www.uib.cat

Introduction to Python

Centre Balear de Biodiversitat

Tommaso Cancellario
Laura Triginer

Universitat de les Illes Balears

July 2023

What is Python?

- High level programming language
- Published in 1991
- Simple, versatile and legible
- One of the most popular programming language worldwide.

This language is used in web development, data science, machine learning and science computation. As Python is a dynamic writing language, it's not necessary to specify the data type when declaring it, that makes Python an easy and flexible code language.

Who created Python?

Python was created by Guido Van Rossum, and first released on February 20, 1991. While you may know the python as a large snake, the name of the Python programming language comes from an old BBC television comedy sketch series called *Monty Python's Flying Circus*.

Increase of people using Python

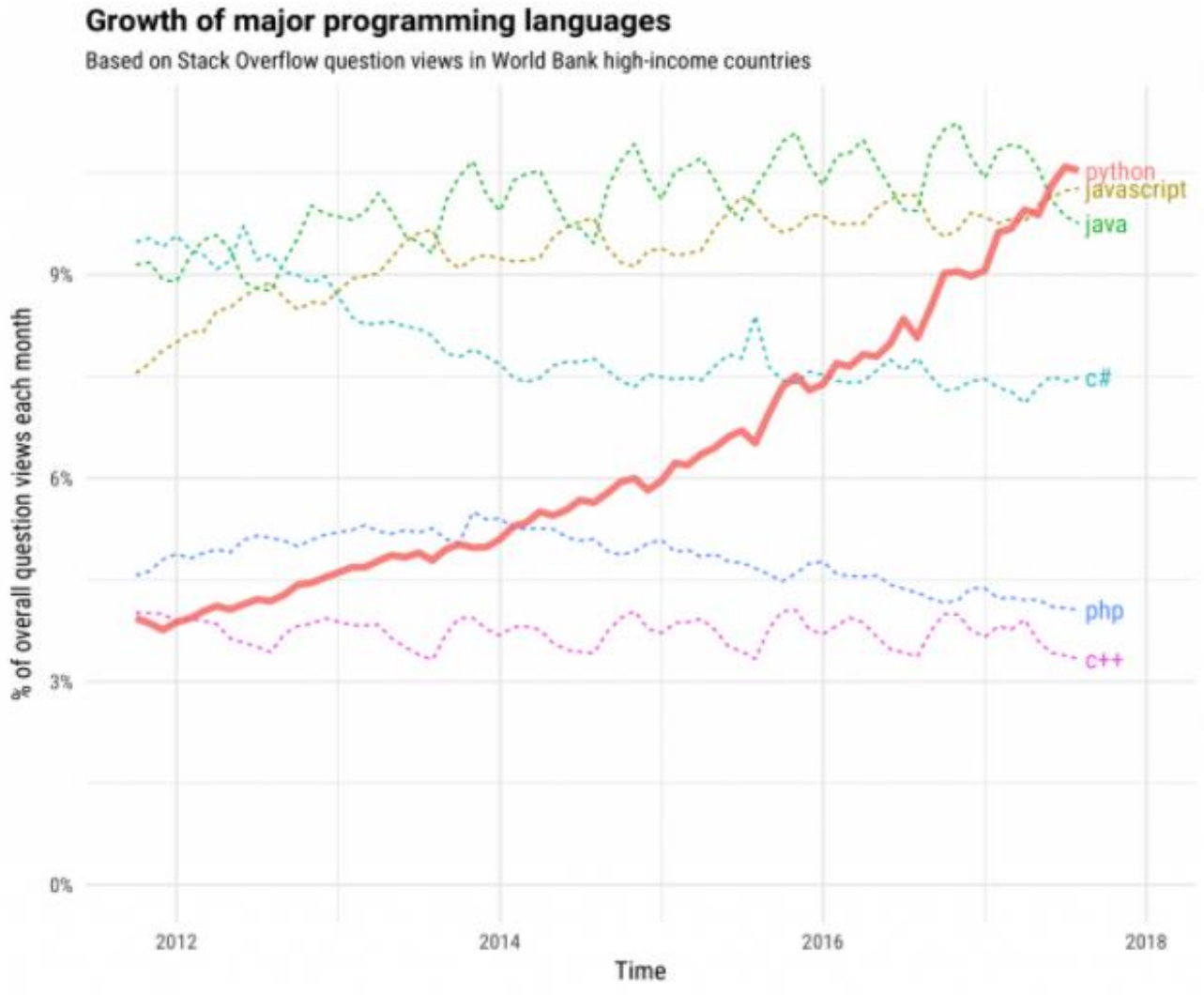


Imagen: https://sitiobigdata.com/2019/01/31/extraordinario-crecimiento-python/

Installing Python

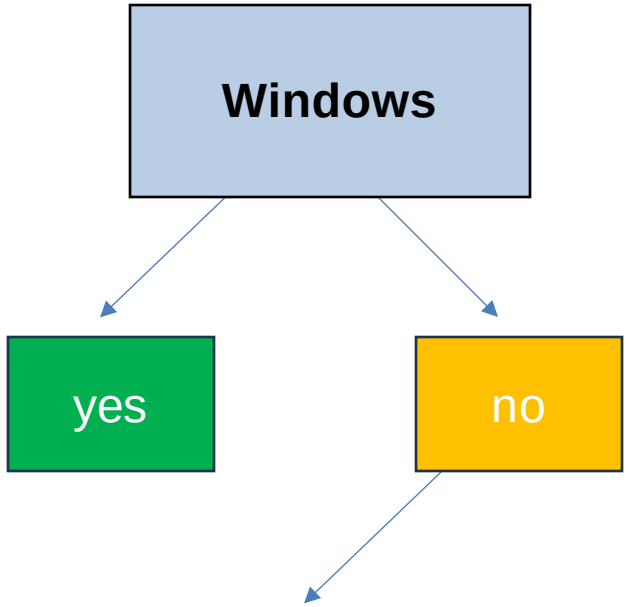
Python (programming language)

Linux: https://www.python.org/downloads/source/
macOS: https://www.python.org/downloads/macos/
Windows: https://learn.microsoft.com/en-us/windows/python/beginners

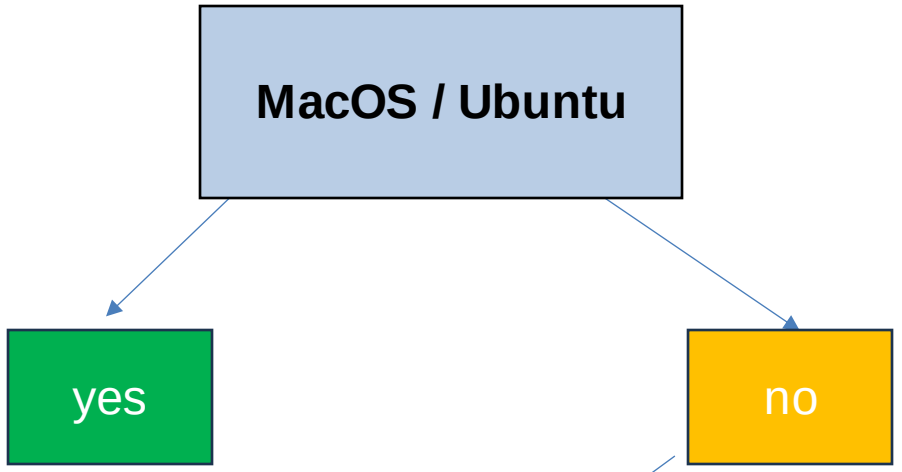
VisualCodeStudio (development environment)

Linux: https://code.visualstudio.com/Download
macOS: https://code.visualstudio.com/Download
Windows: https://code.visualstudio.com/Download

Is Python installed?



Open a terminal and write: "python" and check the version of Python.



Install Python directly from the Microsoft Store

https://www.datacamp.com/blog/how-to-install-python

MacOS: https://docs.python-guide.org/starting/install3/osx/

Ubuntu: sudo apt update
sudo apt install python3

Where is Python installed?

`C:\Program Files\Python37`

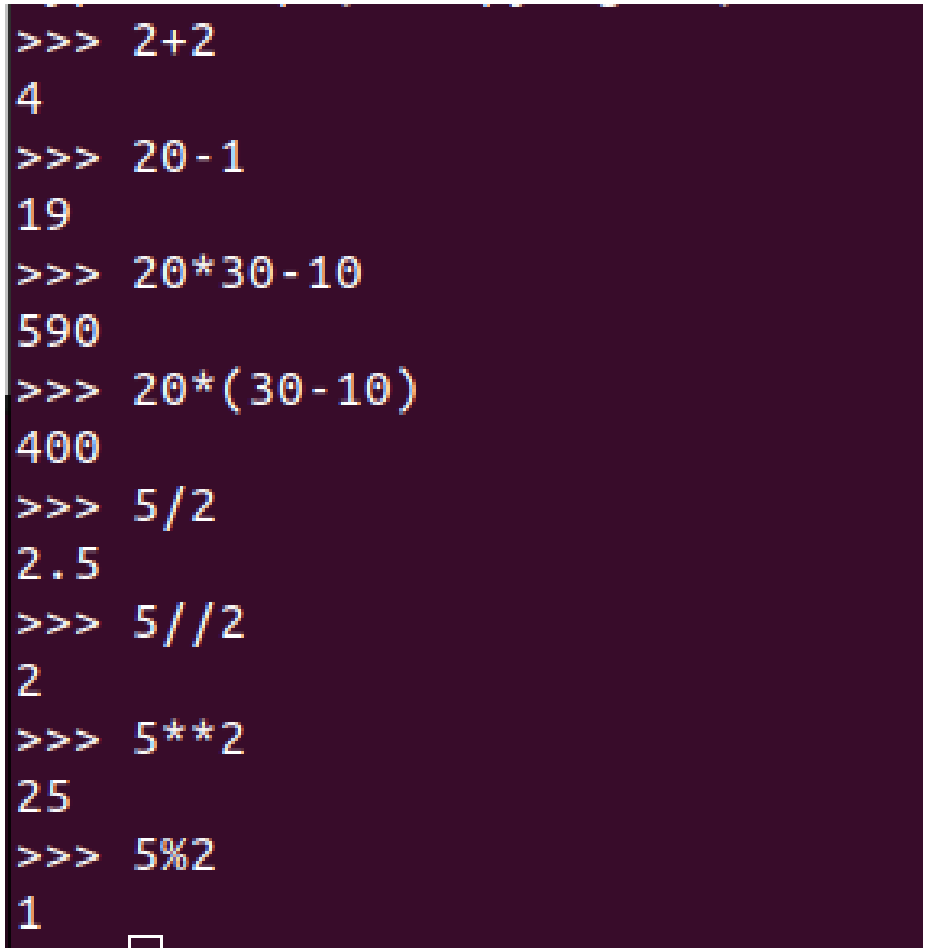
/usr/bin/python37

Differences between Python 2 and 3

	Python 2	Python 3
Release date	2000	2008
Syntax	Complex and difficult	Readable and easier
Performance	Slower performance	Improved performance
Print	Print "hello"	Print ("hello")
Integer division	Integer value, decimals truncated	Float value
Backward compatibility	Easy to convert P2 -> P3	P3 -> P2 not compatible
Libraries	Most libraries are not compatible	Most libraries are not compatible

Using Python as a calculator

Let's start! Open the terminal and write "python"



For addition: '+'

For substraction: ' -'

For multiplication '*'

Parentheses have priority, such as in normal maths

For division: '/' returns a float

To have the integer of a division '//'

To calculate the exponential

To obtain the reminder of a division

Setting variables in Python

In Python we don’t need to set the variable type. However, there are some built-in functions that need an special variable type.

» *Name of variable* = *value*

```
>>> width = 20
>>> height = 5*9
>>> area = width * height
>>> area
900
>>> ars
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'ars' is not defined. Did you mean: 'abs'?
```

Variable types - Numbers

Check the variable type using

»type(variable name)

Integers (**int**)

1, 2, 3, 4, 5, 6 .... 12932449824, 1290173128472384632 78, 0, -1, -2, -2138123712...

A whole number, positive or negative, **without decimals** of unlimited length.

```
>>> x = 12123141
>>> type(x)
<class 'int'>
```

Float (**float**)

1.345, 53.3234, 9.2515, -34324.3213, 3e-5, 4.0e5,

They present real numbers and are written **with decimal point dividing the integer** and the fractional parts.

```
>>> y = 15e-5
>>> type(y)
<class 'float'>
```

Variable types - Strings

Alphanumeric variables or character variables, are treated as **text**.

“ *text* ” or ' *text* '

Strings are arrays, square brackets can be used to access elements.

First element index is 0.

```
>>> word = "Python"
>>> word[0:2]
'Py'
>>> word[-1]
'n'
>>> word[0]
'P'
>>> word[1]
'y'
```

Variable type - String

Python helps you checking where the error is and gives you other options in case you misspelled it.

```
>>> word = "Pythoo"
>>> word[1]
'y'
>>> word[5]="n"
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
```

Strings cannot be modified.

You can concatenate **strings** using "+"

```
>>> s1 = "My name is "
>>> s2 = "Laura"
>>> s3 = s1 + s2 + " " +"Triginer"
>>> s3
'My name is Laura Triginer'
>>>
```

Variable type – Boolean and comparison operators

Used when you need to know if an expression is **True** or **False**

```
>>> print (10 == 10)
True
>>> print (10 > 1 )
True
>>> print (10 < 1 )
False
```

Comparison operators are used to compare two values

==	Equal	9 == 9
! =	Not equal	9 != 8
>	Greater than	9 > 8
<	Less than	9<10
> =	Greater than or equal to	9 >= 8
< =	Less than or equal to	8 <=9

Variable type – Logical operators

Used to combine conditional statements

and	Returns true if both statements are true
or	Returns true if one of both statements is true
not	Reverse the result, returns false if the result is true
Is	Returns True if both variables are the same object
Is not	Returns True if both variables are not the same object

```
>>> x = 10
>>> y = 5
>>> x > 8 and y < 7
True
>>> x > 9 or y == 5
True
>>> not(x>7 and x>11)
True
>>> not(x>7 and x>8)
False
```

```
>>> x is not y
True
>>> x is y
False
```

Variable types – Lists

Used to store multiple items in a single variable

```
>>> notas = [8, 10, 9, 7, 5, 7.8, 8.3]
>>> notas
[8, 10, 9, 7, 5, 7.8, 8.3]
>>> notas[1]
10
>>> notas[-1]
8.3
>>> notas[:]
[8, 10, 9, 7, 5, 7.8, 8.3]
>>> notas + [6, 5.4, 3]
[8, 10, 9, 7, 5, 7.8, 8.3, 6, 5.4, 3]
>>> notas[1] ='a'
>>> notas
[8, 'a', 9, 7, 5, 7.8, 8.3]
>>> notas.append(48)
>>> notas
[8, 'a', 9, 7, 5, 7.8, 8.3, 48]
```

- Written between [ ]
- Are ordered
- Contain any arbitrary objects
- Can be accessed by index
- Are mutable.
- Can have duplicated elements

Variable types – Sets

Used to store multiple items in a single variable

- Unordered
- Unchangeable, but you can remove and add new items
- Do not allow duplicate values
- Written in { }

You cannot access set objects, as they are unordered and cannot know in which index are they placed

```
>>> fruits_set = {"apple", "banana", "grapes"}
>>> fruits_set.add("strawberries")
>>> print(fruits_set)
{'strawberries', 'grapes', 'apple', 'banana'}
```

```
>>> print(fruits_set[1])
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'set' object is not subscriptable
```

Variable types – Dictionaries

Used to store data values in key:value pairs.

- Ordered (from v.3.7)
- Changeable
- Do not allow duplicate values
- Written in { }

key: value

```
>>> data_dictionary = {"ID":"1544","Name":"Laia","Year":"2009", "Clothes":["shirt","trousers","socks"]}
>>> n = data_dictionary.keys()
>>> data_dictionary = {"ID":"1544","Name":"Laia","Year":"2009", "Clothes":["shirt","trousers","socks"]}
>>> data_dictionary.keys()
dict_keys(['ID', 'Name', 'Year', 'Clothes'])
>>> data_dictionary.values()
dict_values(['1544', 'Laia', '2009', ['shirt', 'trousers', 'socks']])
```

Variable types – Dictionaries

You can create a dictionary with another dictionary inside, that's called **nested dictionaries**

```
>>> colleages = {
...     "person1": {
...             "name":"Yves",
...             "year":"1986"
...     },
...     "person2": {
...             "name":"Anna",
...             "year":"1975"
...     }
... }
>>> colleages
{'person1': {'name': 'Yves', 'year': '1986'}, 'person2': {'name': 'Anna', 'year': '1975'}}
```

You can create different dictionaries and then create one dictionary that will contain the others.

Conditionals – if... else, elif

Python uses the logical conditions such as the previous explained.

» if *condition_statement* **:**
then do....
....
else *condition_statement* **:**
then do...
else**:**
do...

It's highly important to ident correctly in Python.

==	Equal	9 == 9
! =	Not equal	9 != 8
>	Greater than	9 > 8
<	Less than	9<10
> =	Greater than or equal to	9 >= 8
< =	Less than or equal to	8 <=9

```
>>> x = int(input('Please, enter an integer value: '))
Please, enter an integer value: 32
>>> if x > 10:
...     print("x is bigger than 10")
... elif x == 10:
...     print("x is 10")
... elif x < 10:
...     print("x is less than 10")
...
x is bigger than 10
```

Loops – while

A set of statements is executed as long as a condition is true

```
>>> a,b=0,1
>>> while a<20:
...     print(a)
...     a,b=b,a+b
...
0
1
1
2
3
5
8
13
>>> print ('El siguiente número es',a)
El siguiente número es 21
```

It's important to increase the value, otherwise the loop will last forever.

Loops – while

Break statement: we can stop the loop even if the while condition is true:

```
>>> i=1
>>> while i < 10:
...     print(i," es un valor menor a 5")
...     i+=1
...     if i > 5:
...             break
...
1  es un valor menor a 5
2  es un valor menor a 5
3  es un valor menor a 5
4  es un valor menor a 5
5  es un valor menor a 5
```

Continue statement: we can stop the current iteration, and continue with the next:

```
>>> i=0
>>> while i<3:
...     i+=1
...     if i==2:
...             continue
...     print(i)
... else:
...     print("el valor de i es ", i)
...
1
3
el valor de i es  3
```

Else can be also used when the statement is no longer true

Loops – For

Used for iterating over a sequence, that can be a list, tuple, dictionary, set or a string.

Works as an interator method.

```
>>> words = ['Tom', 'Anna', 'Yves', 'Enrique', 'María','Laura']
>>> for w in words:
...     print(w, len(w))
...
Tom 3
Anna 4
Yves 4
Enrique 7
María 5
Laura 5
```

Break statement can be used as in while, to stop the loop before it has looped through all the items.

Continue statement can be used as in while, to stop the current iteration and continue with the next.

Loops – For

The range() function is used to loop through a set of code a specified number of times.

```
>>> for x in range(3):
...     print(x)
...
0
1
2
>>> for x in range(3,5):
...     print(x)
...
3
4
>>> for x in range(1,10,2):
...     print(x)
...
1
3
5
7
9
```

Loops – For

The **else** keyboard specifies a block of code to be executed when the loop is finished:

```
>>> for x in range(3,5):
...     print(x)
... else:
...     print("Finish")
...
3
4
Finish
```

A **nested loop** is a loop inside a loop

```
>>> color=['green','blue','yellow']
>>> size = ['long','short']
>>> for c in color:
...     for s in size:
...             print(s, c + ' hair')
...
long green hair
short green hair
long blue hair
short blue hair
long yellow hair
short yellow hair
```

Modules and libraries

Libraries is a collection of precompiled codes that can be used later on a program for some specific well-defined operations.

Is a collection of related modules.

It makes Python programming simpler, so you don't need to write the same code again and again

Commonly used libraries in Bioinformatics:

- Matplotlib: Eases plotting numerical data
- Pandas: Eases data analysis, data manipulation and data cleaning.
- Numpy: Supports large matrices and multi.dimensional data.

import library_name --> library_name.module()
import library_name as f1 --> f1.module()
from library_name import function --> module()

Scripts and Functions

Scripts are sets of commands that are executed one behind the other. Are written in a text editor or in a development program.

Functions is a block of code that runs when is called. You can pass data and can return data.

```
def function_name (input_param) :
        commands
        .
        .
        .
```

Calling a function:
***function_name*(parameter)**

```
>>> def fib(n):
...     a,b=0,1
...     while a < n:
...             print(a, end=' ')
...             a,b = b, a+b
...     print()
...
>>> fib(3000)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584
```

How to use functions in your script

In a function there can be different modules.

Function1.py

```
def my function(n):
        do...

def my_function2(m):
        do....

def my_function3(o):
        do...
```

1. Import function1

Will import all the modules inside the function1,py

» function1.my_function1(300)

2. Import function1 as f1

Will import all modules in function1 but with the alias f1.

» f1.my_function(300)

3. from function import my_function3

Will import the chosen module.

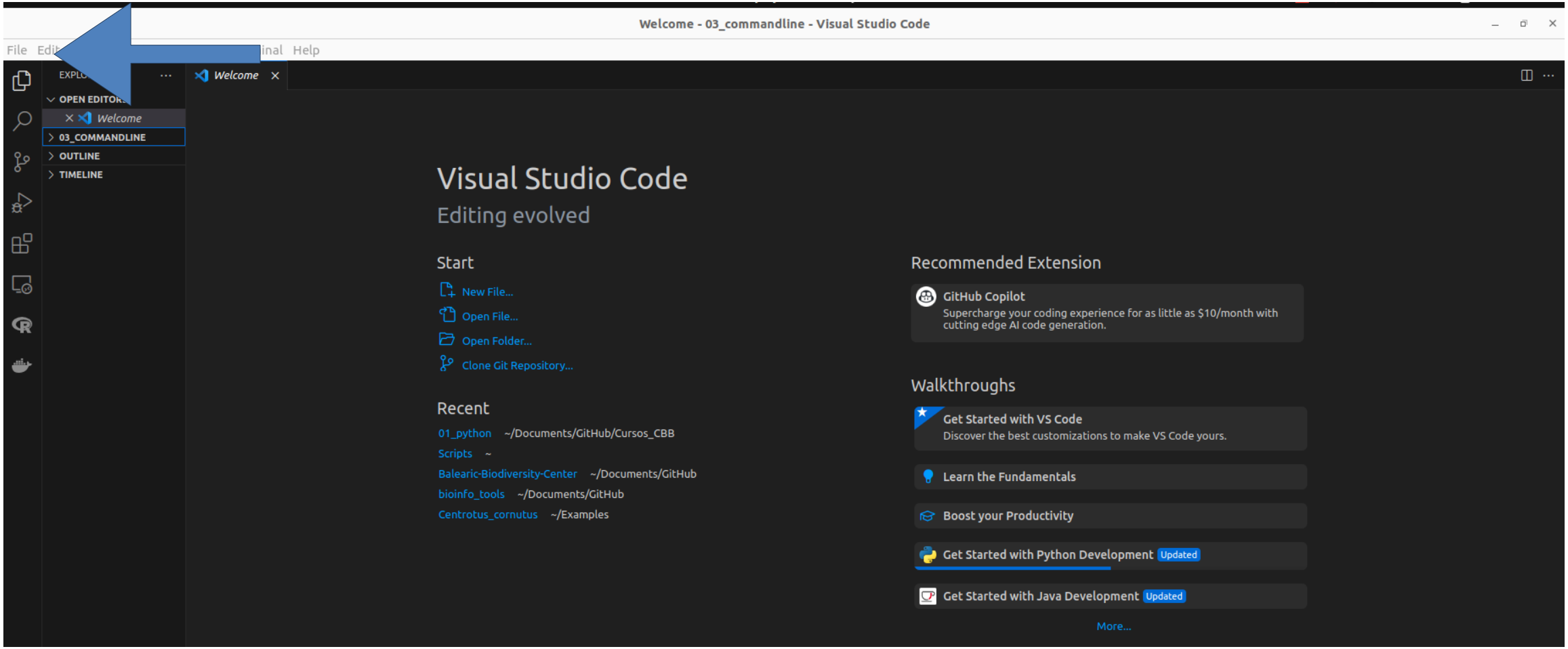
» my_function3(300)

Once you close the interpreter, you need to start all over...

Or you can use an interpreter, such as Visual Code Studio or a simple Text Editor and save your advances and scripts or functions.

Autofiils and makes syntax easier.

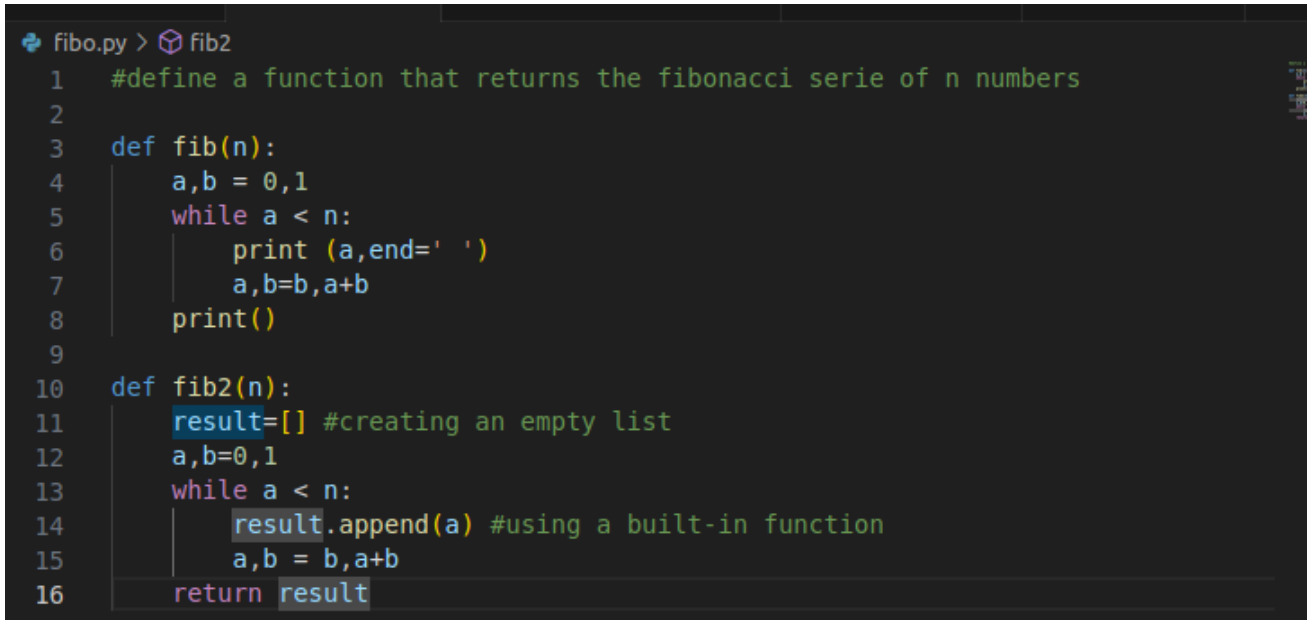
Visual Code Studio



Example

Steps:

1. Download ‘fibo.py’
2. Open the fibo.py file
3. Make sure your current directory is where the fibo.py file is.
4. Import fibo
5. Test the different functions ...



You can create your own functions and scripts, but there are many built-in functions that are ready to use.

Exercise 1

```
## Module 2. Introduction to Python ##
##Exercise 1

# import libraries
import random
from statistics import mean

# create variables
randomlist = []

# create and save 10 random numbers in n list-variable

for i in range(0,10):
    n = random.randint(1,30)
    randomlist.append(n)
print(randomlist)
```

Exercise 1

```
# 2. Sort the numbers
print(str(sorted(randomlist)))
# 3. Retrieve the maximum number of the list
print(str(max(randomlist)))
# 4. Retrieve the minimum number of the list
print(str(min(randomlist)))
# 5. Print the mean
### Option 5.1, not using library

list_mean = sum(randomlist)/10
print(str(list_mean))

### Option 2, using statistics library
print(str(mean(randomlist)))
```

Exercise 2

1. CREATE AND SAVE 20 RANDOM NUMBERS.
2. SORT THE NUMBERS
3. RETRIEVE THE MAXIMUM NUMBER OF THE LIST
4. RETRIEVE THE MINIMUM NUMBER OF THE LIST
5. PRINT THE MEDIAN
6. PRINT THE MODE

Exercise 3

```
## Module 2. Introduction to Python
## Exercise 3

# Libraries
import pandas as pd
from collections import Counter
from matplotlib import pyplot
from shapely.geometry import Point
import geopandas as gdp
from geopandas import GeoDataFrame
import folium as fol

# data path
file_path = "~/Documents/GitHub/Cursos_CBB/data/whale_data.csv"
## read data
file = pd.read_csv(file_path, sep = "\t")
```

Exercise 3

```
#Check the name of the columns
tags = list(file.columns)
print ('names of columns: ', tags)

count_years = (Counter(file["year"]))
count_samples = sum(Counter(file["year"]))
print('número de muestras: ', count_samples)
print('años en los que se ha llevado a cabo el estudio: ' , sorted(file["year"].unique()))

## Bar plot

pyplot.bar(count_years.keys(), count_years.values(), color='green')
pyplot.show()

## Google Maps Folium ##
map = fol.Map(location=[39.5, 2.745], zoom_start=8)
p_lat_long = []
p_lat_long  = list(zip(file['decimalLatitude'],file['decimalLongitude']))
for points in p_lat_long:
    fol.Marker(points, popup='Whale').add_to(map)
map.save('index.html')

##
```

Exercise 4

```
## Module 2. Introduction to Python
## Exercise 4

# Import the libraries that you use


## data path

## read data

# check the name of the columns

#Count the number of columns
# Which institution has been doing the studio?
# In which different months has the studio been done?

## Plot the months in which the observations have been done. Draw the bars in blue.
```

There are many books to learn, online courses..

As there are many many maaany people working and doing research using Python, there’s a huge community online.

Thank you for your attention