your experience so far, what is the difference between the text in the Editor window and the text in the Shell window?

8. Describe what appears to be the purpose of each line of Python code in the Editor window.

a) line 1:

b) line 2:

c) line 3:

d) line 4:

e) line 5:

f) line 6:

Model 2 Python Built-In Functions

You can use built-in Python ***functions*** to perform specific operations. Sometimes a function will require information (referred to as ***arguments***) to perform its operation. A function will also ***return*** a result after the operation.

To ***call*** (or use) a Python function:

- You must include parentheses after the function's name (e.g., `print()`prints a blank line).
- If the function takes one or more arguments to perform its operation, you must put that information in the parentheses (e.g., `print("Hello, world!")` prints a message).

Do not type anything yet! Read the questions first!

Python code	Shell output
input("enter the mass in grams: ")	
mass = input("enter another mass in grams: ")	
Mass	
unit = input("enter the units for mass: ")	
print(mass, unit)	
print(mass / 2)	
ten = 10	
print(ten / 2)	
abs(-1)	
abs(-1 * ten)	

Questions (15 min) **Start time: _________**

9. List the names of the three functions used in Model 2.

10. What are the arguments of the first use of the printfunction? _______

11. Type each line of code in a Python Shell, one line at a time, and write the corresponding output (if observed) in the right column of the table. If an error occurs, write what type of error it was (i.e., the first word of the last line of the error message).

Place an asterisk (*) next to any output for which you were surprised, and note what was unexpected about the output. Don't worry yet about *understanding* any strange output you may see; we will discuss what it all means by the end of class.

12. Which function delayed execution until additional input was entered?

13. Which term, ***user*** or ***programmer***, best defines the role of the person who entered the additional input? Explain.

14. Based on the Shell output, what does the word massrepresent, and how did it get its value?

15. What does the word tenrepresent, and how did it get its value?

16. Do the values of massand tenboth represent a number? Explain why or why not.

Model 3 Variables and Assignment

In programming, an ***assignment statement*** saves a value to a ***variable***. The variable "is set to" the value after the = ***operator***. Selecting concise yet descriptive variable names is considered good programming style and will make your programs easier to read.

Do not type anything yet! Read the questions first!

Python code	Shell output
`data = 12`	
`data`	
`Data`	
`Data = 34`	
`Data`	
`my data = 56`	
`my_data = 78`	
`3data = "hello"`	
`data3 = "world"`	
`data3 = hello`	
`hot = 273 + 100`	
`273 + 100 = hot`	

`import turtle`
`aTurtle = turtle()`	

`import turtle`
`aTurtle = turtle.Turtle()`
`aturtle.forward(100)`	

`import turtle`
`aTurtle = turtle.Turtle()`
`turtle.forward(100)`	

Questions (15 min) **Start time: __________**

17. Based on the information and Python code in Model 3, give an example representing each of the following:

a) an assignment statement

b) the variable being assigned

c) the assignment operator

d) the value of the variable immediately after the assignment

18. Similar to Model 2, type each line of code in a Python Shell and write the corresponding output in the space above. If an error occurs, write what type of error. Place an asterisk (*) next to any output for which you were surprised.

19. What is the observed output of a successful assignment statement?

20. After the successful execution of an assignment statement, how can you confirm the value of this variable?

21. Based on the Model 3 output, indicate whether each statement below is true or false.

a) Variable names in Python can start with a number. ______

b) Variable names in Python must start with a lower-case letter. ______

c) Variable names in Python may not include spaces. ______

d) Variable names in Python are case-sensitive. ______

22. What are the values on the right hand sides of the assignment operator in the last two programs?

23. What kind of errors occurred to the first two turtle programs? How do you correct the errors?

24. Why does the last program run without errors?

Arithmetic Expressions

Now that you've written a few programs, let's take a step back and discuss how to do arithmetic. The behavior of Python operators (+, -, *, /) depends on what type of data you have.

Content Learning Objectives

After completing this activity, students should be able to:

- Execute mathematical expressions similar to a calculator.
- Describe the function of the three Python division operators.
- Explain differences between integer and floating-point data.

Process Skill Goals

During the activity, students should make progress toward:

- Recognizing mathematical operations based on tables. (Information Processing)

Model 1 Python Calculator

In a Python Shell window, “>>>” is a ***prompt*** indicating that the interpreter is waiting for input. All text entered after the prompt will be executed immediately as Python code.

If you type a Python ***expression*** (code that results in a value) after the prompt, Python will show the value of that expression, similar to a calculator. You can use Python’s `math` module to perform more complex mathematical operations like logarithms and trigonometric operations.

Do not type anything yet! Read the questions first!

Python code	Predicted output	Actual output
2 + 3		
3 * 4 + 2		
3 * 4 + 2.0		
3(4 + 2)		
3 * (4 + 2)		
5 / 10		
5 / 10.0		
5 / 9		
2 ** 4		
abs(-2) ** 4		
math.pow(2, 4)		
import math		
math.pow(2, 4)		
sqrt(4)		
math.sqrt(4)		
math.cos(0)		
math.pi		
math.sin(math.pi / 2)		

Questions (15 min) **Start time: __________**

1. In the middle “Predicted output” column, write what value you expect will be displayed, based on your team’s experience using a calculator. If there are any lines you are not confident about, place an asterisk next to your predicted output.

2. Open a Python Shell on your computer. Type each Python expression at the prompt, one line at a time, and write the corresponding Python output in the third column above. If an error occurs, write what type of error it was (i.e., the first word of the last line of the error message).

3. What does the ** operator do?

4. Based on the Python code in Model 1, identify four examples of:

a) mathematical operator

b) mathematical function

5. For addition and multiplication to produce an output with a decimal value, what type of number must be part of the input? Provide justification for your team's answer.

6. Does division follow the same rule as in #5? Provide justification for your team's answer.

7. The output of Model 1 displayed three different errors. Explain the reason for each:

a) TypeError

b) 1st NameError

c) 2nd NameError

8. Identify two differences between using a Python built-in function (e.g., `abs()`) and a function from the `math` module.

Model 2 Dividing Numbers

Table A

9 / 4	*evaluates to*	2.25
10 / 4	*evaluates to*	2.5
11 / 4	*evaluates to*	2.75
12 / 4	*evaluates to*	3.0
13 / 4	*evaluates to*	3.25
14 / 4	*evaluates to*	3.5
15 / 4	*evaluates to*	3.75
16 / 4	*evaluates to*	4.0

Table B

9 // 4	*evaluates to*	2
10 // 4	*evaluates to*	2
11 // 4	*evaluates to*	2
12 // 4	*evaluates to*	3
13 // 4	*evaluates to*	3
14 // 4	*evaluates to*	3
15 // 4	*evaluates to*	3
16 // 4	*evaluates to*	4

Table C

9 % 4	*evaluates to*	1
10 % 4	*evaluates to*	2
11 % 4	*evaluates to*	3
12 % 4	*evaluates to*	0
13 % 4	*evaluates to*	1
14 % 4	*evaluates to*	2
15 % 4	*evaluates to*	3
16 % 4	*evaluates to*	0

Questions (15 min) **Start time: __________**

9. For each operator in Model 2, identify the symbol and describe the type of numerical result.

10. If the result of the / operator were rounded to the nearest integer, would this be the same as the result of the // operator? Explain how the results in Table A compare to Table B.

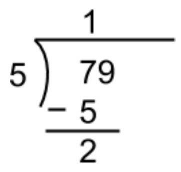
11. If the table included more rows, list all numbers // 4 would evaluate to 2 and all the numbers // 4 would evaluate to 4.

12. Based on the results of Table C, propose another number % 4 evaluates to 0, and explain what all these numbers have in common.

13. Consider the expressions in Table C that evaluate to 1. How do the left ***operands*** in these expressions (i.e., 9, 13) differ from those that evaluate to 0?

14. Describe the reason for the repeated sequence of numbers (0, 1, 2, 3) for the result of % 4.

15. Recall how you learned to do long division in elementary school. Finish solving for 79 ÷5 below. Which part of the answer is `79 // 5`, and which part is `79 % 5`?



16. Imagine that you are given candy mints to divide evenly among your team members.

a) If your team receives 11 mints, how many mints would each student get, and how many are left over? Write a Python expression to compute each result.

b) If your team receives 2 mints, how many mints would each student get, and how many are left over? Write a Python expression to computers this result.

17. Python has three division operators: "floor division", "remainder", and "true division". Which operator (symbol) corresponds to each name?

Model 3 Integers and Floats

Every value in Python has a ***data type*** which determines what can be done with the data. Enter the following code, one line at a time, into a Python Shell. Record the output for each line (if any) in the second column.

Python code	Shell output
integer = 3	
type(integer)	
type("integer")	
pi = 3.1415	
type(pi)	
word = str(pi)	
word	
number = float(word)	
print(word * 2)	
print(number * 2)	
print(word + 2)	
print(number + 2)	
euler = 2.7182	
int(euler)	
round(euler)	

Questions (15 min) **Start time: __________**

18. What is the data type (int, float, or str) of the following values? (Note: if you're unsure, use the `type()` function in a Python Shell.)

a) pi

b) integer

c) word

d) number

19. List the function calls that convert a value to a new data type.

20. How does the behavior of the operators (+ and *) depend on the data type?

21. What is the difference between the `int()` function and the `round()` function?

22. What is the value of `3 + 3 + 3`? What is the value of `.3 + .3 + .3`? If you enter these expressions into a Python Shell, what do you notice about the results?

23. In order to store a number with 100% accuracy, what data type is required? How might you precisely represent a bank account balance of $123.45?

24. Try calculating a very large integer in a Python Shell, for example, 123^{456}. Is there a limit to the integers that Python can handle?

25. Try calculating a very large floating-point number in a Python Shell, for example, 123.0^{456}. Is there a limit to the floating-point numbers that Python can handle?

26. Summarize the difference between the numeric data types (`int` and `float`).

Basic Data Structures

Python has a wide variety of built-in types for storing anything from numbers and text (e.g., int, float, str) to common data structures (e.g., list, tuple, dict).

Content Learning Objectives

After completing this activity, students should be able to:

- Reference a specific element of a sequence by an index.
- Compare and contrast numeric and sequence data types.
- Create a dictionary of strings and look up values by key.

Process Skill Goals

During the activity, students should make progress toward:

- Providing feedback on how well other team members are working. (Teamwork)

Model 1 Lists

A variable can hold multiple values in the form of a ***list***. The values are separated by commas and wrapped in square brackets. For example:

```
primes = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29]
```

Each ***element*** of the list can be referenced by an ***index***, which is the sequential position starting at 0. For example, `primes[4]` is 11.

index	0	1	2	3	4	5	6	7	8	9
value	2	3	5	7	11	13	17	19	23	29

Do not type anything yet! Read the questions first!

Python code	Shell output
odd = [1, 3, 5, 7]	
odd	
odd[2]	
odd[4]	
len(odd)	
number = odd[1]	
number	
odd[1] = 2	
odd	
number	

Questions (10 min) Start time: __________

1. What is the index of the second element of `primes`? What is the value at that index?

2. How does the index number compare to the position of the element?

3. Type each line of code in a Python Shell and write the corresponding output in the space above. If an error occurs, write what type of error. Place an asterisk (*) next to any output for which you were surprised.

4. How did you reference the value of the 3rd element of odd?

5. What did the output of the len() function tell you about the list?

6. The output of Model 1 displayed an error. Explain the reason for the error.

7. Write a statement that assigns a list of three integers to the variable run.

8. Write a statement that assigns the value 100 to the last element of run.

9. Write a statement that assigns the first value of runto a variable named first.

Model 2 Sequences

Lists and strings are examples of ***sequence*** types. Complete the table below to explore how sequences work.

Python code	Shell output
seq1 = "one two"	
type(seq1)	
len(seq1)	
seq1[1]	
seq1[1] = '1'	
seq2 = "one", "two"	
type(seq2)	
len(seq2)	
seq2[1]	
seq2[1] = '1'	
seq3 = ["one", "two"]	
type(seq3)	
seq3[1]	
seq3[1] = 1	
seq4 = ("one", 1)	
type(seq4)	
number = 12345	
number[3]	

Questions (15 min) **Start time: __________**

10. How does a sequence type differ from a number? (See the last row of the table.)

11. What are the names of the three sequence types introduced in Model 2?

12. How does the syntax of creating a tuple differ from creating a list?

13. Is there more than one way (syntax) to create a tuple? Justify your answer.

14. Which sequence types allow their elements to be changed? Which do not?

15. Is it possible to store values of different types in a sequence? If yes, give an example from the table; if no, explain why not.

16. Summarize the difference between lists and tuples. How do they look differently, and how do they work differently?

Model 3 Dictionaries

In Python, a ***dictionary*** stores "key: value" pairs. The pairs are separated by commas and wrapped in curly braces. For example:

```
elements = {C: carbon, H: hydrogen, O: oxygen, N: nitrogen}
```

Key	Value
C	carbon
H	hydrogen
O	oxygen
N	nitrogen

In contrast to sequence types, a dictionary is a ***mapping*** type. Values are referenced by ***keys***, rather than by indexes.

Type the elementsdictionary above into a Python Shell, and then complete the following table to explore how it works.

Python code	Shell output
type(elements)	
elements.keys()	
elements.values()	
elements['C']	
atom = 'N'	
elements[atom]	
elements[N]	
elements['nitrogen']	
elements[1]	
len(elements)	
elements['B'] = 'Boron'	
elements.items()	

Questions (20 min) **Start time: __________**

17. List all the keys stored in the elementsdictionary after completing the table.

18. What is the data type of the keys in the elementsdictionary?

19. Explain the reason for the error after entering each of the following lines:

a) `elements[N]`

b) `elements["nitrogen"]`

c) `elements[1]`

20. Ignoring the ”dict_items()” part, describe the contents and type of data returned by the items() method.

21. Write a Python expression that creates a dictionary for the seven days of the week, i.e., Sun=1, Mon=2, Tue=3, etc. Assign the dictionary to the variable dow.

22. If you assign two different values to the same key (i.e., two assignment statements with one value each), which value is stored in the dictionary? Justify your answer with an example.

23. Another way to store the data in Model 3 is to use two lists:

```
keys = ["C", "H", "O", "N"]
vals = ["carbon", "hydrogen", "oxygen", "nitrogen"]
```

What is a disadvantage of this approach? Explain your reasoning.

Conditions and Logic

Computer programs make decisions based on logic: if some condition applies, do something, otherwise, do something else.

Content Learning Objectives

After completing this activity, students should be able to:

- Evaluate boolean expressions with comparison operators (<, >, <=, >=, ==, !=).
- Explain the syntax and meaning of if/elsestatements and indented blocks.
- Evaluate boolean expressions that involve comparisons with and, or, and not.

Process Skill Goals

During the activity, students should make progress toward:

- Evaluating complex logic expressions based on operator precedence. (Critical Thinking)

Model 1 Comparison Operators

In Python, a comparison (e.g., `100 < 200`) will yield a ***Boolean*** value of either `True` or `False`. Most data types (including `int`, `float`, `str`, `list`, and `tuple`) can be compared using the following operators:

Operator	Meaning
<	less than
<=	less than or equal
>	greater than
>=	greater than or equal
==	equal
!=	not equal

Type the following code, one line at a time, into a Python Shell. Record the output for each line (if any) in the second column.

Python code	Shell output
type(True)	
type(true)	
type(3 < 4)	
print(3 < 4)	
three = 3	
four = 4	
print(three == four)	
check = three > four	
print(check)	
type(check)	
print(three = four)	
three = four	
print(three == four)	

Questions (10 min) **Start time: ___________**

1. What is the name of the data type for Boolean values?

2. Do the words Trueand Falseneed to be capitalized? Explain how you know.

3. For each of the following terms, identify examples from the table in Model 1:
   a) Boolean variables:

   b) Boolean operators:

   c) Boolean expressions:

4. Explain why the same expression three == fourhad two different results.

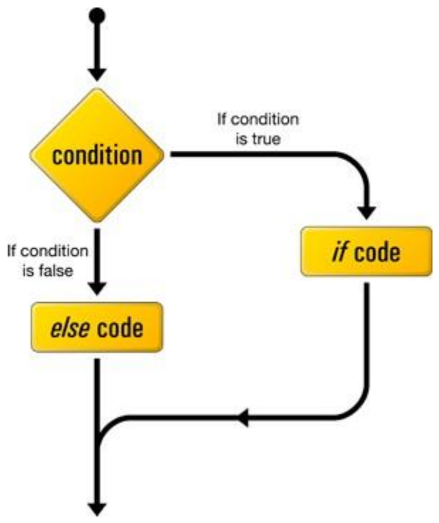
5. What is the difference between the =operator and the ==operator?

6. Write a Boolean expression that uses the !=operator and evaluates to False.

Model 2 if/else Statements

An ifstatement makes it possible to control what code will be executed in a program, based on a condition. For example:

```
number = int(input("Enter an integer: ")) if number <
0:
        print(number, "is negative") else:
        print(number, "is a fine number") print("Until
next time...")
```



Python uses ***indentation*** to define the structure of programs. The line indented under the ifstatement is executed only when number < 0is True. Likewise, the line indented under the else statement is executed only when number < 0 is False. The flowchart on the right illustrates this behavior.

Questions (15 min) **Start time: __________**

7. What is the Boolean expression in Model 2?

8. Enter this short program into a Python Editor. What is the output when the user enters the number 5? What is the output when the user enters the number -5?

9. After an if-condition, what syntax differentiates between (1) statements that are executed based on the condition and (2) statements that are always executed?

10. Enter the line ␣␣␣␣print("Hello")into a Python Editor (where ␣ is a space), save the file as hello.py, and run the program. What happens if you indent code inconsistently?

11. Based on the program in Model 2, what must each line preceding an indented block of code end with?

12. Write an ifstatement that first determines whether numberis even or odd, and then prints the message "(number) is even"or "(number) is odd". (Hint: use the %operator.)

13. Does an ifstatement always need to be followed by an elsestatement? Why or why not? Give an example.

Model 3 Boolean Operations

Expressions may include Boolean operators to implement basic logic. If all three operators appear in the same expression, Python will evaluate not first, then and, and finally or. If there are multiple of the same operator, they are evaluated from left to right.

Do not type anything yet! Read the questions first!

Python code	Predicted output	Actual output
print(a < b and b < c)		
print(a < b or b < c)		
print(a < b and b > c)		
print(a < b or b > c)		
print(not a < b)		
print(a > b or not a > c and b > c)		

Questions (20 min) **Start time: __________**

14. What data type is the result of a < b? What data type is the result of a < b and b < c?

15. Predict the output of each print statement, based on the variables a = 3, b = 4, and c = 5. Then execute each line in a Python Shell to check your work.

16. Based on the variables in #15, what is the value of a < b? What is the value of b < c?

17. If two True Boolean expressions are compared using the and operator, what is the resulting Boolean value?

18. Using the variables defined in #15, write an expression that will compare two False Boolean expressions using the or operator. Check your work using a Python Shell.

19. Assuming P and Q each represent a Boolean expression that evaluates to the Boolean value indicated, complete the following table. Compare your team's answers with another team's, and resolve any inconsistencies.

P	Q	P and Q	P or Q
False	False		
False	True		
True	False		
True	True		

20. Assume that two Boolean expressions are compared using the and operator. If the value of the first expression is False, is it necessary to determine the value of the second expression? Explain why or why not.

21. Assume that two Boolean expressions are compared using the or operator. If the value of the first expression is True, is it necessary to determine the value of the second expression? Explain why or why not.

22. Examine the last row of the table in #15. Evaluate the Boolean expression following the order of precedence rules explained in Model 3. Show your work by rewriting the line at each step and replacing portions with either True or False.

```
a > b or not a > c and b > c
```

23. Suppose you wanted to execute the statement sum = x + y only when both x and y are positive. Determine the appropriate operators, and write a single Boolean expression for the if-condition.

24. Rewrite the expression from #23 using the `not` operator. Your answer should yield the same result as in #23, not the opposite. Describe in words what the new expression means.

25. Suppose that your team needs to execute the statement `sum = x + y` except when both `x` and `y` are positive. Write a Boolean expression for this condition. How is it different from the previous question?

Loops and Iteration

A loop allows you to execute the same statements multiple times. Python has two kinds of loop structures: `for` loops, which iterate over the items of a sequence, and `while` loops, which continue to execute as long as a condition is true.

Content Learning Objectives

After completing this activity, students should be able to:

- Explain the syntax and the purpose of a `for` statement.
- Predict how `range()` works given 1, 2, or 3 arguments.
- Identify the three main components of a `while` loop.

Process Skill Goals

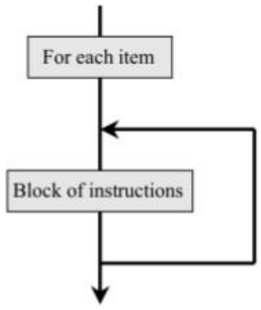
During the activity, students should make progress toward:

- Tracing the execution of while/for loops and predict their final output. (Critical Thinking)

Model 1 for Statements

A for loop executes the same block of code "for each item in a sequence". Create a new file named loops.py, and enter the following code:

```
print("hello")
for x in [2, 7, 1]: print("the number
        is", x)
print("goodbye")
```



Questions (15 min) **Start time: __________**

1. Run the loops.py program. How many times does the indented line of code execute under the for loop?

2. How many times does the line of code NOT indented execute after the for loop?

3. Identify the value of x each time the indented line of code is executed.

   a) 1st time:

   b) 2nd time:

   c) 3rd time:

4. Modify the list [2, 7, 1] in the following ways, and rerun the program each time. Indicate how many times the for loop executes.

   a) non-consecutive numbers: [5, -7, 0]

   b) numbers decreasing in value: [3, 2, 1, 0]

   c) all have the same value: [4, 4]

   d) single value in a list: [8]

5. In general, what determines the number of times that the loop repeats?

6. What determines the value of the variable `x`? Explain your answer in terms of what is assigned (`x = …`) each time the loop runs.

7. Modify the program as follows:

   a) Write a statement that assigns `[0, 1, 2, 3, 4]`to the variable `numbers`.

   b) Rewrite the `for x …`statement to use the variable `numbers`instead.

   c) Does the assignment need to come before or after the `for`statement?

8. Add the following code at the end of your program:

```
for c in "Hi!": print(c)
```

   a) What is the output of this `for`statement?

   b) What determined how many times `print(c)`was called?

   c) Explain what a `for`statement does with strings.

9. What other data types (besides lists and strings) can a `for`loop handle? Experiment by adding examples to your `loops.py`program. Summarize here what works and what doesn't.

Model 2 The range Function

The Python range function will generate a list of numbers. The range function can take up to three numbers as arguments. Fill in the table below by typing the code into a Python Shell:

Python code	Shell output
range(5)	
list(range(5))	
x = range(3)	
print(x)	
print(list(x))	
list(range(5, 10))	
list(range(-3, 4))	
list(range(4, 10, 2))	
for i in range(5): print(i)	

Questions (15 min) **Start time: __________**

10. Explain the difference in output between the first two lines of code (with and without the list function).

11. If the argument of the range function specifies a single number (*x*):

a) What will be the first number listed?
b) What will be the last number listed?
c) How many numbers will be in the list?
d) Use the range function to generate the sequence 0, 1, 2, 3.

12. If the argument of the range function specifies two numbers (*x*, *y*):

a) What will be the first number listed?
b) What will be the last number listed?
c) How many numbers will be in the list?
d) Use the range function to generate the sequence 1, 2, 3, 4.

13. If the argument of the range function specifies three numbers (x, y, z):

a) What will be the first number listed?

b) What does the third argument represent?

c) How many numbers will be in the list?

d) Use the range function to generate the sequence 1, 3, 5, 7.

14. In your Editor, make a copy of the Model 1 code. Then modify the for statement so that the number of times the loop executes is determined by a variable named times.

a) How did you change the for statement?

b) How would you cause the loop to print the values 0 to 5?

15. Consider the two different types of for statements used in Model 1 and Model 2.

a) If you wanted to execute a loop 100 times, which type of for statement would you choose and why?

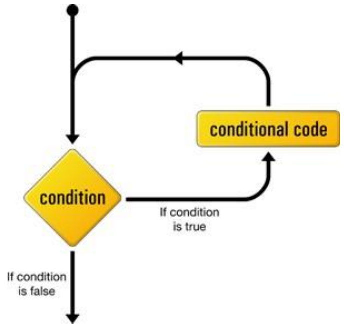
b) If you wanted to use each item of an existing list inside the loop, which type of for statement would you choose and why?

16. Does the range function work with strings? If so, show an example. If not, show how to print the letters A to Z in a loop.

Model 3 while Statements

A more general looping structure is the whilestatement. Add the code below to your current loops.pyprogram:

```
i = 0
while i < 3:
        print("the number is", i) i = i + 1
print("goodbye")
```



Questions (15 min) **Start time: __________**

17. What must the value of the Boolean expression (after the while) be in order for the first print statement to execute?

18. Circle the statement that changes the variable iin the above code.

19. What happens to the value of the variable iduring the execution of the loop?

20. Explain why the loop body does not execute again after it outputs “the number is 2”.

21. Reverse the order of the statements in the loop body:

```
while i < 3:
        i = i + 1
        print("the number is", i)
```

a) How does the order impact the output displayed by the printfunction?

b) Does the order impact the total number of lines that are output?

22. Identify three different ways to modify the code so that the loop only executes twice.

23. Describe the three parts of a `while`loop that control the number of times the loop executes.

24. Comment out the statement `i = i + 1`, and run the module. Then press Ctrl-C (hold down the Ctrl key and press C). Describe the behavior you see, and explain why it happened.

When writing a `while`*loop, it's helpful to answer a few questions before you start:*

- *What needs to be initialized before the loop?*
- *What condition must be true for the loop to repeat?*
- *What will change so that the loop eventually ends?*

25. Consider the function `add(n)`that prompts the user for *n* numbers and returns the sum of these values. For example, when `add(5)`is called, the user is asked to input five numbers. If the user inputs 3, 1, 5, 2, and 4, the function would return the value 15.

a) Describe the variable that needs to be initialized before the loop begins.

b) Describe the Boolean expression that must be true for the loop to continue.

c) Describe what will need to change so that the loop will eventually end.

d) Now list what else needs to happen inside the body of the loop for you to calculate the sum of the user input.

e) Given your previous answer, are there any other values that need to be initialized before the start of the loop?

Defining Functions

Python programs typically have one or more functions, each of which has one or more statements. Defining new functions allows you to break down a complex program into smaller blocks of reusable code.

Content Learning Objectives

After completing this activity, students should be able to:

- Explain the flow of execution from one function to another.
- Describe the syntax of function definitions and function calls.
- Write short functions that have parameters and return results.

Process Skill Goals

During the activity, students should make progress toward:

- Tracing the execution of functions with Python Tutor. (Information Processing)

Model 1 Flow of Execution

In addition to using Python's built-in functions (e.g., print, abs) and functions defined in other modules (e.g., math.sqrt), you can write your own functions.

```
 1  def model_one():
 2          word = input("Enter a word: ")
 3          L = len(word)
 4          ans = word * L
 5          print(ans)
 6
 7  def main():
 8          print("Starting  main...")
 9          model_one()
10          print("All  done!")
11
12  main()
```

Questions (20 min) **Start time: ___________**

1. Based on the program in Model 1:

   a) What is the Python keyword for defining a function?

   b) On what line is the model_onefunction defined? _______ called? _______

   c) On what line is the mainfunction defined? _______ called? _______

2. Open a web browser and go to PythonTutor.com. Click on "Visualize your code", and type (or paste) the program above. Make sure the line numbers match.

3. Click the "Visualize Execution" button. As you step through the program, pay attention to what is happening on the **left side** of the visualization.

   a) What does the **red** arrow indicate?

   b) What does the **green** arrow indicate?

4. Notice the order in which the program runs:

   a) After line 12 of the program executes (Step 3), what is the next line that executes?

   b) After line 9 of the program executes (Step 6), what is the next line that executes?

5. Go back to the beginning of the program execution. This time as you step through the program, pay attention to what changes on the **right side** of the visualization.

a) Describe what changes in the visualization after Step 1.

b) Describe what changes in the visualization after Step 2.

6. In general, what happens on the right side of the visualization when a function is called?

7. In terms of execution order, what is the effect of calling a function?

8. Draw the right side of the visualization for Step 11 in the space below.

9. Notice that the variable ans is printed from within the model_one function. What happens if you try to print(ans) inside the main function?

10. Explain what happened in the previous question in terms of frames in the visualization.

11. In the space below, write a definition for a function called str_to_list that prompts the user to enter a word. The function should convert the string to a list and print the list.

12. Edit the program in Python Tutor so that, instead of defining and calling the function model_one, it defines and calls the function str_to_list. Verify your changes by visualizing the execution, and draw a picture of the right side immediately after the list is printed.

Model 2 Passing Arguments

Instead of using input inside a function to get data, we can define a function to take a ***parameter*** (variable). When we call the function, we need to provide an ***argument*** (value). Change the program in Python Tutor as follows:

```
def model_two(word):
        ans = word * len(word)
        print(ans)

def main():
        print("Starting  main...")
        w = input("Enter a word: ")
        model_two(w)
        print("All  done!")

main()
```

Questions (15 min) **Start time: __________**

13. Underline the parameter in the model_two function definition, then circle each use of the parameter inside the function.

14. Find the model_two function call in main, and underline the argument being passed by the function call.

15. Visualize the execution of model_two until Step 8.

a) How does the frame for model_two at this point in the execution differ from the frame for model_one previously?

b) Write the implied assignment statement to show how the parameter word gets its value.

c) When a variable is used as an argument, does the name of the variable need to be the same as the parameter variable name?

16. Assume that `s1 = "Hi"`and `s2 = "ya"`. In the function call `model_two(s1 + s2)`:

a) What is the argument for the function call?

b) Write the implied assignment statement that happens during the call.

c) What will be the value of parameter `word`when `model_two`begins executing?

d) Predict the output that will be produced by the function call.

17. Review the two implied assignment statements that you have written. What exactly gets "passed" when you call a function?

18. Change `model_two`so that, instead of multiplying `word`by the length of `word`, it will mul- tiply by an integer passed as the second argument to the function. Write the new version of `model_two` in the space below. Use `times`for the name of the new integer parameter.

19. How does the call to `model_two`in `main`need to change so that it matches the new function definition? Give an example.

Model 3 Returning Values

Functions may optionally send a value back to the calling function using a return statement. Change the program in Python Tutor as follows:

```
1  def model_three(word):
2          ans = word * len(word)
3          return ans
4
5  def main():
6          print("Starting  main...")
7          w = input("Enter a word: ")
8          result = model_three(w)
9          print(result)
10         print("All  done!")
11
12 main()
```

Questions (10 min) **Start time: __________**

20. Aside from the function name, how does line 8 in Model 3 differ from line 8 in Model 2?

21. At what step number (in the simulation) has model_three completed its execution, but control has not yet returned to the main function?

In the space below, draw the frame for model_three after this step.

22. In general, what value will be returned by model_three?

23. What changes in the frame for mainat Step 12 of the execution?

24. Edit model_three and delete the return statement at the end of the function. Visualize the execution. What value is returned by a function when there is no return statement?

25. Edit model_three again, and add the return statement back to the end of the function. Then change line 8 so that model_three is still called but there is no assignment to result. What do you predict will happen in main after the model_three function call completes?

26. Why is a function that returns the value of a variable more useful than a function that simply prints the value of that variable?

Lists and Strings

Many interesting problems involve manipulating sequences of data. You've learned about lists and strings before, but this activity provides a more in-depth look at what they can do.

Content Learning Objectives

After completing this activity, students should be able to:

- Name four methods that lists provide, and describe what each method does.
- Explain the syntax and meaning of slice operations, with and without indexes.
- Name four methods that strings provide, and describe what each method does.

Process Skill Goals

During the activity, students should make progress toward:

- Gaining insight about data structures from many examples. (Information Processing)

Model 1 Working with Lists

Recall that a variable can hold multiple values in the form of a list. The values are separated by commas and wrapped in square brackets.

Lists have ***methods*** (built-in functions) that can be called using dot notation. For example, to add a new element to the end of a list, we can use the `append`method.

Python code	Shell output
rolls = [4, 6, 6, 2, 6]	
len(rolls)	
print(rolls[5])	
rolls.append(1)	
print(rolls)	
print(rolls[5])	
lucky.append(1)	
lucky = []	
print(lucky[0])	
lucky.append(5)	
print(lucky)	
print(lucky[0])	
rolls.count(6)	
rolls.remove(6)	
print(rolls)	
help(rolls.remove)	
help(rolls)	

Questions (15 min) **Start time: _________**

1. What is the result of calling the `append`method on a list?

2. What must be defined prior to using a method like `append`?

3. Explain why two lines in Model 1 caused an IndexError.

4. What is the result of calling the removemethod on a list?

5. Based on the helpoutput, name several list methods not shown in Model 1. Do not include methods that begin and end with two underscores (e.g., __add__).

6. Give one example of a list method that requires an argument and one that does not.

7. Describe the similarities and differences between using a list method like append and Python built-in functions like print.

8. Complete the function below (two lines are missing). It should prompt the user for numbers and build a list by adding one number at a time to the end of the list. The loop terminates when the user inputs the number 0.

```
def input_numbers(): x =
    1

    ______________________

    while x != 0:
        x = int(input("Enter the next number: "))

        __________________________________

    return numbers
```

Model 2 Indexing and Slicing

A string is a sequence of characters in single quotes (') or double quotes ("). Depending on the application, we can treat a string as a single value (e.g., dna), or we can access individual characters using square brackets (e.g., dna[0]). We can also use ***slice notation*** (e.g., dna[4:8]) to refer to a range of characters. In fact, all types of sequences (including list and tuple) support indexing and slicing.

Python code	Shell output
dna = 'CTGACGACTT'	
dna[5]	
dna[10]	
len(dna)	
dna[:5]	
dna[5:]	
dna[5:10]	
triplet = dna[2:5]	
print(triplet)	
dna[-5]	
dna[-10]	
dna[:-5]	
dna[-5:]	
triplet = dna[-4:-1]	
print(triplet)	

Questions (15 min)

Start time: __________

9. What is the *positive* index of each character in the dnastring? Check your answers above.

Character:	C	T	G	A	C	G	A	C	T	T
Index:										

10. What is the *negative* index of each character in the dnastring? Check your answers above.

Character:	C	T	G	A	C	G	A	C	T	T
Index:										

11. Based on the previous questions, what are `dna[2]`and `dna[-2]`? Explain your answers.

12. Explain the IndexError you observed. What is the range of indexes for the `dna`string?

13. Consider the notation of the operator `[m:n]`for slicing the string.

a) Is the value at `m`the same as the corresponding index value (i.e., `dna[m]`)? If not, describe what it means.

b) Is the value at `n`the same as the corresponding index value (i.e., `dna[n]`)? If not, describe what it means.

c) Explain what it means when only a single number is referenced when creating a slice, such as `[m:]`or `[:n]`.

14. What is the simplest way to get the first three characters of `dna`? What is the simplest way to get the last three characters?

15. Write a Python expression that slices `GACT` from `dna`using positive indexes. Then write another expression that slices the same string using negative indexes.

16. Write a Python assignment statement that uses the `len`function to assign the last letter of `dna`to the variable `last`.

17. Write a Python assignment statement that uses a negative index to assign the last letter of `dna`to the variable `last`.

Model 3 Common String Methods

Like lists, strings have ***methods*** (built-in functions) that can be called using dot notation. See https://docs.python.org/3/library/stdtypes.html#string-methods for more details.

Python code	Shell output
dna = 'CTGACGACTT'	
dna.lower()	
print(dna)	
lowercase = dna.lower()	
print(lowercase)	
dnalist = list(dna)	
print(dnalist)	
dnalist.reverse()	
print(dnalist)	
type(dna)	
dna = dna.split('A')	
print(dna)	
type(dna)	
dna.replace('C', 'g')	
print(dna[0])	
type(dna[0])	
dna[0].replace('C', 'g')	
print(dna)	

Questions (15 min) **Start time: __________**

18. Does the lowermethod change the contents of the dnastring? Justify your answer.

19. Describe the listfunction—what does list(dna)return in Model 3?

20. Why is it possible to call the `replace` method on `dna[0]` but not `dna`?

21. Name several other string methods not shown in Model 3. (Read the documentation.)

22. Consider the application of a method on a variable:

a) Does a string variable change after applying a method? Provide justification.

b) Does a list variable change after applying a method? Provide justification.

c) Identify the data type that is ***immutable*** (i.e., the value never changes).

23. Write a single statement to change the final contents of `dna` to `['CTG', 'cc', 'CTT']`. Confirm that your code works in a Python Shell.

24. Why do you think Python has a `replace` method for strings but not for lists?

Importing Modules

Python comes with an extensive library of built-in modules that make it easy to accomplish everyday tasks. With just a few lines of code, you can do anything from generating random numbers and drawing graphics to sending emails and accessing websites.

Content Learning Objectives

After completing this activity, students should be able to:

- Use the `random` module to generate random float and integer sequences.
- Explain the purpose of the common line `if __name__ == "__main__"`.
- Summarize several built-in modules, including `random` and `turtle`.

Process Skill Goals

During the activity, students should make progress toward:

- Navigating the Python standard library documentation. (Information Processing)

Model 1 Random Numbers

You can generate a sequence of numbers using the Python random module. A mathematical function is used to produce the sequence based on a ***seed*** value. (If no seed is given, the current system time is used.) The sequence is more accurately described as ***pseudorandom***, since its output is inherently predictable.

Python code	Shell output
import randint	
import random	
randint(1, 10)	
random.randint(1, 10)	
from random import randint	
randint(1, 10)	
seed(100)	
random.seed(100)	
random.random()	
random.random()	
random.seed(100)	
random.random()	
random.random()	

Questions (20 min) **Start time: _________**

1. What is the name of the module that must be imported before generating a random number?

2. Based on Model 1, what are the names of three functions defined in the random module?

3. Identify the syntax of the statement to import:

   a) a module

   b) a function

4. Identify the syntax of a function call assuming:

   a) the module was imported

   b) the function was imported

5. How could you eliminate the need for typing the word "random" twice (in a function call) to generate a random number?

6. Compare the shell output of your team with at least one other team. Describe the similarities and differences observed.

7. What is the effect on the random numbers generated after calling the `seed`method?

8. Describe one reason to set the same seed each time a program is run, and one reason to not use the `seed` method.

9. Run `random.random()`multiple times. Based on the results, describe:

   a) the range of numbers returned by the `random`function

   b) the nature of the distribution of numbers generated. (Do they appear clustered around a particular value, or are they spread out uniformly over the range?)

10. Run `random.randint(1, 10)`multiple times. Based on the results, describe:

    a) the range of numbers returned by the `randint`function

    b) the nature of the distribution of numbers generated. (Do they appear clustered around a particular value, or are they spread out uniformly over the range?)

Model 2 Multiple Modules

Create a new file move.py, and enter the code:

```
import random

def angle():
        number = random.randint(-90, 90)
        return number

print("in move:        __name__ ==",    __name__)

print("will always execute: angle ==", angle())

if __name__ == "__main__":
        print("only if True: angle ==", angle())
```

Run move.py, and record the output below.

Output Line 1	
Output Line 2	
Output Line 3	

Create a new file stop.py(in the same directory), and enter the code:

```
import move

print("in stop:        __name__ ==",    __name__)

print("from module: angle ==", move.angle())
```

Run stop.py, and record the output below. Draw an arrow from each line of output to its corresponding print statement in the code.

Output Line 1	
Output Line 2	
Output Line 3	
Output Line 4	

Questions (15 min) **Start time: __________**

11. Upon execution of move.py:
    a) what is the value of the variable __name__?

    b) does the output correspond solely to the print statements contained in this file?

12. Upon execution of stop.py:
    a) what is the value of the variable __name__from the print statement in move

    b) what is the value of the variable __name__from the print statement in stop

    c) does the output correspond solely to the print statements contained in this file?

13. What was the reason to include the import movestatement in stop.py?

14. Based on the output of stop.py, describe what happens (as a side effect) when another module is imported.

15. What line in move.pydid not print when stop.pywas executed? Why?

16. In order for the output of stop.pyto correspond solely to the print statements contained in stop.py, what modifications need to be made to move.py?

17. Describe what code in general to include inside if __name__ == "__main__", and why.

Model 3 Turtle Graphics

The turtle module can be used to create graphics. Create a new file draw.py (in the same directory), and enter the following code. Run the program and see what happens.

```
import move
import turtle

def randomwalk(steps):
    turtle.shape("turtle")
    turtle.color("green")
    for i in range(steps):
        turtle.left(move.angle())
        turtle.forward(10)
    turtle.bye()

if __name__ == "__main__":
    randomwalk(100)
```

Questions (10 min) **Start time: __________**

18. For each outcome, describe the type of edit necessary to draw.pyand move.py:

a) a blue turtle

b) a longer simulation

c) a smaller range of angles (e.g., -45 to 45) that define the direction of the turtle

d) a random range of integers (e.g., 10 to 20) that define the length of a turtle move

19. Describe the type of edit necessary to produce the same outcome in Question #18d if the argument of forward is move.length() instead of 10:

20. Go to https://docs.python.org and click the modules link in the upper right corner. Find at least two built-in modules that interest you, and summarize what functions they provide.

Nested Structures

Containers are objects that store other objects. For example, list stores a sequence of objects, and dict stores a mapping of objects to objects. Containers can also hold other containers, which makes it possible to represent any type (or shape) of data.

Content Learning Objectives

After completing this activity, students should be able to:

- Explain how rows and columns of data can be stored in lists.
- Write nested for loops to iterate data and compute functions.
- Navigate a complex data structure with dictionaries and lists.

Process Skill Goals

During the activity, students should make progress toward:

- Developing algorithms that loop through lists to compute a result. (Problem Solving)

Model 1 Lists of Lists

Connect Four (® Hasbro, Inc.) is a two-player game in which the players take turns dropping colored discs into a six-row by seven-column grid. The objective of the game is to be the first player to form a horizontal, vertical, or diagonal line of four of one's own discs. (paraphrased from https://en.wikipedia.org/wiki/Connect_Four)

```
# current state of the game grid = [
        [·,·,·,·,·,·,·],
        [·,·,·,·,·,·,·],
        [·Y·,·,·,·,·Y·,·Y·,·],
        [·R·,·,·,·Y·,·R·,·R·,·],
        [·R·,·R·,·Y·,·R·,·Y·,·R·,·],
        [·R·,·Y·,·R·,·Y·,·Y·,·Y·,·R·],
]
```

Enter the grid code above into a Python Shell, and run each line of the table below. If the output is longer than one line, summarize it with a few words.

Python code	Shell output
print(grid)	
print(grid[5])	
print(grid[5][0])	
type(grid)	
type(grid[5])	
type(grid[5][0])	
len(grid)	
len(grid[5])	
len(grid[5][0])	
import pprint	
help(pprint)	
pprint.pprint(grid)	
for item in grid: print(item)	
for i in range(len(grid)): print(grid[i])	

Questions (15 min) Start time: __________

1. What does gridlook like when you first printit? (How is the output different from the original format shown in Model 1?)

2. What does gridlook like when you use pprintinstead? Explain what pprintmeans.

3. When viewed as a rectangle, how many “rows” and “columns” does gridhave?

4. What type of object is grid? What type of objects does it contain?

5. What type of object is grid[5]? What type of objects does it contain?

6. In the expression grid[5][0], which index corresponds to the row, and which index corre- sponds to the column?

7. Is grida list of rows or a list of columns? Justify your answer.

8. Describe how to append one more row to grid.

9. What is necessary to append a “column” to grid?

Model 2 Nested for Loops

Example A

We typically use a forloop to examine the contents of a list:

```
1  groceries = ["Apples", "Milk", "Flour", "Chips"]
2  for item in groceries:
3          print("Dont forget the", item)
```

Example B

If a list contains another list, we need a forloop that contains another forloop. For example, to count the “spaces” in the gridfrom Model 1:

```
4  count = 0
5  for row in grid:  # outer loop
6          print("row =", row)
7          for cell in row:  # inner loop
8                  print("cell =", cell)
9                  if cell == ..:
10                         count += 1
11 print(count, "spaces remaining")
```

Questions (15 min) Start time: ___________

10. As a team, discuss the two examples from Model 2. Predict how many times each of the following lines will execute. Then run the code and check your answers based on the output.

a) How many times does Line 3 execute? Predicted: ________ Actual: ________

b) How many times does Line 6 execute? Predicted: ________ Actual: ________

c) How many times does Line 8 execute? Predicted: ________ Actual: ________

d) How many times does Line 10 execute? Predicted: ________ Actual: ________

11. What determined how many times the “for item” loop would run?

12. Answer the following questions in terms of grid.

a) What determined how many times the “for row” loop would run?

b) What determined how many times the “for cell” loop would run?

13. In the example below, predict how many times the print statement will execute. Then run the code to verify your answer.

```
for i in range(6):  for j in
          range(7):
                    print(i, '+', j, '=', i+j)
```

14. Rewrite the nested for loops in Model 2 Lines 4–10 using the range function. Replace the variables row and cell with i and j, respectively. For simplicity, you may omit the print statements in your answer.

15. Write a for loop (using range) that computes the factorial of a given integer n. Recall that $n! = n * (n-1) * (n-2) * ... * 1$. Store your result in a variable named fact.

16. Write nested loops that compute and display the factorial of each integer from 1 to 20. Use your code from the previous question as the inner loop. Your output should be in this format:

```
The factorial of 1 is 1
The factorial of 2 is 2
The factorial of 3 is 6
The factorial of 4 is 24
The factorial of 5 is 120
```

Model 3 Nested Dictionaries

Containers can be nested in arbitrary ways. For example, the following data could be described as a "dictionary of dictionaries of integers and lists of strings".

Enter the following code into a Python Shell, and complete the table. If the output is longer than one line, summarize it with a few words.

```
movies = {
    "Casablanca": {
        "year": 1942,
        "genres": ["Drama", "Romance", "War"],
    },
    "Star Wars": {
        "year": 1977,
        "genres": ["Action", "Adventure", "Fantasy"],
    },
    "Groundhog Day": {
        "year": 1993,
        "genres": ["Comedy", "Fantasy", "Romance"],
    },
}
```

Python code	Shell output
movies	
movies["Casablanca"]	
movies["Casablanca"]["year"]	
movies["Casablanca"]["genres"]	
type(movies)	
type(movies["Casablanca"])	
type(movies["Casablanca"]["year"])	
type(movies["Casablanca"]["genres"])	
len(movies)	
len(movies["Casablanca"])	
len(movies["Casablanca"]["year"])	
len(movies["Casablanca"]["genres"])	
for key in movies: print(key)	
for key, val in movies.items(): print(key, val)	

Questions (15 min) Start time: __________

17. Explain the TypeErroryou encountered.

18. In the expression movies["Casablanca"]["genres"], describe the purpose of the strings "Casablanca"and "genres".

19. When iterating a dictionary using a forloop (i.e., for x in movies), what gets assigned to the variable?

20. What is wrong with the following code that attempts to printeach movie?

```
for i in range(len(movies)):
    print(movies[i])
```

21. Write nested loops that output every *genre* found under the moviesdictionary. You should have nine total lines of output.

22. Each movie in Model 3 has a title, a year, and three genres.

a) Is it necessary that all movies have the same format?

b) Name one advantage of storing data in the same format:

c) Show how you would represent The LEGO Movie (2014) with a runtime of 100 min and the plot keywords “construction worker” and “good cop bad cop”.

File Input/Output

Most data is stored in files, not input by the user every time. In this activity, you'll learn the basics of reading and writing plain text files.

Content Learning Objectives

After completing this activity, students should be able to:

- Create a new text file, and output several lines to it.
- Open an existing file, and append several lines to it.
- Read a text file line by line, and extract data from it.

Process Skill Goals

During the activity, students should make progress toward:

- Justifying answers based on the results of an experiment. (Critical Thinking)

Model 1 Writing to a File

The following example creates a new file (in the current/default folder) named out.txtand writes several lines of output to it. Run the code, and examine the contents of the resulting out.txtfile. In the space below, write the contents of out.txtto the right of the code.

```
outfile = open("out.txt", "w")
outfile.write("Example ")
outfile.write("output ")
outfile.write("text file\n")
outfile.write("xyz Coordinates\n")
outfile.write("MODEL\n")
outfile.write("ATOM %3d" % 1)
seq = "n %5.1f%5.1f%5.1f" % (0, 1, 2)
outfile.write(seq)
outfile.write("\n")
outfile.close()
```

out.txt

Questions (15 min) **Start time: __________**

1. Based on the Python code:

   a) How many arguments are passed to open? What are their types?

   b) What variable stores the ***fIle object*** returned by the openfunction?

   c) Identify the names of all methods used on this file object in the code.

   d) What type of data does the writemethod require for its argument?

2. Based on the out.txtfile:

   a) How many times was the writemethod called to create the first line of text?

   b) How many times was the write method called to create the second line of text?

   c) What does the "\n"character do?

   d) How is the writemethod different from the printfunction?

3. Write a program that creates a file named lines.txtand writes 100 lines like this:

```
Line #1
Line #2
Line #3
Line #4
...
```

Model 2 Appending to a File

The second argument of open specifies the ***mode*** in which the file is opened. When writing output to a file, there are two basic modes:

- The write ("w") mode will overwrite/replace the file contents.
- The append ("a") mode will add new data to the end of the file.

Either mode will create the file automatically if it does not already exist. Enter the following lines into a Python Shell, and record the output at each step.

Python code	Shell output
afile.write("new line\n")	
afile = open("out.txt", "a")	
afile.write("new line\n")	
afile.write(2.0)	
afile.write("2.0")	
afile.close()	
afile.write("new line\n")	

Questions (10 min) **Start time: __________**

4. Explain what happens as a result of the line: afile = open("out.txt", "a")

5. How do the arguments passed to the open function differ for writing a new file in comparison to appending an existing file?

6. What does the write method return? Run help(afile.write) to check your answer.

7. Explain the reason for the error observed after entering:

   a) the first line of code: afile.write("new line\n")

   b) the last line of code: afile.write("new line\n")

   c) the statement: afile.write(2.0)

Model 3 Reading from a File

Programs often require input data from an external file source. Not surprisingly, there are methods for reading the contents of files. Enter the following lines into a Python Shell.

Python code	Shell output
`infile = open("out.txt", "r")`	
`infile.readline()`	
`infile.readline()`	
`infile.readlines()`	
`infile.readline()`	
`infile.close()`	
`infile = open("out.txt", "r")`	
`for line in infile: print(line)`	
`infile.close()`	
`infile = open("out.txt", "r")`	

`for i in range(3):`
`    infile.readline()`	

`line = infile.readline()`	
`line`	
`print(line[0])`	
`print(line[0:5])`	
`words = line.split()`	
`words`	
`print(words[0])`	
`infile.close()`	

Questions (20 min) **Start time: __________**

8. Based on the output above:

   a) What type of data does the readlinemethod return?

   b) What type of data does the readlinesmethod return?

9. Why did the readlinemethod return different values each time?

10. What happens if you try to read past the end of the file? Justify your answer.

11. What is the difference between the two forloops in Model 3?

12. Consider the output of the first forloop:

   a) Why does the program display the file as if it were double spaced?

   b) How would you change the code to avoid printing extra blank lines?

13. Based on the second half of Model 3:

   a) Why was it necessary to open the file again?

   b) Write code that would output 1.0 using line

   c) Write code that would output 1.0 using words

14. Consider a file names.txtthat contains first and last names of 100 people, with one name per line (e.g., "Anita Borg"). Write a program that prints all the last names (the second word of each line) in the file.

Visualizing Data

As a general-purpose programming language, Python is incredibly useful for analyzing data and visualizing results. This activity is a first look at matplotlib, one of the most widely used 2D plotting libraries.

Content Learning Objectives

After completing this activity, students should be able to:

- Explain the basic structure of code for plotting a mathematical function.
- Analyze visually the behavior of the Python random number generator.
- Read data from a CSV file and generate histograms of various columns.

Process Skill Goals

During the activity, students should make progress toward:

- Navigating the documentation for a third-party library. (Information Processing)

Model 1 Simple Plot

When analyzing data, it's helpful to create charts, plots, and other visualizations. Doing so allows you to see important numerical relationships. Enter the following code into a Python Editor, and run the program.

```
1  import matplotlib.pyplot as plt
2  import numpy as np
3
4  def model_one():
5      x = np.arange(0.0, 2.0, .01)
6      y = np.sin(2 * np.pi * x)
7      plt.plot(x, y)
8      plt.xlabel('time (s)')
9      plt.ylabel('volts (mV)')
10     plt.show()
11
12 if __name__ == "__main__":
13     model_one()
```

Questions (15 min) **Start time: __________**

1. Identify in the source code which line numbers:

   a) generated the data?

   b) set the axes properties?

   c) displayed the window?

   d) plotted the actual data?

2. Describe in your own words what is being plotted.

3. Modify the code to plot only one cycle of the sine wave (instead of two). Write the edited line of code below.

4. Change the third argument of np.arangefrom 0.01 to 0.15. What is the result?

5. Add "o"as a third argument to the plotfunction. What is the result?

6. How does the third parameter of np.arangeaffect how the plot looks?

7. How would you modify the code to plot the function $y = x^2 - 1$ instead? Show the results from -2 to +2.

8. Which three Python libraries are used in Model 1? Quickly search the Internet and find their websites. Write a one-sentence description about each library.

Model 2 Histograms

Recall that you can generate a sequence of numbers using the randommodule. Merge the code below into your program from Model 1. Run the program, and view the output.

```
import matplotlib.pyplot as plt
import random

def model_two(npts):
    numbers = []
    for _ in range(npts):
        numbers.append(random.random())
    plt.hist(numbers)
    plt.show()

if __name__ == "__main__":
    model_two(100)
```

Questions (10 min) **Start time: _________**

9. Based on the Python code:

   a) What is the range of values generated by the randomfunction?

   b) How many random values are generated?

10. Based on the figure plotted:

    a) How many bars are displayed?

    b) What is the width of each bar?

    c) What is the sum of the heights of the bars?

11. Based on your answers above, what are appropriate labels for the *x* and *y* axes?

12. Increase the argument of model_twoto 1000, 10000, and 100000. Describe how the output plot changes when you run the program.

13. Add the number 50 as second argument to the histfunction. What is the meaning of the result?

14. In general, describe what the histfunction does with the list of random numbers to create this type of plot.

Model 3 CSV Data

“Comma Separated Values” is a common file format when exporting data from spreadsheets and databases. Each line of the file is a row, and each column is separated by a comma. Cells that contain commas are wrapped in quote marks.

data.csv file contents:

```
Name,Location,URL,Students
Westminster College,"Salt Lake City, UT",westminstercollege.edu,2135 Muhlenberg
College,"Allentown, PA",muhlenberg.edu,2330
University of Maine,"Orono, ME",umaine.edu,8677
James Madison University,"Harrisonburg, VA",jmu.edu,19019 Michigan State
University,"East Lansing, MI",msu.edu,38853
```

Python includes a csv module (https://docs.python.org/3/library/csv.html) that makes it easy to read and write CSV files.

```
import csv

infile = open("data.csv") data =
csv.reader(infile)
names = next(data)  # column names for
row in data:
    print(row[1])          # 2nd column
```

Program output:

```
Salt Lake City, UT
Allentown, PA Orono,
ME Harrisonburg, VA
East Lansing, MI
```

Questions (20 min) **Start time: __________**

15. In the example data.csv file above:

a) In what way is the first line different?

b) How many rows of data are there?

c) How many columns are there?

16. Compare data.csv with the program output:

a) Are quote marks included in the lines of data?

b) Are quote marks included in the program output?

c) What is the purpose of the quote marks?

17. In the Python code above:

a) Which line of code reads the first line of the file?

b) What type of data does the variable row contain?

In 2013, the U.S. Department of Education released the "College Scorecard" website to help students and families compare institutions of higher education. The Scorecard data includes information like average cost of attendance, graduation and retention rates, student body demographics, etc.

18. Download the "Scorecard Data 7 MB CSV" from https://collegescorecard.ed.gov/data/ (listed halfway down under "Featured Downloads"). Open the CSV file in Excel or a similar program, and skim its contents.

a) How many rows does it have?

b) How many columns does it have?

19. Column CH is named UGDS, which means "Enrollment of undergraduate certificate / degree-seeking students".

a) What is the range of values in this column?

b) Which school has the most students enrolled?

c) Do all rows have an integer value for UGDS?

20. Based on the code in Model 2 and Model 3, write a program that plots a histogram of the UGDS column. Complete the following steps to consider each part of the program.

a) What two import statements will you need at the top?

b) What three statements prepare the csv file for reading?

c) What code is necessary to read the entire column into a list? (Note: Column CH in Excel is `row[85]` in Python.)

d) By default, data from text/csv files are read as strings. Write the code to convert the `row[85]`values to integers. Be sure not include the `"NULL"`values in the final list.

e) Write the last two lines that plot and show the histogram.

21. Run the program, and compare your results with another team's. What does the histogram tell you about undergraduate enrollments in the United States?

22. What other questions could you ask about this data? How would you answer them using histograms, line charts, and scatter plots?

Defining Classes

In this activity, we'll take a first look at object-oriented programming. Classes provide a means of bundling data and functionality together.

Content Learning Objectives

After completing this activity, students should be able to:

- Write a class definition that has several attributes and methods.
- Explain what a constructor is, when it is called, and what it does.
- Discuss what "object-oriented" means using concrete examples.

Process Skill Goals

During the activity, students should make progress toward:

- Developing and testing the design of a program incrementally. (Problem Solving)

Model 1   Attributes and Methods

Previously you have used built-in types like int, str, and list. Each of these types comes with its own ***methods***, such as isdigit and append. You can create new types of data, and methods to go with them, by defining a class. Classes specify the ***attributes*** (instance variables) and methods (functions) that each object belonging to the class will have.

```
class Atom:
    """An element from the periodic table."""

    def neutrons(self):
        """Returns the number of neutrons the element has"""
        number = self.isotope - self.atomic
        print("%s has %d neutrons" % (self.symbol, number))
        return number

    def grams_to_moles(self, grams):
        """Converts the mass of an element in grams to moles"""
        moles = grams / self.mass
        print("%.1f g is %.1f moles of %s" % (grams, moles, self.symbol))
        return moles

if  __name__ == "__main__":


    oxygen = Atom()  # create an Atom object
    oxygen.symbol = "O"
    oxygen.atomic = 8
    oxygen.mass = 15.999
    oxygen.isotope = 16
    carbon = Atom()  # create another Atom object
    carbon.symbol = "C"
    carbon.mass = 12.001
    oxygen.neutrons()
    oxygen.grams_to_moles(24)
    carbon.grams_to_moles(24)
```

Questions (15 min)   Start time: __________

1. Examine the ***class deflnition*** (the top half of the code):

   a) What is the name of the class?

   b) What are the names of the two methods?

   c) What is the name of the first parameter for all methods?

2. Now examine the `"__main__"`block of code:

   a) How many different `Atom`objects were created?

   b) Identify the variable name of each object.

   c) How many attributes were assigned in the `oxygen`object? List the names.

   d) How do the number of arguments for each method call differ from the number of parameters specified in the method definition?

3. How does the syntax referencing an attribute differ inside vs. outside the class definition?

4. When the `grams_to_moles`method is called (in the last two lines), what is the value of the `self` parameter?

5. Enter the expression `type(oxygen)`in a Python Shell. Explain the meaning and significance of the output.

6. Write code to create a new `Atom`object called `hydrogen`, and assign one of the attributes listed in Question #2c.

7. Call the `neutrons`method on carbon in a Python Shell. What is the reason for the error?

Model 2 Constructors

For each class defined, you can provide a ***constructor*** that initializes attributes of a new object. In Python, the constructor is always named __init__(with two underscores before and after the word init). The constructor is called automatically when you create a new object.

Add the following constructor to the top of your Atom class. By convention, the constructor is typically the first method in a class definition. Also edit the "__main__"block of code as shown.

```
class Atom:
    """An element from the periodic table."""

     def  __init__(self, symbol, atomic, mass, isotope=12):
         """Constructs an Atom with the given values.""" self.symbol
         = symbol
         self.atomic = atomic
         self.mass = mass
         self.isotope = isotope

    ... previous methods from Model 1 ...

if __name__ == "__main__":

    oxygen = Atom('O', 8, 15.999, 16)
    carbon = Atom('C', 6, 12.001)
    oxygen.neutrons() carbon.neutrons()
    oxygen.grams_to_moles(24)
    carbon.grams_to_moles(24)
```

Questions (15 min) **Start time: __________**

8. What is always the name of the constructor?

9. Although there is no direct call to the constructor, explain how you know this method is executed when an object is created.

10. Consider your answer to Question #7. What is one advantage of defining a constructor for a class?

11. In a Python Shell, try to create a new Atomobject called hydrogenwith only two arguments. Write your statement in the space below. What is the reason for the error you see?

12. When creating an object of the Atomclass, what is the value of isotopeif:

a) three arguments are given?

b) four arguments are given?

13. Print the value of self.isotopein a Python shell.

a) What is the reason for the error?

b) In order to eliminate this error, what should be printed instead?

14. For each line below, what is the value of self?

a) oxygen = Atom('O', 8, 15.999, 16)

b) carbon = Atom('C', 6, 12.001)

c) oxygen.neutrons()

d) carbon.neutrons()

15. Recall that a variable may be "local" (defined within a function), "global" (defined in the non-indented or "__main__"block of code), or "built-in" (part of Python itself).

a) Explain why the isotopeattribute is not a global variable.

b) Explain why the isotopeattribute is not a local variable.

c) How is each method of the class able to access the isotopeattribute?

Model 3 Object-Oriented

Edit the Atom class further to include the variable avogadros, the method grams_to_atoms, and the modified "__main__" block of code. Note that ***class variables*** (like avogadros) are typically defined before the __init__ method.

```
class Atom:
    """An element from the periodic table."""

    avogadros = 6.02E23

    ... previous methods from Model 2 ...

    def grams_to_atoms(self, weight):
        """Converts the mass of an element in grams to number of atoms.""" answer =
        Atom.avogadros * self.grams_to_moles(weight)
        print("%.1f g is %.1e atoms of %s" % (weight, answer, self.symbol)) return answer

if __name__ == "__main__":

    oxygen = Atom("O", 8, 15.999, 16)
    carbon = Atom("C", 6, 12.001)
    oxygen.neutrons() oxygen.isotope = 18
    oxygen.neutrons()
    oxygen.grams_to_atoms(24)
    carbon.grams_to_atoms(24)
```

Questions (15 min) Start time: ___________

16. Examine the grams_to_moles method (from Model 1):

a) Identify the three main variables used in grams_to_moles:

b) For each variable, what is its scope? (local or global)

17. What determines whether a variable is defined as an attribute or a local variable?

18. Now examine the grams_to_atoms method (from Model 3).

a) What variable was initialized in the Atom class outside the constructor and methods?

b) How does the syntax of a class variable differ from an attribute (instance variable)?

19. Would it be possible to rewrite the grams_to_atoms method as a global function instead? If so, explain how the function would differ.

20. How would you rewrite the line oxygen.grams_to_atoms(24) to call the global function defined in the previous question?

21. Consider the built-in str class:

a) Given the statement s = "Hello", what data is stored in the str object?

b) Show an example line of code that calls the upper method on the object s.

c) If the upper method were defined as a global function instead, how would you call it?

22. Based on the previous two questions, explain what the term "object-oriented" means.

23. Summarize the advantages you perceive for writing code as methods in classes instead of global functions.

Extending Classes

A major benefit of object-oriented programming is the ability to inherit classes and eliminate duplicate code. Inheritance allows you to define a new class based on an existing class.

Content Learning Objectives

After completing this activity, students should be able to:

- Read and interpret UML class diagrams for an existing code base.
- Evaluate pros and cons for designs with multiple similar classes.
- Define inheritance and demonstrate how to extend a base class.

Process Skill Goals

During the activity, students should make progress toward:

- Working with all team members to reach consensus on hard questions. (Teamwork)

Model 1 UML Class Diagrams

Unified Modeling Language (UML) provides a standard way of visualizing how programs are designed (http://www.uml.org/what-is-uml.htm). For example, a ***class diagram*** is a graphical summary of the attributes and methods of a class.

```
class Cow:

    def   __init__(self): self.position
          = (0, 0)
          self.respired = 0
          self.stomach = []

    def eat(self, item):
          if item == "grass":
                self.stomach.append(item)
                return "ate " + item
          else:
                return "This herbivore doesnt eat " + item

    def move(self, delta):
                         self.position = (self.position[0] + delta[0],
                                          self.position[1] + delta[1])
          return "This quadruped walked to " + str(self.position)

    def respire(self): self.respired
          += 1
          return "This cow respired through its nostrils."

    def speak(self):
          return "moo"
```

Cow
position : tuple respired : int stomach : list

__init__()
eat(item : str)
move(delta : tuple)
respire()
speak()

Questions (15 min) **Start time: __________**

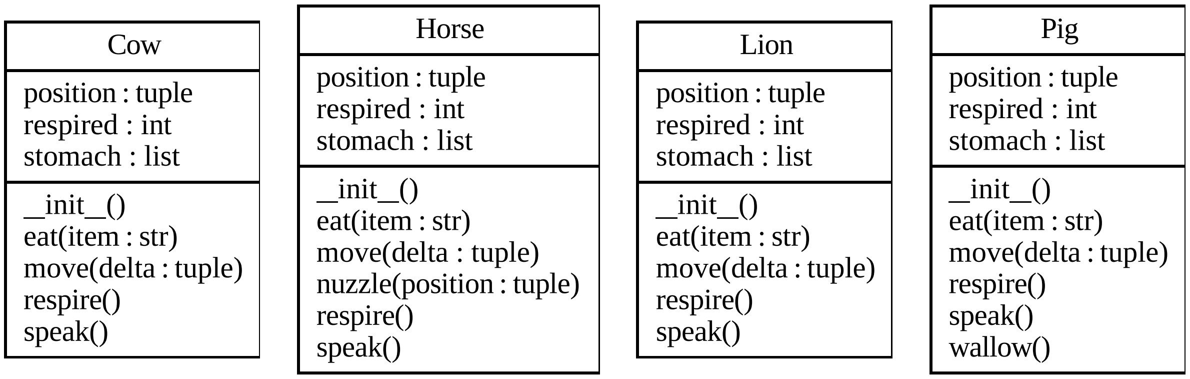
1. Draw an arrow from each name in the diagram to where it's defined in the code.

2. What are the attributes of `Cow`? What are the methods of `Cow`?

3. What is listed in each section of the UML class diagram?

   a) Top section: b) Middle section: c) Bottom section:

4. Consider the following class diagrams:



a) What attributes do the classes have in common?

b) What methods do the classes have in common?

c) What methods are unique to a particular class?

5. Quickly examine the source code for each of the classes to identify similarities and differences. Write one or two words in each table cell to summarize your findings.

	Cow	**Horse**	**Lion**	**Pig**
eat	grass			
move	walked			
nuzzle	N/A			
respire	nostrils			
speak	moo			
wallow	N/A			

6. Consider what it would take to add a new method named `sleep`to each of the classes.

a) Describe the process of adding the same method to each source file.

b) If a mistake is found later on, how would you correct the method?

c) What problems do you see with this approach as more classes are added?

Model 2 Single-Class Approach

Given that the classes from Model 1 are similar, we could try combining them into a single class. In order to keep track of differences, we would need to store additional attributes. We could provide the information from Question #5 when creating an object:

```
angus = Animal("Cow", "grass", "walked", "nostrils", "moo")
```

The UML diagram below outlines this approach. As a team, discuss this design and become familiar with the accompanying source code.

Animal
eats_only : str moved : str name : str nasal : str position : tuple respired : int sound : str stomach : list

__init__(name : str, eats_only : str, moved : str, nasal : str, sound : str)
eat(item : str)
move(delta : tuple)
nuzzle(position : tuple)
respire()
speak()
wallow()

Questions (10 min) **Start time: __________**

7. Circle the three original attributes that were defined in Model 1.

8. Circle the two methods that were NOT common to all four classes in Model 1.

9. Write a statement that creates an Animalobject representing a lion. Assign it to a variable named simba.

10. What methods does simbanow have that it did not have in Model 1?

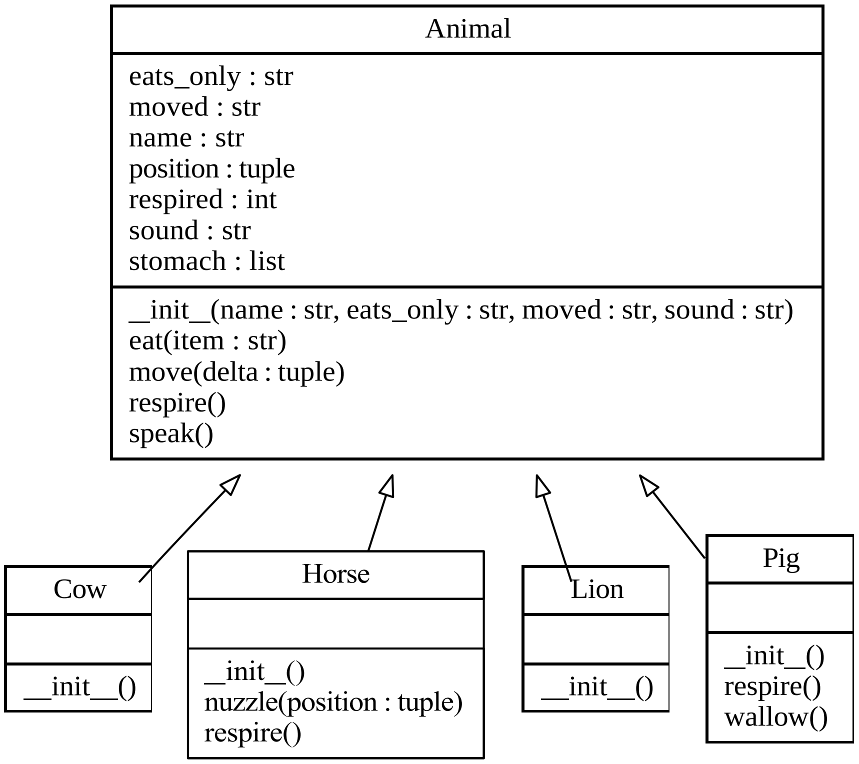
11. Describe 1–2 advantages the Model 2 design has compared to Model 1.

12. Describe 1–2 disadvantages the Model 2 design has compared to Model 1.

Model 3 Derived Classes

We can improve the code from Model 2 by using ***derived classes*** for Cow, Horse, Lion, and Pig. These classes only contain the attributes and methods specific to them. Animal is a ***base class*** that contains attributes and methods they all have in common. This language feature is known as ***inheritance***, because derived classes "inherit" attributes and methods from the base class.

The UML diagram below outlines this approach. As a team, discuss this design and become familiar with the accompanying source code.



Questions (20 min) **Start time: __________**

13. Open the lion.pysource file. How many methods are defined in the Lionclass? List the name of each one.

14. Type the following code into a Python Shell (in the same location as the source files). What methods are listed in the help?

```
from lion import Lion
help(Lion)
```

15. Write a statement that creates a Lionobject. Assign it to a variable named simba. How is this statement different from Question #9?

16. In a Python Shell, what is the value of simba.speak()? Where did this value come from?

17. Does simbahave any methods that it did not have in Model 1? Justify your answer.

18. Based on the source files, how does the __init__method of Animaldiffer from the __init__ methods of the derived classes?

19. What is the meaning of the built-in function super()that is used in the derived classes?

20. Describe 1–2 advantages the Model 3 design has compared to Model 2.

Recursive Functions

Sometimes when solving a problem, we can compute the solution of a simpler version of the same problem. Eventually we reach the most basic version, for which the answer is known.

Content Learning Objectives

After completing this activity, students should be able to:

- Identify the base case and recursive step of the factorial function.
- Trace a recursive function by hand to predict the number of calls.
- Write short recursive functions based on mathematical sequences.

Process Skill Goals

During the activity, students should make progress toward:

- Evaluating mathematical functions to gain insight on recursion. (Information Processing)

Model 1 Factorial Function

"In mathematics, the factorial of a non-negative integer n, denoted by $n!$, is the product of all positive integers less than or equal to n. For example, $5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$."

Source: https://en.wikipedia.org/wiki/Factorial

n	$n!$
0	1
1	1
2	2
3	6
4	24
5	120

Questions (15 min) **Start time: __________**

1. Consider how to calculate $4! = 24$.

   a) Write out all the numbers that need to be multiplied:
   $4! =$

   b) Rewrite the expression using $3!$ instead of $3 \times 2 \times 1$:
   $4! =$

2. Write expressions similar to #1b showing how each factorial can be calculated in terms of a smaller factorial. Each answer should end with a factorial (!).

   a) $2! =$

   b) $3! =$

   c) $100! =$

   d) $n! =$

3. What is the value of 0! based on Model 1? Does it make sense to define 0! in terms of a simpler factorial? Why or why not?

If we repeatedly break down a problem into smaller versions of itself, we eventually reach a basic problem that can't be broken down any further. Such a problem, like 0!, *is referred to as the* **base case**.

4. Consider the following Python function that takes *n* as a parameter and returns *n*!:

```
1  def factorial(n):
2      # base case
3      if n == 0:
4          return 1
5      # general case
6      product = 1
7      for i in range(n, 0, -1):
8          product *= i
9      return product
```

a) Review your answer to #2c that shows how to compute 100! using a smaller factorial. Convert this expression to Python by using the function above instead of the ! operator.

b) Now rewrite your answer to #2d in Python using the variable nand the function above.

c) In the source code above, replace the "1" on Line 6 with your answer from b). Then cross out Lines 7 and 8. Test the resulting function in a Python Shell. Does it still work?

d) What specific function is being called on Line 6?

e) Why is the ifstatement required on Line 3?

5. A function that refers to itself is called **recursive**. What two steps were necessary to define the recursive version of factorial?

6. Was a loop necessary to cause the recursive version of factorialto run multiple times? Explain your reasoning.

Model 2 Fibonacci Numbers

The Fibonacci numbers are a sequence where every number (after the first two) is the sum of the two preceding numbers: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, . . .

Source: https://en.wikipedia.org/wiki/Fibonacci_number

We can define a recursive function to compute Fibonacci numbers. Enter the following code into a Python Editor, and run the program to see the sequence.

```
1   def fibonacci(n):
2       # base case
3       if n == 1 or n == 2:
4           return 1
5       # general case
6       return fibonacci(n - 1) + fibonacci(n - 2)
7
8   if __name__ == "__main__":
9       for i in range(1, 6):
10          print(fibonacci(i))
```

Questions (10 min) **Start time: __________**

7. Based on the source code:

a) How many function calls are needed to compute `fibonacci(3)`? Identify the value of the parameter `n`for each of these calls.

b) How many function calls are needed to compute `fibonacci(4)`? Identify the value of the parameter `n`for each of these calls.

c) How many function calls are needed to compute `fibonacci(5)`? Identify the value of the parameter `n`for each of these calls.

8. Check your answers for the previous question by adding the following `print`statements to the code and rerunning the program:

- Insert `print("n is", n)`at Line 2, before the `# base case`comment
- Insert `print("fib(%d) is..." % i)`at Line 10, before the `print`statement

9. What happens if you try to compute `fibonacci(0)`in the Python Shell?

10. How could you modify the code so that this situation doesn't happen?

Model 3 Summation

"In mathematics, summation (capital Greek sigma symbol: Σ) is the addition of a sequence of numbers; the result is their sum or total."

$$\sum_{i=1}^{100} i = 1 + 2 + 3 + ... + 100 = 5050$$

Source: https://en.wikipedia.org/wiki/Summation

Questions (20 min) **Start time: __________**

11. Consider how to calculate $\sum_{i=1}^{4} i = 10$.

a) Write out all the numbers that need to be added:

$\sum_{i=1}^{4} i =$

b) Show how this sum can be calculated in terms of a smaller summation.

$\sum_{i=1}^{4} i =$

12. Write an expression similar to #11b showing how any summation of *n* integers can be calculated in terms of a smaller summation.

$$\sum_{i=1}^{n} i =$$

13. What is the base case of the summation? (Write the complete formula, not just the value.)

Here are important questions to consider before writing a recursive function:

- *How can you define the problem in terms of a smaller similar problem?*
- *What is the base case, where you solve an easy problem in one step?*
- *For the recursive call, how will you make the problem size smaller?*

To avoid infinite recursion, make sure that each recursive call brings you closer to the base case!

14. Implement a recursive function named `summation` that takes a parameter `n` and returns the sum $1+2+...+n$. It should only have an `if` statement and two `return` statements (no loops).

15. Enter your code into a Python Editor, and test the function. Make sure that `summation(100)` correctly returns 5050.

16. Implement a recursive function named `geometric` that takes three parameters (`a`, `r`, and `n`) and returns the sum "$a+ar+ar^2+ar^3...$" where $n+1$ is the total number of terms.

a) What is the base case?
`geometric(a, r, 0)` returns:

b) What is the recursive case?
`geometric(a, r, n)` returns:

c) Write the function in Python:

17. Enter your code into a Python Editor, and test the function. For example, if $a = 10$ and $r = 3$, the first five terms would be 10, 30, 90, 270, and 810. Make sure that `geometric(10, 3, 4)` correctly returns 1210 (the sum of those five terms).

Appendix

MANAGER

- **Helps the team get started quickly and remain focused.**
  - *"I think we have everything; are we ready to begin?"*
  - *"We're getting off topic; could we talk about that later?"*

- **Takes care of time management; keeps an eye on the clock.**
  - *"I think we need to focus on ______so we complete this section on time."*
  - *"Let's skip this question for now until we can ask the instructor for help."*
  - *"We have ______minutes before we need to discuss. Let's get this done."*

- **Makes sure that all voices in the team are heard and respected.**
  - "*(Name), would you be willing to read question ______out loud?"*
  - *"(Name), what do you think about our team's answer to _____?"*

PRESENTER

- **Communicates questions and clarifications with the instructor and other teams.**
  - *"Our team is confused about how ______relates to _____."*
  - *"Would you explain what question ______means by _____?"*

- **Ensures that all team members reach consensus before asking outside sources.**
  - *"Does anyone in our team know the answer for _____?"*
  - *"Before we ask the instructor, could someone clarify _____?"*
  - *"Does everyone agree that we need to find out _____?"*

- **Presents conclusions of the team to the class, when requested by the instructor.**
  - *"How should I explain this idea when asked to report out?"*
      - *"Our team found the answer to number ______by _____."*

RECORDER

- **Records the important aspects of group discussions, insights, etc.**
    - *"This seems like an important conclusion to write down."*
    - *"Let's stop for a minute so I can get this into our report."*

- **Guides consensus building process; helps team agree on responses.**
    - *"Would you all agree that _____is a good answer for number _____?"*
    - *"Is our answer completely supported by the explanation we gave?"*
    - *"Would that response make sense to someone from another team?"*

- **Ensures that accurate revisions happen after class discussions.**
    - *"Lets go back and revise what we wrote down for question _____."*
    - *"What did other teams say that we should include in our report?"*

REFLECTOR

- **Observes team dynamics and behavior with respect to the learning process.**
    - *"I think what (name) said earlier is important; would you repeat that?"*
    - *"Let's wait for (name) to finish writing that down before we move on."*

- **Reports to the team periodically during the activity on how the team performs.**
    - *"We're doing really well right now by including all team members."*
    - *"I have a suggestion on how we could be more productive as a team."*
    - *"What process skills are we doing well? What do we need to improve?"*

- **Be ready to report to the entire class about how well the team is operating.**
    - *"Overall, how effective would you say that our team was today?"*
        - *"We found that when _____happens, it works better if we _____."*

Meta Activity: Team Roles

Decide who will be what role for today; we will rotate the roles each week. If you have only three people, one should have two roles. If you have five people, two may share the same role.

Manager:
Presenter:
Recorder:
Reflector:

Questions (15 min) **Start time: _________**

1. What is the difference between **bold** and *italics* on the role cards?

2. Manager: invite each person to explain their role to the team. Recorder: take notes of the discussion by writing down key phrases next to the table above.

3. What responsibilities do two or more roles have in common?

4. For each role, give an example of how someone observing your team would know that a person is <u>not</u> doing their job well.

- Manager:
- Presenter:
- Recorder:
- Reflector:

Meta Activity: Team Disruptions

Common disruptions to learning in teams include: talking about topics that are off-task, teammates answering questions on their own, entire teams working alone, limited or no communication between teammates, arguing or being disrespectful, rushing to complete the activity, not being an active teammate, not coming to a consensus about an answer, writing incomplete answers or explanations, ignoring ideas from one or more teammates.

Questions (10 min) Start time: ________

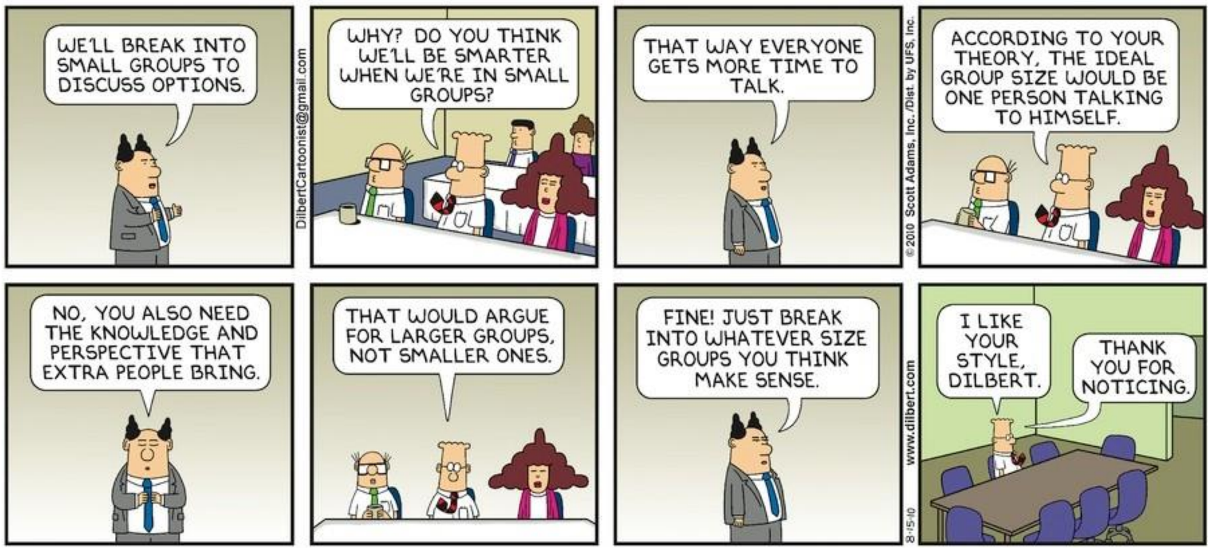
1. Pick four of the disruptions listed above. For each one, find something from the role cards that could help improve the team's success. Use a different role for each disruption.

a) Manager:

b) Presenter:

c) Recorder:

d) Reflector:


WE'LL BREAK INTO SMALL GROUPS TO DISCUSS OPTIONS.
WHY? DO YOU THINK WE'LL BE SMARTER WHEN WE'RE IN SMALL GROUPS?
THAT WAY EVERYONE GETS MORE TIME TO TALK.
ACCORDING TO YOUR THEORY, THE IDEAL GROUP SIZE WOULD BE ONE PERSON TALKING TO HIMSELF.
NO, YOU ALSO NEED THE KNOWLEDGE AND PERSPECTIVE THAT EXTRA PEOPLE BRING.
THAT WOULD ARGUE FOR LARGER GROUPS, NOT SMALLER ONES.
FINE! JUST BREAK INTO WHATEVER SIZE GROUPS YOU THINK MAKE SENSE.
I LIKE YOUR STYLE, DILBERT.
THANK YOU FOR NOTICING.
DilbertCartoonist@gmail.com
© 2010 Scott Adams, Inc. /Dist. by UFS, Inc.
www.dilbert.com
8-15-10

Dilbert by Scott Adams. © Andrews McMeel Syndication. http://dilbert.com/strip/2010-08-15

Meta Activity: What Employers Want

The following table is from the *Job Outlook 2019* survey by the National Association of Colleges and Employers (NACE). A total of 172 organizations responded to the survey.

Attributes Employers Seek on a Candidate's Resume

	Attribute	**% of respondents**
1.	Communication skills (written)	82.0%
2.	Problem-solving skills	80.9%
3.	Ability to work in a team	78.7%
4.	Initiative	74.2%
5.	Analytical/quantitative skills	71.9%
6.	Strong work ethic	70.8%
7.	Communication skills (verbal)	67.4%
8.	Leadership	67.4%
9.	Detail-oriented	59.6%
10.	Technical skills	59.6%

Source: https://www.naceweb.org/talent-acquisition/candidate-selection/

Questions (10 min) **Start time: __________**

1. Consider the top three attributes. Why do you think they are the most sought-after?

2. How is communication (written and verbal) related to problem solving and teamwork?

3. Which of these skills do each of you individually need to work on? Justify your answers.

4. How will you develop these skills in college: in courses, or other activities? Be specific.

Meta Activity: Group vs Team

Throughout the course, you will need to examine and process information, ask and answer questions, construct your own understanding, and develop new problem-solving skills.

Questions (10 min) **Start time: __________**

1. What are some advantages to working in groups/teams?

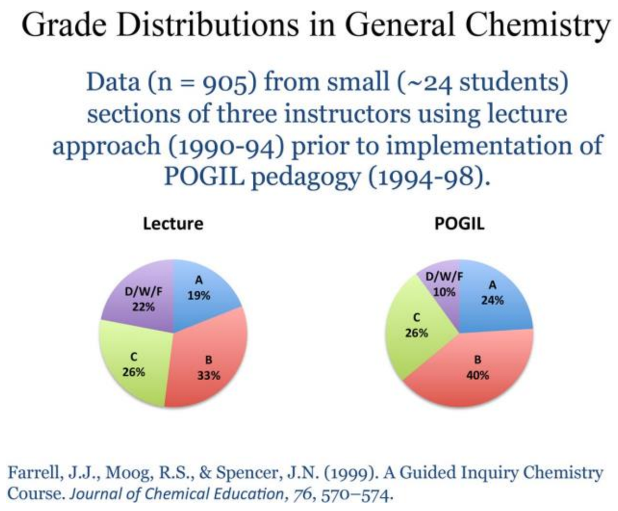
2. What are some disadvantages to working in groups/teams?

3. What is the difference between a group and a team? Come up with a precise answer.

4. How can working as a team help you accomplish the tasks described above? Give at least two specific examples.

Meta Activity: POGIL Research

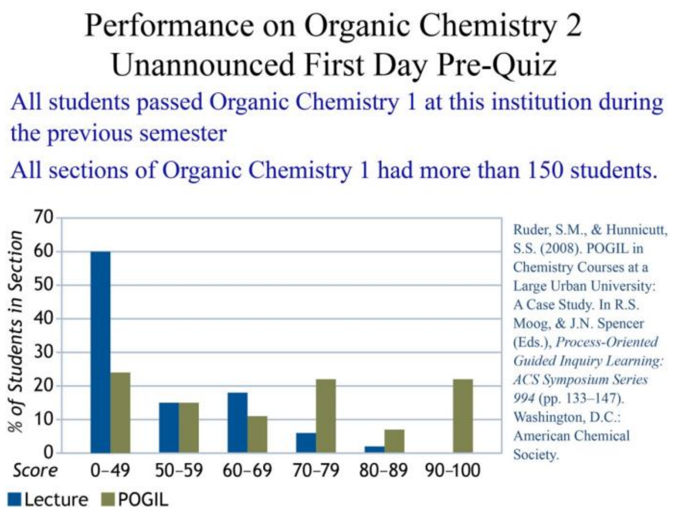
Process-Oriented Guided Inquiry Learning (see pogil.org) is a student-centered, group-learning instructional strategy and philosophy developed through research on how students learn best. The following two figures are from peer-reviewed articles published in education journals.



Grade Distributions in General Chemistry

Data (n = 905) from small (~24 students) sections of three instructors using lecture approach (1990-94) prior to implementation of POGIL pedagogy (1994-98).

Farrell, J.J., Moog, R.S., & Spencer, J.N. (1999). A Guided Inquiry Chemistry Course. *Journal of Chemical Education*, *76*, 570–574.



Performance on Organic Chemistry 2 Unannounced First Day Pre-Quiz

All students passed Organic Chemistry 1 at this institution during the previous semester

All sections of Organic Chemistry 1 had more than 150 students.

Ruder, S.M., & Hunnicutt, S.S. (2008). POGIL in Chemistry Courses at a Large Urban University: A Case Study. In R.S. Moog, & J.N. Spencer (Eds.), *Process-Oriented Guided Inquiry Learning: ACS Symposium Series 994* (pp. 133–147). Washington, D.C.: American Chemical Society.

Questions (10 min) **Start time: _________**

1. How large were the classes at each of the universities shown above?

2. What are the measures of performance shown in each of the figures?

3. What does the figure on the left suggest about POGIL's impact on student success?

4. What does the figure on the right suggest about students' retention of knowledge?

Case Study: Panic Attack

Frank was behind in his programming assignment. He approached Martin to see if he could get some help. But he was so far behind and so confused that Martin just gave him his code with the intent that he would "just look at it to get some ideas."

In the paraphrased words of Frank: "I started the assignment three days after you put it up. But then other assignments came in and I started on them too. I felt like I was chasing rabbits and began to panic. It was already past the due date and I got really scared. That's when I went to Martin to see if he could help." Frank copied much of the code and turned it in as his own.

Questions (10 min) **Start time: _________**

1. Which, if any, of the students were at fault? Why?

2. Which specific Honor Code violations occurred?

3. What should Martin have done in this situation?

4. What options did Frank have besides cheating?

Case Study: A Friend Indeed

Jeffrey was having trouble with one of the last programming assignments. He didn't even know where to begin and it was already late. Another student, Stephen, lived in his hall and Jeff was pretty friendly with him. Jeff went to Stephen's room and told him that his computer was acting flaky. He asked if could he borrow Stephen's laptop to finish up the program? Stephen was on his way out to dinner and told him okay. When he got back Jeffrey was gone.

While Stephen was out, Jeffrey searched for and located the code for the assignment on Stephen's machine. Jeffrey copied it onto his flash drive and took it back to his room, where he modified the code a bit before submitting it for a grade.

Questions (10 min) Start time: _________

1. Which, if any, of the students were at fault? Why?

2. Which specific Honor Code violations occurred?

3. What should Stephen have done in this situation?

4. What options did Jeffrey have besides cheating?

Case Study: Oops!

Emily was working in the lab on her programming assignment. She finished the program, submitted it, and went on to do some other work. Shortly thereafter, she left the lab.

Another student, Kyle, was working nearby. He knew that Emily had successfully submitted the assignment, and he had not been able to get his to work properly. When Emily left, he noticed that she had not logged out of her computer. He moved to her workstation, found the work under her Documents directory, and copied it onto his flash drive. He then logged out, logged in as himself, and copied the code onto his Desktop where he modified the program a bit, then successfully submitted it.

Questions (10 min) **Start time: _________**

1. Which, if any, of the students were at fault? Why?

2. Which specific Honor Code violations occurred?

3. What should Emily have done in this situation?

4. What options did Kyle have besides cheating?

Case Study: Too Close for Comfort

Bill and Jeff were two freshman rooming together in the fall semester and were taking CS 139. Bill was an excellent student for whom the work in CS 139 seemed to come easily. Jeff seemed to struggle a bit more, but was able to do the work and turned most labs and programs in on time.

Both programs came in looking virtually the same. When confronted, Jeff claimed the work as his own and stated that he did not know how the code from the roommates was the same. Later Bill told the professor that while programming was easy for him, he had struggled with another class and then got behind with this program. He didn't think the professor would find out and so stole his roommate's code and turned that in.

Questions (10 min) **Start time: _________**

1. Which, if any, of the students were at fault? Why?

2. Which specific Honor Code violations occurred?

3. What should Jeff have done in this situation?

4. What options did Bill have besides cheating?

Case Study: Let's Make a Deal

It was mid-semester and the pressure was on, not only in CS but in other classes. Jamie and Pat were each working on the programming assignment in the lab, but neither was having much success. Jamie had started several days ago, but he was having trouble debugging his current work. Pat had just started that day and knew that she would be turning it in late. Regardless, Pat offered to help Jamie work out the problems with his code.

Together, and after a couple of hours of work, they got the program to work. Pat said, "Now that yours is working, can you give me the code so that I can also get credit for this assignment?" When Jamie objected, Pat said, "Hey, you wouldn't have gotten it finished if it weren't for my help, and now mine will be even later!" So Jamie turned over a copy of the code. Pat made some changes to a few sections, and then turned in the final program.

Questions (10 min) **Start time: _________**

1. Which, if any, of the students were at fault? Why?

2. Which specific Honor Code violations occurred?

3. What should Pat have done in this situation?

4. What should Jamie have done in this situation?

Case Study: A Friendly Assist

George was struggling with the programming assignment on the night it was due. He had gone home over the weekend before, thinking that it would be easy to do this assignment, but it turned out to be more difficult than he thought. After working on some parts of it and giving up in frustration, he turned to Shelley, a senior CS student who had taken the course several semesters before. He showed Shelley the assignment, and the two of them worked on it late into the night. They successfully submitted the program with a late penalty of only one day.

Questions (10 min) **Start time: _________**

1. Which, if any, of the students were at fault? Why?

2. Which specific Honor Code violations occurred?

3. What should Shelley have done in this situation?

4. What options did George have besides cheating?

Case Study: A Team Effort

Stan, Jane, and Tom were part of a project team. At the start of the semester they all agreed to be responsible for everything that was submitted by their project. As the project got underway, they divided up the work and decided to complete and submit the three parts individually. Tom was behind on his work as the deadline approached, and so as not to be late he copied several pages from files of a roommate who had completed the same work last year. The team project was submitted on time, though only 2/3 of it was original work.

Questions (10 min) **Start time: _________**

1. Which, if any, of the students were at fault? Why?

2. Which specific Honor Code violations occurred?

3. What options did Tom have besides cheating?

4. Should all three students be penalized? Why or why not?