Programminglesson-12-09

September 15, 2020

1 Welcome to your first class!

Let's start with looking at an already built program, and code behind it.

The below class definition and methods(another name for functions when are defined inside a class and associated with objects) is an examaple of a basic program.

```
In [7]: # This are the libraries used in this Program
        from IPython.display import clear_output
        import time

        class Classroom(object):
            '''
            This class simulates a classroom, and just like a classroom it has a Teacher, Stud
            '''
            def __init__(self):
                '''
                This initializes a classroom, taking in different details and storing it in di
                '''
                self.welcomeDisplay()


            def welcomeDisplay(self):
                '''
                This functions displays a short greeting welcoming to the classroom.
                '''
                print("Welcome to the Classroom.")
                self.changeScreenin(2)
                print("In this classroom, we will learn as best as we can.")
                self.changeScreenin(5)
                print("and we will do it with joy! :)")

            def changeScreenin(self, seconds):
                '''
                This clears the screen after few seconds.
                '''
                time.sleep(seconds)
                clear_output(wait=False)
```

```
In [8]: new_class = Classroom()
```

```
and we will do it with joy! :)
```

The `new_class` created here is an example of the *user defined* object. An object has generally attributes(variables) and methods(functions) and assessed with `.` notation. The following is an example

```
In [9]: new_class.welcomeDisplay()
```

```
and we will do it with joy! :)
```

```
In [42]: print("Hello World") #This line prints Hello World
```

```
Hello World
```

1.0.1 Now that we have looked a general program, let's go to the basic elements.

Goal:
Revising basic concepts before getting to data anaytics.
Concepts being revised

1. Programming:
   1. What?
      Programming is a tool for problem solving
   2. Why?
      Makes our life easier, even though sometimes technology created without sufficient foresight can lead to unexpected problems
   3. How?
      Using different languages avaliable, such as python, CPP, Java, Kotlin, being some of the examples. Generally, languages can also problem specific. For example, Kotlin being used for android development
2. Computational thinking vs language
   Even though learning a programming language plays integral role in programming, majorily a programmer needs to develop computational thinking. The ability to create by taking a creation apart into blocks, and bringing them together. During this course we will explore together how to pratice and learn this methodology.
3. Basic elements of the language

```
1. Variables
```

```
In [10]: name = input("Please Enter your name\n")
```

```
Please Enter your name
Flavia
```

Variables are labels for storing values. They can stores numberic values(integers, decimals), strings(such as ‘a’, ‘apple’), and Boolean values(True or False). After being initialized, they can be accessed in the program by calling the name such as

```
In [12]: name

Out[12]: 'Flavia'
```

2. Statements

A single line of code that is executable is a statement. All that the have written till now are statements.

3. Functions

We have already used a function in the above code known as input(), which takes in an input from the user in string format. In essence, every function has a function. Python has a lot of built in functions. but we can also define them.

```
In [13]: # This is an example of a built in function
         print("Hello Flavia")

Hello Flavia
```

Traditionally, programming lessons start with printing *Hello World* on the screen. In python3 this is done using `print()` function.

```
In [15]: def itssayhello():# You can also take in input parameters here, instead of leaving it
             '''
             It says hello based on the student name
             '''
             name = input("Please enter your name\n")
             print("Hello",name)
             return # A function ends with a return statements, and it is also possible to ret

         itssayhello() # This is how the function that needs to be used is called, similar to

Please enter your name
Julio
Hello Julio
```

User defined functions are created and called in the above format. `''' docstring '''` The docstring written in the above function is a multiline commment which is used to document the function and its usage, and can accessed by calling `help(function_name)`

```
In [45]: help(print)

Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
```

```
In [46]: help(itssayhello)

Help on function itssayhello in module __main__:

itssayhello()
    It says hello based on the student name
```

4. Control Flow

Control flow refers to order in which the program statements are executed. They can be done through conditionals such as `if else` statements or through loops such `for` or `while` statements.

```
In [49]: a = 6
         if a > 0:
             print("If is executed")
         else:
             print("else is executed")

         print("This program is done")

If is executed
This program is done
```

The above snippet of code illustrates control flow using `if-else` statements. General version of this way of control flow is `if elif else` statement. The following snippet gives an example of this approach.

```
In [5]: n = int(input("Please enter a number between 0 to 9\n"))

        if (n > 5) & (n % 2 == 0): # This reads as condition number greater than 5 AND gives r
```

```
            print("The given number is greater than 5 and is even")
        elif (n < 5) & (n % 2 == 0):
            print("The given number is less than 5 and is even")
        elif (n > 5) & (n % 2 != 0):
            print("The given number is greater than 5 and is odd")
        elif (n < 5) & (n % 2 != 0):
            print("The given number is less than 5 and is odd")
        else:
            print("The number you have entered is 5")
```

```
Please enter a number between 0 to 9
5
The number you have entered is 5
```

```
In [53]: a = 10

         while a > 0:
             a = a - 1 # You can also write it as a -= 1, which means the same
             print(a)

         print("This program is done")
```

```
9
8
7
6
5
4
3
2
1
0
This program is done
```

The above code shows how a `while` loop is implemented. The control of the program stays inside the body of the `while` loops as long as the condition(`a > 0`) is evaluated to be true. Similarly loops can implemented be by `for` statement.

```
In [16]: for i in range(10): # can you figure out what range() function does?
             print(i)
```

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

```
6
7
8
9
```

In this case, the loop remains active as long as the iterator `i` is in in the list `[0,1, 2,..,9]`

`5. Libraries`

Libraries are modules containing groups of functions together.
Examples of libraries are numpy, scipy, pandas. These example libraries are used prominently in data analysis through python. *When a person says he or she is well versed in a language, it ought to imply the person is well versed in the languages libraries, or atleast is cabable of using them.* To use a libraries function, all one needs is to import the library. This done through `import library_name`. The following code snippet gives an example of this code snippet.

```
In [20]: import numpy as np

         a = [1, 2, 4, 5]
         sum_of_list = np.sum(a)
         print("Sum of all the elements is", sum_of_list)
```

```
Sum of all the elements is 12
```

`6. Classes or Object Oriented Programming`

Remember we started our class with defining a `class`. Refer to the beginneing to see how a class is defined, and how it is being used. Most of the data analysis can be done without touching classes, but it is neverthless a good idea to what they are how, and they can be used to create `objects`.
And remember everything in python is an object. This mean that everything we have seen or will see can have attributes and properties that will be accessed using the `.` notation.

1.1 Finally, challenge before next class:

1.1.1 Goal:

To build something using some of the concepts learnt in the first class.
Assignment 1
Write a program that takes an input for any number, and checks whether it is a prime number or not?
Assignment 2
Write a program that takes the currency in Euros, and converts it your home currency.
Assignment 3 Write a program to asking how your friend feels, and respond back with sentence, based on how he or she is feeling.
Example, input: sad, output: Don't worry, everything will be alright soon.
You can choose one of them to practise or all of them. :) Happy coding! ;)

```
In [ ]:
```