Business Analysis Summer 2022

Class 2

Sri Akhil Jonnalagadda
sjonnala@stedewards.edu

Python Code Blocks

- Critical concept: in python, code runs in “blocks” which are segments of code that share an indentation. If you indent a code block, it has a meaning.. It means it is a code block.

Example 1:

```
print(“hi”)

print(“sir or madam”)
```

Example 2:

```
age = 20

if (age == 20):

    print “almost”
```

Notice the indent!

Basic Loops

Loop	Description
for	This is traditionally used when programmers had a piece of code and wanted to repeat that 'n' number of times.
while	The loop gets repeated until the specific Boolean condition is met.
Nested	Programmers can use one loop inside another; i.e., they can use for loop inside while or vice - versa or for loop inside for loop or while inside while.

For loop

```
for iterating_var in sequence:

    #execute your code


for x in range (0,3) :

   print ('Loop execution %d' % (x))


cars = [“Cobra”,”Mustang”,”Corvette”]

for x in cars:

  print(x)
```

While + Nested loops

```
while count <= 100:
    print (count)
    count = count + 1
#nested:
cars = [“Cobra”,”Mustang”,”Corvette”]
for x in cars:
    while count <= 100:
      print(x)
      count = count + 1
```

Control Statements [break out of loop or skip to top]

Control Statements	Description
Break statement	It is used to exit a while loop or a for a loop. It terminates the looping & transfers execution to the statement next to the loop.
Continue statement	It causes the looping to skip the rest part of its body & start re-testing its condition.
Pass statement	It is used in Python to when a statement is required syntactically, and the programmer does not want to execute any code block or command.

Quick Review Game…
Which side is correct?
Side A or Side B

```
fruit = “apple”
if (fruit==“apple”):
    print(“it fell”)
    some_number = 10.1
    print((“it is:”+ some_number))
    while (some_number > 1)
        print(“ it is > 1”)
        some_number = some_number -1
```

```
Fruit = “apple”
if (fruit=“apple”):
    print(“it fell”)
    someNumber = 10.1
    print(“it is: %f” % someNumber)
    while (someNumber > 1):
        print(someNumber + “ is > 1”)
        someNumber = someNumber -1
```

In-Class Program
Part 1/2

- Write a program that:
  - Runs in an infinite loop
    - *asks you your name*
    - *asks you your age*
    - *asks you your approximate current GPA*
    - *asks you how much is a cup of coffee*

Quick In-Class Program
Part 2/2

- Add to your program:
  - After it asks you the questions, it should tell you a little story using the three data points..
    - *Make it a little funny.*
    - *For example: (this is not that funny, you can be more creative)*
      - NAME went for a walk in the woods. He found COFFEEPRICE laying on the ground. Instead of picking it up, NAME decided to walk past it GPA times… finally, after the GPA time, NAME picked it up and spent it on candy.
  - After it tells you a story,
  - Have the program ask if you'd like to hear another story (Y or N)?
  - If user presses yes, take them back to the top of the loop and start over.
  - If no, then say "Thanks for Playing" and exit.

Functions in Python

A function is a block of code that you might want to do several times…

It can also take an input parameter (or even parameters) so that it can be called with different data!

Why?

1. It reduces typing.
2. It makes your code more readable.
3. It makes your code more re-usable.
4. To call a function in a program, just use its name followed by ()
    1. Any input parameters go inside the ()
5. Almost all of Python uses functions… you already use them (for example input() is a function, and it takes an optional parameter called prompt_string.

```
name = input("Whats your name? ")
```

Making functions

- You should make a function for your code if:
  - You will call it several times
  - And/or you want to make your code more readable/understandable
- Making a function is easy, use def function_name(argument_variable):
  - Then use indentation to block out the function code.
  - return command is used to exit the function and give back optional data

```
def sum3(a, b, c):
  return a + b + c


def avrg(first, *rests):
   return (first + sum(rests)) / (1 + len(rests))
# Sample use, Putting values
print (avrg(1, 2))
print (avrg(1,2,3,4))
```

To accept any number of keyword arguments, the arguments have to start with *.
The * argument can appear only in the last position of function's argument.

Let's Code!
Functions and Loops

- Write a program that will ask:
  - How many datapoints do you have?
    - *Then it should ask for each data point: "Input data item #: " (where # is the number of the data point)*
  - Then it should do 2 calculations (use a function for each of these):
    - *Function 1: sum_all : Adds up all the data, returns sum*
    - *Function 2: max_value: Finds maximum value, returns max value*
  - Your program should then print:
    - *"sum is ___"*
    - *"max is ___"*

Lists (review)

- alist = [1,2,3]
- alist.append(4)
- len(alist)
- alist[0]
- alist[0]=3
- a_slice = alist[0:2]
- string_list = [“hi”, “you”]