Week 9 - Strings and Modules

Model 1 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:

1. Does the `lower` method change the contents of the `dna` string? Justify your answer.

2. Describe the `list` function—what does `list(dna)` return in Model 1?

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

4. Name several other string methods not shown in Model 1. (Read the documentation.)

5. 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).

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

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

Model 2 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:

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

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

10. Identify the syntax of the statement to import:

a) a module

b) a function

11. Identify the syntax of a function call assuming:

a) the module was imported

b) the function was imported

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

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

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

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

16. 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?)

17. 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 3 Multiple Modules

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

```
1  import random
2
3  def angle():
4      number = random.randint(-90, 90)
5      return number
6
7  print("in move: __name__ ==", __name__)
8  print("will always execute: angle ==", angle())
9
10 if __name__ == "__main__":
11     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:

```
1  import move
2
3  print("in stop: __name__ ==", __name__)
4  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:

18. 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?

19. 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?

20. What was the reason to include the `import move` statement in `stop.py`?

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

22. What line in `move.py` did not print when `stop.py` was executed? Why?

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

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