# Python(Data Science)

# Introduction:-

Python is a high-level, interpreted scripting language developed in the late 1980s by Guido van Rossum at the National Research Institute for Mathematics and Computer Science in the Netherlands. The initial version was published at the alt.sources in 1991, and version 1.0 was released in 1994. Python 2.0 was released in 2000, and the 2.x versions were the prevalent releases until December 2008. At that time, the development team made the decision to release version 3.0

* General Purpose : Build anything
* Open Source! Free!
* Python Packages, also for DataScience
  + Many Applications and Fields

Version Python3.x

# The Python Interface

In the Python script on the right, you can type Python code to solve the exercises. If you hit *Run Code* or *Submit Answer*, your python script (script.py) is executed and the output is shown in the IPython Shell. *Submit Answer* checks whether your submission is correct and gives you feedback.

You can hit *Run Code* and *Submit Answer* as often as you want. If you're stuck, you can click *Get Hint*, and ultimately *Get Solution*.

You can also use the IPython Shell interactively by simply typing commands and hitting Enter. When you work in the shell directly, your code will not be checked for correctness so it is a great way to experiment.

* Experiment in the IPython Shell; type 5 / 8, for example.
* Add another line of code to the Python script on the top-right (not in the Shell):

print(7 + 10).

* Hit *Submit Answer* to execute the Python script and receive feedback.

# Example, do not modify!

print(5 / 8)

print("shelly")

print(250 \* 5)

print(330 / 3)

print(490 + 310)

print(5589 - 3266)

# Print the sum of 7 and 10

print(7+10)

# When to use Python?

Python is a pretty versatile language. For which applications can you use Python?

##### Possible Answers

* You want to do some quick calculations.
* For your new business, you want to develop a database-driven website.
* Your boss asks you to clean and analyze the results of the latest satisfaction survey.
* All of the above.

# Any comments?

Something that Hugo didn't mention in his videos is that you can add **comments** to your Python scripts. Comments are important to make sure that you and others can understand what your code is about.

To add comments to your Python script, you can use the # tag. These comments are not run as Python code, so they will not influence your result. As an example, take the comment in the editor, # Division; it is completely ignored during execution.

Above the print(7 + 10), add the comment

# Addition

# Division

print(5 / 8)

#Addition

print(7 + 10)

# Python as a calculator

Python is perfectly suited to do basic calculations. Apart from addition, subtraction, multiplication and division, there is also support for more advanced operations such as:

* Exponentiation: \*\*. This operator raises the number to its left to the power of the number to its right. For example 4\*\*2 will give 16.
* Modulo: %. This operator returns the remainder of the division of the number to the left by the number on its right. For example 18 % 7 equals 4.

The code in the script gives some examples.

Suppose you have $100, which you can invest with a 10% return each year. After one year, it's 100×1.1=110 dollars, and after two years it's 100×1.1×1.1=121. Add code to calculate how much money you end up with after 7 years, and print the result.

# Addition, subtraction

print(5 + 5)

print(5 - 5)

# Multiplication, division, modulo, and exponentiation

print(3 \* 5)

print(10 / 2)

print(18 % 7)

print(4 \*\* 2)

# How much is your $100 worth after 7 years?

print(100\*1.1\*\*7)

# Variables

* Specific, case-sensitive name
* Call up value through variable name
* 1.79 m - 68.7 kg

**Calculate BMI**

height = 1.79

weight = 68.7

BMI = weight / height \*\* 2

print(BMI)

print(type(BMI))

x = "body mass index"

y = 'This works too'

print(type(x))

print(type(y))

z = True

print(type(z))

* Different type = Different behaviour!

print(2 + 3)

print('ab' + 'cd')

# Variable Assignment

In Python, a variable allows you to refer to a value with a name. To create a variable use =, like this example:

x = 5

You can now use the name of this variable, x, instead of the actual value, 5.

Remember, = in Python means *assignment*, it doesn't test equality!

* Create a variable savings with the value 100.
* Check out this variable by typing print(savings) in the script.

# Create a variable savings

savings = 100

# Print out savings

print(savings)

print(type(savings))