Programminglesson-05-09

September 5, 2020

1 This is our first python class

1.1 Overall structure of the course

- Learn the fundamentals of programming on python.
- Make a small animation or game on scratcj
- Checking out different gaming libraries in python
- Making your first python based game.

1.2 What we going to learn today?

1. What is programming?
   - It is passing a set of instructions to the computer to do something we like.
2. Why do we need to program?
   - It is a tool to solve problems
3. How do we program?
   - Languages
     1. High level programming languages - python, cpp, java, scratch
     2. Low programming languages - assembly or machine language
   - Computational thinking
     1. It is more about being able to think a certain way than using a particular language
4. Making our first python program!
   - Let's say Hello to the World!

2 This program says hello world

```
In [44]: print('Hello World')
```

```
Hello World
```

2.1 Comments

It is very useful to write comments. They are written next to your line of code, and is a way of telling what your code does. They are written using #

Example:

```
In [50]: # this is a comment
         print('This code shows how to write comments')
```

```
This code shows how to write comments
```

Note that the comment is not executed when the program is run.

2.2 Below is an example of a example of small program which was written before the class

```
In [ ]: # This is a short code written to introduce the student to the class

        from IPython.display import clear_output

        def sayHello(studentname):
            clear_output(wait=False)
            print("Hello", studentname)
            print('Welcome to the class')
            print('I wish you all the best!')
            return
        name = input('Please enter your name: ')
        sayHello(name)
```

3 Things we learn today

3.1 Variable

```
- Variables are things used to store values
- Variables can be of different type. It can be integer, string, boolean, or float.
- You can check the type by `type(variable_name)`
- It is also possible to try converting from one type to another.
```

3.1.1 Variables can be combination of letters, numbers, . *They should start with a letter or* . They cannot be keywords either

Keywords are special words such as `type`. You cannot therefore name your variable as them, as they are already used by python. #### They are case sensitive. variable go is not same as `Go`. ## Example of variables

```
In [46]: # This is an integer variable
         integer = 625
```

```
# This is a string variable
name = 'Julio Mateos'

# This is a floating variable
decimal_number = 3.14

# This is a boolean variable
true_statement = True # It also be False
```

3.2 Checking type of a variable

```
In [48]: # This program checks the type of the variable
         print(type(name))
         print(type(true_statement))
         print(type(integer))
         print(type(decimal_number))
```

```
<class 'str'>
<class 'bool'>
<class 'int'>
<class 'float'>
```

3.3 Type conversion

Now let's convert a float to integer and see what happens.

```
In [51]: print('Orginal number: ', decimal_number)
         converted_integer = int(decimal_number)
         print('After conversion: ', converted_integer)
```

```
Orginal number:  3.14
After conversion:  3
```

3.3.1 How about converting from integer to float?

```
In [54]: print('Orginal number: ', integer)
         converted_float = float(integer)
         print('After conversion: ', converted_float)
```

```
Orginal number:  625
After conversion:  625.0
```

3.3.2 *You can play with and see conversion from and to, between other types.*

3.4 Operators

There are 7 basic operators in python. They are `+`, `-`, `*`, `/`, `**`, `//` and, `%`. They can act between variables. The functions of them are shown in code below.

```
In [56]: # This block of code shows the function of different operators.
         num1 = 4
         num2 = 5
         print(num1+num2)
         print(num1-num2)
         print(num1*num2)
         print(num1/num2)
         print(num1//num2) # quotient
         print(num1%num2) # remainder
         print(num1**num2) # power to
```

```
9
-1
20
0.8
0
4
1024
```

3.5 Statements

A single line of executable code is called a statement. Everything we have written till now are lines of statements.

3.6 Functions

Functions are groups of statements put together, than serve to do something. They help in replacing multiple statements into a single line of code for later use.

An example is of an `add` function.

```
In [63]: # This is definition of simple function that adds two numbers together and returns th

         def add(x,y):
             sum = x + y
             return sum
```

3.6.1 Function definition

As seen above the syntax/code of defining a function is

```
   def function_name(input_parameter1, input_parameter2, input_parameter3):
statments     return something
```

3.6.2 Calling a function

To call a function we write the code, `function_name(input)`, as shown below for `add` function.

```
In [64]: add(4,5)
```

```
Out[64]: 9
```

3.6.3 *It is also possible to store the return value of a function in a variable*

```
In [66]: value_from_function = add(0,5)
         print(value_from_function)
```

```
5
```

3.7 Now let's generalize our add function.

Instead of each time giving the numbers to be added, we take input from the user, and display the end result. **Remember that the `input()` function takes values in string type, and therefore we need to convert to int/float type when we expect integers or decimal numbers**

```
In [68]: number1 = int(input('Enter your first number: '))
         number2 = int(input('Enter your second number: '))

         result = add(number1, number2)
         print('Your result is', result)
```

```
Enter your first number: 12
Enter your second number: 32
Your result is 44
```

3.7.1 *Similarly it is very easy to make a substract function based on the concepts and style we already learnt.*

```
In [71]: def substract(x,y):
             result = x - y
             return result


         number1 = int(input('Enter your first number: '))
         number2 = int(input('Enter your second number: '))

         result = substract(number1, number2)
         print('Your result is', result)
```

```
Enter your first number: 34
Enter your second number: 14
Your result is 20
```

3.7.2 It is also possible to create functions which do not take inputs or return values such as

```
In [69]: # A function which says hello world.
         def sayHello():
             print('Hello World')
```

```
In [70]: sayHello()

Hello World
```

3.7.3 *A fun thing to try would be make a function on your own, taking the name of the person as input.*

And always feel free to ask questions! :)

```
In [ ]:
```