Week 3 - Loops and Functions

February 17, 2022

1 CIS7026 - Business Process and Data Analysis

2 Week 3 Python loop statements and functions

2.0.1 In the week 3 workshop, you will learn/revise the basic concepts of programming with Python such as loops and functions statements. There are empty cells where you are required to do small tasks. Please, write below # line in those empty cells as # is used to comment code in Python.

3 Note

This course does not require intensive programming in Python. Therefore, we will not go deep in the programming but rather do a crash course during workshop sessions. So that you may get familiar with the basic concepts of Python programming with hands-on practice.

You can always learn more in your own time with the Online resources provided in the Python Programing section at Moodle.

4 for Loops

A for loop acts as an iterator in Python; it goes through items that are in a *sequence* or any other iterable item. Objects that we've learned about that we can iterate over include strings, lists, tuples, and even built-in iterables for dictionaries, such as keys or values.

We've already seen the for statement a little bit in past lectures but now let's formalize our understanding.

Here's the general format for a for loop in Python:

```
for item in object:
    statements to do stuff
```

4.1 Example 1

Create a list of integers

[1]:
```
# create a list of integers from 1-10
# create empty list
list1 = []
```

```
for x in range(1,11): # range is a special function in python to generate a␣
↪range of numbers, here 1 is inclusive and 11 is exclusive
    list1.append(x)
```

[2]:
```
#check list
list1
```

[2]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

4.2 Example 2

Iterating through a list

[3]:
```
# printing list elements 1 by 1
for x in list1:
    print(x)
```

```
1
2
3
4
5
6
7
8
9
10
```

4.3 Example 3

Finding odd number in the list

[4]:
```
for x in list1:
    if x%2!=0:
        print(x)
```

```
1
3
5
7
9
```

4.4 Example 4

find sum of elements in list

[5]:
```
total_sum = 0

for x in list1:
    total_sum = total_sum + x
```

```
[6]: print(total_sum)
```

```
55
```

4.5 Task for you

Try to find factorial of 10 using 'for' loop. (Hint it works similar to Example 4)

```
[11]: count=1
      for i in range(1, 11):
          count=count*i
      print("factorial of 10 =", count)
```

```
factorial of 10 = 3628800
```

5 while Loops

The while statement in Python is one of most general ways to perform iteration. A while statement will repeatedly execute a single statement or group of statements as long as the condition is true. The reason it is called a 'loop' is because the code statements are looped through over and over again until the condition is no longer met.

The general format of a while loop is:

```
while test:
    code statements
else:
    final code statements
```

5.1 Example 1

```
[12]: x = 0

      while x<=5:
          print(x)
          #increment x in each iteration
          x=x+1
```

```
0
1
2
3
4
5
```

while loop keeps going until condition is True (here x<=5). Once the condition is False, loop is terminated

5.2 Example 2

Let's use else with while loop. Finding factorial of 5

[13]:
```python
x = 1
factorial = 1
while x <=5:
    factorial = factorial * x
    print ("Finding factorial")
    #don't forget to update the condition, otherwise it will last forever
    x = x+1
else:
    print("Factorial of 5 is {}".format(factorial))
```

```
Finding factorial
Finding factorial
Finding factorial
Finding factorial
Finding factorial
Factorial of 5 is 120
```

5.3 Functions

A function is a useful to set of statements so they can be run more than once. We can specify parameters that can serve as inputs to the functions.

Functions allow us to not have to repeatedly write the same code again and again. For example print() is a function and the input that we provide in it, will be printed

Functions will be one of most basic levels of reusing code in Python

5.4 Using def keyword

How to build a function

[27]:
```python
def function_name(arg1,arg2):
    '''
    This is where the function's Document String (docstring) goes.
    When you call help() on your function it will be printed out.
    '''
    if arg1<50 or arg2<50:
        print("Unfortunately, you are fail")
    else:
        percentage=((arg1+arg2)/200)*100
        print("Your overall percentage is : {}".format(round(percentage,2)))

x=int(input("Enter your marks in Business and data analysis module: "))
y=int(input("Enter your marks in Technology project management: "))
function_name(x, y)
```

```
Enter your marks in Business and data analysis module: 35
Enter your marks in Technology project management: 40
Unfortunately, you are fail
```

Example 1 simple function that prints hello world

[14]:
```python
def hello_world():
    print("Hello World!")
```

[15]:
```python
# call this functions
hello_world()
```

```
Hello World!
```

In the example above we didn't define input variables. #### Accepting parameters (arguments) let's extend this example to create another function that gets user's age as input and print it #### Example 2

[28]:
```python
def print_age(age):
    print("You are {}".format(age))
```

[29]:
```python
#check
# get age from user
age = input("Please, enter your age ")
print_age(age)
```

```
Please, enter your age 26
You are 26
```

Using return So far we've only seen print() used, but if we actually want to save the resulting variable we need to use the **return** keyword.

a function to *return* a result that can then be stored as a variable, or used in whatever manner a user wants.

Example 3: Addition function create a function that gets two input arguments/variables and return sum

[30]:
```python
def add(a,b):
    return a+b
```

[31]:
```python
#check - we are storing sum in another variable here
addition = add(2,3)
addition
```

[31]: 5

Example 4: using if else and loops with functions Write a function which returns odd numbers from the list else empty list

```
[32]: def check_odd_numbers(list_num):
          odd_num = []

          # iterate list by for loop
          for x in list_num:
              if x%2!=0:
                  odd_num.append(x)
          return odd_num # notice indentation here
```

```
[33]: # let check now
      # create a list
      # if you want to use list comprehension, it is advance level
      # a = [x for x in range(1,21)]
      a = []
      for x in range(1,21):
          a.append(x)

      a
```

```
[33]: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
```

```
[36]: # List comprehension
      a = [x for x in range(1,26)]
      print(a)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22,
23, 24, 25]
```

```
[37]: # check our function
      check_odd_numbers(a)
```

```
[37]: [1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25]
```

Now we don't have to rewrite long code for checking if the number is odd or not, we will just call this function.

5.5 Note

For your final assignment part related to market basket analysis, functions are really important as you will be calling functions most of the time.

5.6 Task for you

Write a function which ask users to enter name of an item that they recently bought. Check if the item is in the list ["Milk", "Butter", "Apple"] and recommends a discount on Oranges. Else it prints "Happy shopping with us!"

```
[ ]:
```