Q and A for Sections 2.3 – 2.5

CS 106
Professor Victor T. Norman, DDS
Sequence

Properties of a sequence:

• A sequence is an **ordered collection** of multiple sub-items.
  – there is a first item, a last item, and all other items have a previous item and a next item.

• Can be iterated over. (Preview of coming attractions.)

• list, str, tuple.
Immutability

• Immutable → code cannot change value after initial creation.
• Only mutable type we know is a list (so far).
• str, int, float, bool, tuple: immutable.
• Why should you care?
  – Right now, not much reason...
  – Stops you from trying to change a part of a str.
  – There are no mutator methods for str.
Implications of Immutability

• \( a = 3 \)
  \( a = 4 \)
  \( \rightarrow \) We are not changing 3 to 4 in memory.

• \( \text{age} = 50 \)
  \( \text{herAge} = \text{age} \)
  \( \text{age} = 51 \)
  \( \# \) What is the value of \( \text{herAge} \) now?

• \( s = "Hello world" \)
  \( s[0] = "J" \)
  \( \rightarrow \) Syntax error: strings are immutable.

• To change H to J, you have to create a new string from the old one.
Strings (str)

- a sequence of characters
- immutable
- use pair of single-quote, double-quote, triple single-quote, or triple double-quote to create a literal. (Why so many ways?)
- How is a string different than a list of characters?
  - immutable
  - many useful str methods are defined.
""" and """

Why have """" and """" available to make string literals in python?

• If you need to have a single quote and a double-quote in your string, you need to start/end it with a triple quote.
  
  q = """"'He said, "'Twas the night before Xmas"'"""

• If you need newlines in your str, you need to use triple quotes:

  menu = """"1. Pancakes
          2. Waffles
          3. French Toast"""

• q = '''He said, "'Twas the night before Xmas"''''
Q: What is a *mutable object*?

A: It is an object that can be altered after it has been created.
Q2

Q: A string (type `str`) is an immutable object. Thus, any operation that looks like a mutator (like `capitalize()`) must return a new object.

A: new object

(In fact, mutators aren’t even defined for `str`.)
Q3

Q: Write code to create a new empty string and store it in a variable s.

A: s = str()
   or
   s = ""
   or
   s = ""
Q: Write code to make an identifier im_sorry refer to "I’m sorry". Then, write code to put make a variable not_really refer to You are “forgiven”.

A: im_sorry = "I’m sorry"
    not_really = ‘You are “forgiven”.’
Escape character

• \ (NOT /)
• In a string, changes the next character from normal to special, or vice versa. Can be used to indicate a tab, newline, linefeed, etc. (Used to be used to ring the computer’s “bell”, etc.)
• \n: newline
• \t: tab
• \\
: backslash
• ‘: single quote, in a string created with single quotes
Q5

Q: What is the escape character in python?

A: backslash

Q: What is it for?

A: it changes a special character into a normal one, and vice versa.
How to print this out?

I want to print out these instructions to the user:

Please set your workspace to C:\tmp\CS104\n
Write the code to do that.

print “Please set your workspace to C:\\tmp\\CS104\\”
Why use a tuple?

Tuple:
- sequence (has an ordering) (like a list)
- can hold items of any type (like a list)
- is immutable (not like a list)
- create with () and ,s. (use [] and ,s for a list)

Q: When would you ever use a tuple?
A: When you need an immutable object... 😊
You can only use an immutable object in some places – like as an “index” into a dictionary.

Honestly, tuples aren’t the useful... Sometimes a tuple just feels natural. Like when you specify a (x, y) location on the screen.
integer vs. true division

• integer / integer $\rightarrow$ integer result (by truncating the fractional part)
• float / integer or integer / float or float / float $\rightarrow$ true division.

(This changes in python 3. Use // to do integer division. Otherwise, it is true division. Do NOT use // in this class – we are using python 2.)
When would you use integer division?

• Suppose you want to find the middle item in a list (sound familiar? :-)
• aList = [ .......... ]
• print aList[ len(aList) / 2]
• This is integer division, which is perfect, since we need an integer to index into aList. And, if there are an odd # of items (say, 7), we’ll get the middle item (the 3-th item).
% operator

Q: Can you explain %?
A: % is the “modulo” operator. It returns the remainder from division. So...

I read “x % y” as “the remainder when dividing x by y”.
When would you use %?

If I say we should meet in 15 days, how many weeks and days is that?

Write code to compute that: I’ll give you totalDays and you compute the others based on it.

totalsDays = ______________
numWeeks = __________________________
numExtraDays = __________________________
Converting types

Q: What is the point of converting strings to integers and integers to floats? Couldn't you just as easily create a new str, int, or float?

A: Some built-in functions return a string, but you might need to work with an int or float. So, you need to convert it.

E.g., raw_input() returns a str.
Q9

Q: A slice is an operation on a sequence (a list, str, or tuple) that returns a _____________.

A: new sequence (list, str, or tuple)
Q10

Q: What is in subpoem when this is done?:
(assume no newline in 1st line)

poem = “I slit a sheet, a sheet I slit,
upon the slitted sheet I sit.”


A: subpoem refers to new string “sleet”.
Q11

Q: How about this?
  sub2 = poem[:14]
  sub3 = sub2[7:]

A: sub2 refers to “I slit a sheet”
  sub3 refers to “a sheet”
Q12

Q: What does this do?:
newpoem = poem[ : :-1]

A: It makes newpoem refer to a new string that has poem reversed.
**BIG Question**

Q: Why do we need to write code to manipulate strings? I don’t understand the point of all this string stuff.

A: Much of computing is about manipulation of strings. Look at facebook. How many numbers to you see displayed? There is lots of arithmetic computation happening behind the scenes, but ultimately it is about displaying a bunch of strings on the screen for you to waste time looking at. Or, consider email. Or Microsloth Word. Or any webpage, which is a string sent over the network and rendered in your browser.
String terminology

• concatenation: putting two strings together.
  – newStr = “hi” + “there” results in newStr referring to “hithere”

• s.strip(): make new string by removing whitespace from the beginning and ending of s.

• s.strip(chars): remove every character in chars from beginning/ending of s.

• s.split(): return new list of strings dividing s into pieces on whitespace.

• s[m:n:o]: return new string with characters from s starting at index m and going up to n, by o steps.
Lexicographic ordering

• “dictionary” ordering: compare first item. If same, compare next item...
• Done for comparing any pair of sequences.
• \([0] < [1]\): True or False?
  – True
• \([0, 0] < [0, 1]\)
  – True
• \([0, 1, 2, 3] < [0, 1, 4]\)
  – True
• \([0] < [0, 77]\)
  – True
• “hello world” < “hello world!”
• rarely used on lists/tuples. Sometimes used on strings.
join()

• Syntax:  `s.join(seqOfStrings) → produces a string.
• “ " .join([“Vic”, “and”, “Susan”]) creates string “Vic and Susan”.
• (Like the opposite of split()).
• “.” .join([“153”, “106”, “4”, “1”]) → “153.106.4.1” (www.calvin.edu IP address)
Extra slides
% is not always modulo

Q: When is the % operator not considered a modulo operator? Is there a case where it would be called something other than an modulo?

A: % is only remainder with numbers. It is the “format” operator when used with a string on the left side. This is polymorphism: the types of the operands determine the meaning of the operation.
Literals

Q. Can you go over what a literal is?

A. A literal is a value, written in the program. Because Python needs to know the type of everything it is dealing with, you have to tell the interpreter what type of thing every literal is. Thus, you have to give quotes around strings, a decimal in a number to make it a float, etc.
Float $\rightarrow$ int

Q: What happens when you convert a float to an int?
A: You lose everything after the decimal point – i.e., you truncate the number or “floor” it.
Floats

Q: I am not sure I understand the idea of a float — isn’t it still rounding off the number?

A: A computer represents a floating point number as best it can — as precisely as it can. So, yes, most times it does have to round it off (after 10 or 12 or 15 decimal places.)
Int vs. Long

Q: What is the purpose of the long class? When does an integer switch to a long?

A: A long is stored in more memory than an int. (64 bits vs 32 bits, typically). Python notices when an int would “overflow” and “promotes” the type to long instead.
Conversions between all types?

Q: Did I understand properly that there is a type conversion function between all types?
A: There are conversions defined between most “simple” built-in types. E.g., bool <-> int, bool <-> float, single-character string <-> int, etc. There are not conversions for multi-character strings to ints/float/etc, nor for tuples/lists to ints, etc.
tuple/list conversion

Q: Can you convert a list to a tuple and vice versa?
A: Yes!

tuple([“this”, “is”, “a”, “list”]) makes (“this”, “is”, “a”, “list”)
list( (3, 4) ) \rightarrow [3, 4]
list("Hello") weirdness

Q: I am confused by the list ('hello') syntax.
A: “Hello” is a str and strings are sequences. list() is a constructor of a new list, and can take a sequence as an argument and makes a new list with each element of the sequence as an item in the list.

(Whoa! Hold the phone! Is list() a constructor or a type conversion function?!)
Constructors vs. type conversion functions

Note: there are no type conversion functions in python. int(), float(), tuple(), list(), str(), etc., are all constructors for the types. But, they can take an argument and will make a new object out of the given argument, thus doing what seems like a type conversion.

ord() takes a single-character string and returns its corresponding integer in the ASCII character set – even it does not do type conversion.
isalpha, etc., examples

Q: Can we go through the examples of isalpha(), isdigit(), islower(), and isupper()?

A: These functions test the characters in a string and return a True or False value.

isalpha() → True if str not empty and str all alphabet characters.

Others are similar.
Q: How do you access the list that you have made by evoking the split method on a string? Like in the examples on page 54, how would you access the final list that you have made because you did not name it when using the split method?

A: If you don’t save the result, you don’t access it.
join()

• string method, so: 

  

  `.join(aSeqOfStr)`

• takes a list as the parameter and returns a str
• puts the str in between every string in the sequence.
Q6

Q: What is in variable a_str after this code is run?

a_str = """"Four score and seven years ago."""

A: a_str is a string that contains a newline in it, between score and and.
Q7

Q: Rewrite the previous code, but without using the triple quotes, but still containing a newline.

A:

    a_str = "Four score\nand seven years ago"
Q8

Q: What is the output from this?:
   >>> s = “Hi there.”
   >>> s.count(‘e’)

A: 2

Q: but, count() is a method for lists, no?
A: Yes, but a list and a string are both sequences, and share a lot of the same methods.
Q13

Q: What is “whitespace”?

A: spaces, tabs, or newlines in a string
Q: What does this do?
   parts = poem.split()
   print len(parts)

A: Line 1 splits poem by whitespace, producing a list of words. Line 2 prints how many words are in the list called parts.
Q15

Q: What does this do?
   parts = poem.split(',,')
   print len(parts)

A: It prints the number of phrases in the poem.
Q16

Q: And this?:
   >>> "ab cd ef gh".split()

A:
   ['ab', 'cd', 'ef', 'gh']
   >>>
Q: “[A] tuple is used to represent an immutable sequence of objects.” What does this mean?

A: It is something that holds a list of items, but the list cannot be altered -- i.e., there are no mutator operations available.
Q19

Q: If I have this:
   color = (20, 50, 70)
   representing the red, green, and blue components of a pixel, how do I change the green part in color?

A: I don’t.
Q: Is it necessary (in general) to know whether you are dealing with an int or a long?

A: No. As the authors say, “python does a nice job of automatically converting from the use of int to long when necessary.”
Q: What is the output?
  num = 23
  div = 4
  print num / div
  print num // div

A: both outputs are 5. The first divides 2 integers so integer division is used. The second uses integer division explicitly.
Q22

Q: How would you get floating point division to happen with num and div?

A: print float(num) / div
   or
   print num / float(div)
Q: Suppose you are writing code inside a loop where the loop variable \( i \) has a value from 0 to 10000. In this loop, you want to print a message every 100 iterations -- i.e., when \( i \) has value 0, 100, 200, 300, ... Describe how you would do this using the \% operator.

A: test if \( i \% 100 \) equals 0
Q24

Q: What does this do?
   is_even = (i % 2 == 0)

A: is_even is True if i is even, False otherwise.
Q25

Q: If len(some_list) returns the length of some_list, and round(some_float) returns some_float rounded off, what do these do?
int(some_float)
float(some_int)
str(some_int)
int(some_str)

A: int(p): constructors a new integer, float(p) constructs a new float, str(p) constructs a new string