Recitation03

September 9, 2020

0.1 Loops

Loops allow us to "copy and paste" lines of code with a similar structure, replacing certain aspects of those lines with values from a list, tuple, or dictionary. The two most common types of lists are "For loops" and "While loops"

0.1.1 For loops

You can use a For loop to run through values in a list:

[173]:
```
for x in [1,2,10,'Fiddle']:
    print(x)
```

```
1
2
10
Fiddle
```

The For loop above ran through values in the list `[1,2,10,'Fiddle']`, assigned each value in order to the variable `x`, and then executed the code "inside" the For Loop. In this case the only code to execute was `print(x)`.

Here is a second example where we do a bit more with each item in the list.

[174]:
```
for x in [10,4,3,23]:
    y = x-10
    print(y)
```

```
0
-6
-7
13
```

We see that each item in the list `[10,4,3,23]` is assigned the variable `x`. Then we create a variable `y` that is `x-10` and print the value of `y`.

We can also loop through tuples

[175]:
```
for foodAmountsInGrams in (100,23,343,10):
    foodAmountsInLbs = foodAmountsInGrams*0.00220462
    print(foodAmountsInLbs)
```

```
0.220462
0.05070626
0.75618466
0.022046200000000002
```

Looping through dictionaries is a bit different. To use a For loop with a dictionary, you need to provide two variables in the loop: one variable for the key and a second variable for the value. You also need to add `.items` to the end of the dictionary.

[176]:
```
for (birdNumber,birdCoordsAndFeed) in birdData.items():
    print(birdNumber)
    print(birdCoordsAndFeed[0])
```

```
1
(2.3, 4.5, 0.9)
2
(-2.3, 44.5, -0.9)
3
(4.3, 14.5, 0.99)
```

But what is `birdData.items()` doing? When you append the code `.items()` to any dictionary, a list is created that contains two pieces of information: a key and the corresponding value.

[177]:
```
print(birdData.items()) # notice how the below code encloses everything in [], a␣
↪list!
```

```
dict_items([(1, [(2.3, 4.5, 0.9), [430, 222, 450, 233]]), (2, [(-2.3, 44.5,
-0.9), [430, 100]]), (3, [(4.3, 14.5, 0.99), [130, 22, 330]])])
```

This is not the only way to loop through a dictionary. You can also loop through just the keys

[178]:
```
for birdNumber in birdData:
    print(birdNumber)
```

```
1
2
3
```

As a recap of For loops, before we move to While loops, a For loop has the following structure.

```
for <variable> in <Python Object>:
    Multiple lines of code
    to run
    for each value in <Python Object>
```

0.1.2 While Loops

While loops have a similar structure to For loops. The key difference is that a While loop is finished after a specific condition is met whereas a For loop is finished after it runs through the last value in your list, tuple, or dictionary.

Let's look at an example of a While Loop.

[179]:
```
x = 0
y = 0
while x < 4:
    y = y+10 # create a variable y and assign to it the value of y in memory␣
 ↪plus an additioanl "10"
    x = x+1  # create a variable x and assign to it the value of x in memory␣
 ↪plus an additional "1"
print(y)
```

```
40
```

A while loop has the following structure

```
while <condition>:
    Multiple lines of code
    to run
    for each value in <Python Object>
```

and once the `<condition>` is met the loop finishes.

0.1.3 If else

One very common conditional structure in any programming language, including Python, is the if-else statement. The if-else statement executes lines of code dependent on whether a list of conditions are met or not.

The structure looks like this

```
    if <some condition>:
        indented lines of code
        to
        run if the condition is True
    elif <some other condition>:
        indented lines of code
        to
        run if the first condition was False and the above condition was True
    elif <some other condition>:
        indented lines of code
        to
        run if the above two condition were False and this one is True
    else:
        indented lines
        of code
        to run if none
        of the above statements were true
```

If-else statements are a way to introduce structure and logic into your program. For example, if I wanted to pick out birds whose first meal was greater than 400 grams I could write

[180]:
```
birdNumberWhoAteMoreThan400Grams = []                    # This is an EMPTY list
for (birdNum, coordinatesAndMeals) in birdData.items():  # look at all the code␣
↪we're writing already!
    meals = coordinatesAndMeals[1]                       # extract the second␣
↪entry in our list, the meals.
    if meals[0] > 400:                                   # is the first meal␣
↪they ate more than 400?
        birdNumberWhoAteMoreThan400Grams.append(birdNum) # if yes then append␣
↪their number to a list
print(birdNumberWhoAteMoreThan400Grams)
```

```
[1, 2]
```

0.1.4 Functions

Functions are pieces of code that take inputs and return outputs, much like a mathematical function. In Python functions are created with the keyword `def` and include the keyword `return`. Functions are essential components of any program in any language and allow you to run the same piece of code for different inputs.

The structure of a function is

```
def <name of your function>(<inputs separated by commas>):
    lines of code
    that
    are indented
    return <the Python object, or information, you want to return>
```

An example of a function could be to count the number of observations that fall inbetween two numbers. The input would be the observations, and the two numbers to fall between, and we would return the number of observations.

[181]:
```
def countObsBtw2Numbers(observations, firstNum,seconNum):
    count = 0
    for obs in observations:
        if firstNum <= obs < seconNum: # an example of a condition to be met.
            count = count+1
    return count
```

We can then generate some observations and apply our function to them.

[182]:
```
import numpy as np
OneHundredRandomObservations = np.random.normal(0,1,100)  # dont worry, this is␣
↪the next part of recitation.

obsBtwNeg1AndPos1 = countObsBtw2Numbers( OneHundredRandomObservations, -1, 1)
print(obsBtwNeg1AndPos1)
```

```
73
```

In the code above, we first generated 100 random observations. Next we applied our function `countObsBtw2Numbers` which took three inputs: * The observations we want to count * The first number observations need to be greater than or equal to * The second number observations need to be strictly less than

Our function returns an integer called `count` and in our code we assigned count to a variable called `obsBtwNeg1AndPos1`.

0.2 Modules

In the code there was a line we didn't talk about yet. It looked like this `import numpy as np`. This is an example of a module—a computer program that is a (usually large) list of functions.

Modules can be downloaded from the web and installed in Python, and many modules we'll use are pre-installed. To access a module you use the `import` keyword. The structure for importing a module looks like this

```
import <name of module> as <an optional way to refer to the module>
```

In the code above we imported a module named **NumPy** an enormous list of functions for numerical computing in Python. Inside Numpy (this is the same for other modules) are several different programs that contains lists of functions. You can access a list of functions by placing a dot between the module and list of functions you want to access.

In our example we wanted to choose a subset of functions in the `random` list under the NumPy module so we wrote `np.random.<the function inside random which is inside Numpy>`.

0.3 One attempt at taking a SRS of a population.

Let's write some code to draw a simple random sample from a population like we discussed in class last week. A simple random sample chooses a subset of observations from a population with equal probability.

In the below example I'll generate a population and ask that you write code to draw a simple random sample of 100 observations. Remember you can ask me questions! Ask me during office hours, during non-office hours, over zoom or by email. I'm here to help you learn.

[183]:
```
population = list( np.random.normal(25,2,10**3) ) # a list of 1,000 observations␣
↪that we will consider our population.

# Write your code for a Simple Random Sample below this line.
# ------------------------------------------------------------
```