Python Workshop Day 2

Coding Pals | UBC Edith Lando

Homework

i. CCC13J1

ii. CCC16J1

iii. CCC20J2

Review

if Statements

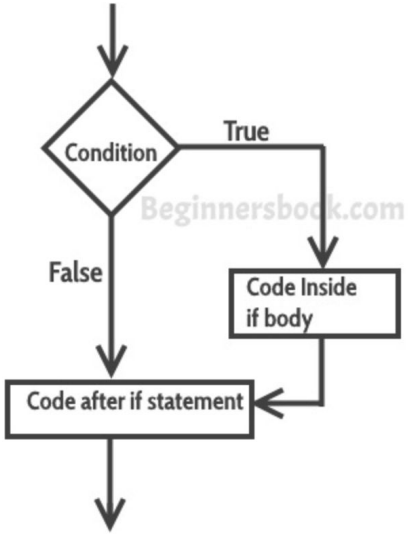
Example:

```
flag = True

if True:
    print("Welcome")
    print("To")
    print("Coding Pals")
```

Output:
Welcome
To
Coding Pals

If statement flow diagram



elif Statements

The `elif` keyword is pythons way of saying "if the previous conditions were not true, then try this condition".

```
a = 33

b = 33

if b > a:

  print("b is greater than a")

elif a == b:

  print("a and b are equal")
```

else Statements

The `else` keyword catches anything which isn't caught by the preceding conditions.

```
a = 200

b = 33

if b > a:

  print("b is greater than a")

elif a == b:

  print("a and b are equal")

else:

  print("a is greater than b")
```

How to use While Loops

In order to use While loops, you need to declare it in a way similar to If statements. The most basic method requires an integer, which is increased each time the loop repeats until it no longer matches the loop condition.

In the following code, the number 4 is printed out 4 times.

```
n = 1

while n < 5:

  print(4)

  n += 1
```

How to use For Loops

The most basic way to use “For loops” is to use it with the range() function. range() starts at 0 by default, increases by 1, and returns each number until an end number. Remember: Python numbering makes it so that the real end number is one less!

The following code prints all the numbers from 0 to 5.

```
for x in range(6):
  print(x)
for x in range(0,6):
  print(x)
```

What are Strings?

In Python, strings are *sequences of characters*. Characters, simply put, are just any symbols or numbers. Strings are a data type, just like integers.

Strings are especially useful in coding assignments, since you may be required to store important words and numbers and retrieve it for later use.

How do we use Strings?

In most cases, you will need to declare your strings at the very top of your code. To do this, you will need a name for your string. You will also need to indicate that it is a string with quotation marks.

The following code consists of a string assigned “Hello”, which is then printed out with the print function.

```
string = “Hello”

print(string)
```

What is Slicing?

One useful tool to use when it comes to strings is *slicing*. Slicing is when you only refer to smaller sections of a string. Slicing occurs in this format: s[start:end].

The end index is not included.

The following code consists the same string that has been sliced and printed to only show the letters “el”. So we start at index 1 and we stop at index 3.

```
string = “Hello”

print(string[1:3])
```

How do we get String Length?

In many cases, it may be important to get the length of the string. To do this, simply use the len() function.

The following code consists the same string, with the length printed out.

```
string = “Hello”

print(len(string))
```

Check String

To check if a certain phrase or character is present in a string, we can use the keywords in or not in

```
sentence = "The rain in Spain stays mainly in the plain"
check = "ain" in sentence
print(check)

x= "Andrew is a cool kid"
if ”cool kid” in x:
    print(“yes”)
else:
    print(“no”)
```

Combining Strings

To concatenate (technical term for combine) two strings you can use the + operator.

```
a = "Hello"
b = "World"
c = a + b
print(c)
```

Additional Methods Part 1

The lower() method will return the string with all lowercase letters.

```
string = "Hello"

print(string.lower())
```

hello

The upper() method will return the string with all uppercase letters.

```
string = "Hello"

print(string.upper())
```

HELLO

Additional Methods Part 2

The strip() method removes any whitespace from the beginning or the end:

```
a = " Hello, World! "

print(a.strip()) # returns
"Hello, World!"
```

The replace() method replaces a string with another string:

```
a = "Hello, World!"

print(a.replace("H", "J"))
```

Jello, World

Example Question: CCC'2013 Junior 2

- We will do a step to step tutorial right now
- Link to problem

Break!

What is a list?

A way to store multiple values in a variable

A way to group similar information together under the same name

Like a to-do list

What do lists look like in python?

Sticking with the to-do list example:

```
todo = ["log on to computer", "join zoom call", "learn about python lists"]
```

This creates a list called “todo” that stores 3 strings

You can also separate the elements on new lines because python ignores whitespace inside lists.

```
todo = [
    "log on to computer",
    "join zoom call",
    "learn about python lists"
]
```

What do lists look like in python?

The elements in python lists don’t have to be the same data type

For example

```
l = [1, 1.5, ”1.5”]
```

is a valid python list

How do I access elements in a list

```
todo = ["log on to computer", "join zoom call", "learn about python lists"]
```

Earlier we had this list, how do we access each item?

The easiest way is to tell python which place the element comes in the list

E.g. I want the element in the first/second/third place

Keeping in mind that python counts from 0, so the first element is the 0th index.

```
>>> print(todo[0])
log on to computer
>>> print(todo[1])
join zoom call
```

How do I access elements in a list

You can also access multiple elements through the use of list slices

```
>>> l = [1, 2, 3, 4]
>>> print(l[1:3])
[2, 3]
>>> print(l[:2])
[1, 2]
>>> print(l[2:])
[3, 4]
>>> print(l[::2])
[1, 3]
```

Leaving the start or the end blank defaults to the first and last elements of the list

Adding elements

To add an element to a list use the append method

```
>>> l = [1, 2, 3]
>>> l.append(4)
>>> print(l)
[1, 2, 3, 4]
```

To add a list to the end of a list use the extend method

```
>>> l = [1, 2, 3]
>>> l.extend([4, 5, 6])
>>> print(l)
[1, 2, 3, 4, 5, 6]
```

Removing elements

To remove an element by its index (e.g. to remove the first, second, etc. item) use the del keyword

```
>>> l = [1, 2, 3]
>>> del l[1]
>>> print(l)
[1, 3]

>>> l = [“a”, “b”, “c”, “d”]
>>> del l[1:3]
>>> print(l)
[“a”, “d”]
```

Removing elements

To remove an element by its value (e.g. remove the string “hello” from a list) use the remove method

```
>>> l = [1, 2, 3]
>>> l.remove(2)
>>> print(l)
[1, 3]
```

Finding the length of a list

You can use the len function to find the length of a list

```
>>> l = [1, 2, 3]
>>> print(len(l))
3
```

Change List Value

To change the value of a specific item, refer to the index number

```
thislist = ["apple", "banana", "cherry"]

thislist[1] = "blackcurrant"

print(thislist)
```

```
["apple", "blackcurrant", "cherry"]
```

Check if item exists

To determine if a specified item is present in a list use the in keyword

```
thislist = ["apple", "banana", "cherry"]

if "apple" in thislist:

  print("Yes, 'apple' is in the fruits list")
```

Sorting a list

Python provides a sort method for lists

```
>>> l = [2, 1, 3]
>>> l.sort()
>>> print(l)
[1, 2, 3]
```

To avoid modifying the original list you can use the sorted function

```
>>> l = [2, 1, 3]
>>> print(sorted(l))
[1, 2, 3]
>>> print(l)
[2, 1, 3]
```

Iterating through a list: index referencing

With the for loop we can set the range to be the size of the list or string and reference the index when we print each letter. This works because the iterator “i” will increase from 0 to the size of the list with every iteration of the loop.

Example 1:

```
fruits = ["apple", "banana", "cherry"]
for i in range(len(fruits)):
  print(fruits[i])
```

Example 2:

```
name = “Prad”
for i in range(len(name)):
  print(name[i])
```

Iterating through a list: for each loop

Using the for-each loop, we change the iterator i to reference each element in the list or string directly

Example 1:

```
fruits = ["apple", "banana", "cherry"]
for i in fruits:
  print(i)
```

Example 2:

```
name = “Prad”
for i in name:
  print(i)
```

Notice how you do not need to use fruits[i] or name[i] and can instead print “i” directly

List Methods

The `pop()` method removes the specified index, (or the last item if index is not specified)

```
thislist = ["apple", "banana", "cherry"]
thislist.pop()
print(thislist)
```

The `clear()` method empties the list

```
thislist = ["apple", "banana", "cherry"]
thislist.clear()
print(thislist)
```

List Methods

The `reverse()` method reverses the sorting order of the elements

```
fruits = ['apple', 'banana',
'cherry']

fruits.reverse()
```

The `index()` method returns the position at the first occurrence of the specified value.

```
fruits = ['apple', 'banana',
'cherry']

x = fruits.index("cherry")
```

What are functions?

A function is a set of statements that might take inputs, does some specific computation and produces output.

What do functions look like?

Functions Structure:
def <function name> (parameter list):
    <statements>
A function is defined using the keyword def

Function name is used when calling the function

The parameter list is used for information that needs to be passed into the function(ie. list, strings, integers)

The statements are the operations that occur

Why use functions?

You can use a function anytime you need to do a set of operations multiple times. Functions help reduce duplication in your code and breaks the code down into simpler pieces.

Number of Parameters

By default, a function must be called with the correct number of parameters. Meaning that if your function expects 2 parameters, you have to call the function with 2 parameters, not more, and not less.

This function expects 2 arguments, and gets 2 arguments:

```
def my_function(fname, lname):
  print(fname + " " + lname)

my_function("Kevin", "Guo")
```

Function return and calling a function

1. If the function has: return variable/ number
    a. print(myfunction())

2. If the function has: return True/False
    a. print(myfunction())

3. If the function contains: print(variable or number)
    a. myfunction()

Multiple Parameters Example

```
# A simple Python function to determine the larger of two numbers
def size( x, y):
    if x>y
        print("x is larger")
    elif x==y:
          print("x and y are the same")
    else:
        print("y is larger")
size(4,7)
size(5,5)
```

Output:
y is larger
x and y are the same

Break!

Random Module in Python

The Random module allows us to generate random values on the computer and use them in our code

For our Hangman game, we will have a word bank and pick a word at random from the bank. Therefore, we need to use the random module to pick a random word

We will use the randint() function, which has two parameters and generates a number between those two parameters (inclusive)

```
Ex: print(random.randint(0,10)) #prints a random integer between 0 and 10
```

Hangman
Follow along in your own repl!

Hangman Game

Hangman game reference: https://replit.com/@davisc3126/HangmanPython

The End!
Feel free to reach out with any questions!