# Topic 2 - Intro to Python (2 hours)

July 5, 2022

#

Computational Statistics with Python

##

Topic 2: Basic Python programming

##

Expected lecture time: 3 hours

Giancarlo Manzi

Department of Economics, Management and Quantitative Methods

University of Milan, Milan, Italy

## 0.1 What we will see

- Some basic Python programming.
- After introducing the the language basics, we will cover some object-oriented and functional programming in Python.
- In addition, several relevant modules of the Python basic libraries will be introduced.
- We will then focus on Python paradigms and best practices for solving frequent tasks.
- Finally, we will study some basic libraries of data manipulation and visualization, as well as we will give some hints on programming for machine learning with Python

## 0.2 Specific points in this Python introduction

- The Python interpreter
- Variables dynamic typing.
- Null values.
- Tuples and lists
- Conditional clauses and cycles.
- List slices and list comprehension.
- Dictionaries.

### 0.3 Specific points in this Python introduction (cont'd)

- The numpy library and the numpy array.
- Main operations with numpy.
- Pandas: Python Data Analysis Library
- Indices, Series and Pandas Data Frame.
- Other data structures (pkl).
- CSV importing, simple statistics, simple graphics (Pandas, Matplotlib).
- Classes and objects of the Scikit-learn library.
- Some statistics applications (probably tomorrow).

## 1 Different programming typology (***paradigms***)

- *Procedural* programming:
  - FORTRAN;
  - Pascal;
  - COBOL;
  - C;
  - Algol.
- *Object-oriented* programming (*untyped*):
  - Simula;
  - SmallTalk;
  - Python;
  - Ruby;
  - JavaScript.
- *Object-oriented* programming (*typed*):
  - C++;
  - Java;
  - Scala;
  - C#;
  - JavaScript.
- *Functional* programming (*untyped*):
  - LISP;
  - Scheme;
  - Racket.
- *Functional* programming (*typed*):
  - ML;
  - OCaml;
  - Haskell;
  - F#.
- *Logical* or *declarative* programming:
  - Prolog;
  - Curry;
  - SQL.

## 2 Interpreters, compilers and virtual machines

- Let $ L_S $ be a *source* programming language, $L_T$ a *target* language and $L_I$ an *implementation* language.
- Then an *interpreter* for $L_S$ is a $L_I$ program that executes $L_S$ programs.
- When both $L_S$ and $L_I$ are low-level, an interpreter is called *virtual machine*.
- A *translator* from $L_S$ to $L_T$ is a $L_I$ program that translates programs written in $L_S$ into *equivalent* programs in $L_T$.
- When $L_T$ is low-level, a translator to a $L_T$ program is called a *compiler*.

## 3 Interpreter language

- An interpreter is therefore a program that can execute programs using only the source code.
- It executes a program in a high-level language without compiling it and transforming it into object code.
- When using an interpreter language, the instructions are executed in the language used and the interpreter will immediately translate them into machine language instructions.
- Unlike an interpreter, a compiler does not execute the program it receives as input, but translates it into machine language (by storing the object code on file ready for direct execution by the processor).

## 4 Python: a multi-paradigm language

- Most programming languages support more than one programming paradigm to allow programmers to use the most suitable programming style and associated language constructs for a given job.
- Python is a *multiparadigm* programming language.
- When using a programming language there are many different ways to get the same output.
- For example we want to calculate $(4 + 4) \cdot (8 + 8)$.
- Mandatory or procedural programming: proceed in sequence. Example:
- `python          suma1 = 4+4        suma2 = 8+8          producto = suma1  * suma2          print(producto)          >>128`

[1]:
```python
suma1 = 4+4
suma2 = 8+8
producto = suma1  * suma2
print(producto)
```

```
128
```

## 5 Python: a multi-paradigm language (2)

- *Functional* programming: we use functions. Examples:
- `python          print (prod(         suma(8,8), suma (4,4)          ) )         >>128`

[3]:
```python
def som(c,d):
    summa = c + d
    return summa
```

[4]:
```python
som(3,4)
```

[4]:
```
7
```

[5]:
```python
def prod(a,b):
    product = a * b
    return product
```

[6]:
```python
prod(som(8,8),som(4,4))
```

[6]:
```
128
```

## 6 Python: a multi-paradigm language (3)

- *Object-oriented* programming: we create objects with properties and methods. Example:
- `python          suma1 = Sumador()          suma2 = Sumador() producto = Moltiplicador()          suma1.somma(4,4)          suma2.somma(8,8) producto.prod(suma1,suma1)          print(producto.getRisultato()) >>128`
- Python is "object-oriented".

[7]:
```python
class Adder:
    def func(self):
        a=4
        b=4
        sum = a+b
        print(sum)

ob = Adder()
ob.func()
```

```
8
```

## 7 Intro to Python: Anaconda

- Anaconda is a distribution of software, platforms, utilities, etc. useful for the development of data science.
- It is a "package manager" for Python, in the sense that the main data science packages are built into it.
- He is also an "environmental manager", in the sense of managing all the environment that Python needs for development.
- It is a collection of nearly 2000 very advanced packages.
- It is open and easy to install, and has behind it a community of developers who give continuous information on the network important for development.

## 8 Intro to Python: Anaconda (2)

- Installation takes place at www.anaconda.com/distribution
- Select the last version of Python for your OS.

## 9 Intro to Python: Anaconda (3)

- Select a destination folder to install Anaconda and click the "Next" button.
- Choose whether to register Anaconda as the default Python
- Click on the "Install" button.

## 10 Intro to Python: Anaconda (3)

- Anaconda Navigator is a desktop application that allows you to launch applications and easily manage packages, environments and channels without using terminal commands.

## 11 Intro to Python: the Jupyter notebook

- The Jupyter Notebook is an interactive open web application that allows you to share documents, prepare slides, text documents and simultaneously develop Python code.
- We will use it throughout the course. A more experienced programmers platform is *Atom. Visual Studio* is also popular.
- Jupyter is set up as if it were a web page (and in fact text, comments and code can be saved in html format as a web page).

## 12 Intro to Python: how Jupyter works

- Once launched, a Jupyter server "works" through port 8888 (by default), or through an ad hoc specified port.

- Jupyter should start from Anaconda directly; otherwise you have to open a web page at https: // localhost: 8888.
- We open a new notebook with "New" + "Python3": a notebook file with the extension * .ipynb * will be created (python files on other platforms have the extension * .py *)

[8]:
```
# Conversione Euro-Sterlina e viceversa

# Euro da cambiare in sterline:
Euro = 20

# Sterline da cambiare in Euro
Sterline = 20

#Tasso di cambio al 14 ottobre 2019:  1 sterlina = 1.145
tassoSE= 1.145

#Tasso di cambio Euro-Sterlina

tassoES = 1/tassoSE

# Sterline da Euro

print("Conversione da euro in sterline")

print("Al tasso di cambio del 14 ottobre 2019:", Euro , " Euro sono ", Euro *␣
 ↪tassoES, "sterline")
```

```
Conversione da euro in sterline
Al tasso di cambio del 14 ottobre 2019: 20  Euro sono  17.467248908296945
sterline
```

## 13 Data types in Python

- Numeric values: float, integers, complex.
- Strings.
- Logical values.
- Etc.

## 14 Numerical values

- `int`: defines an integer with any number of digits.
  - `In []: a = 77`.
  - `In []: b = 8888888888888888888888`.
- `float`: defines a "real" number.
  - `In []: p = 3.141592`.

- `complex`: defines a complex number.
  - `In []: c = 3+4j`.
  - `In []: c = complex(3, 4)`.

```
[ ]:
```

## 15 Values for Boolean operators

- `In []: t = True`.
- `In []: F = not t`.
- `In []: F or t`.
- `Out []: True`.
- `In []: F and t`.
- `Out []: False`.

```
[1]: t = True
     F = not t
     F or t
```

```
[1]: True
```

```
[ ]: F and t
```

## 16 Brackets ( ) with Boolean operators: different results according to where they are positioned

- `In []: a = False`
- `In []: b = True`
- `In []: c = True`
- `In []: (a and b) or c`
- `Out[]: True`
- `In []: a and (b or c)`
- `Out[]: False`

```
[8]: a = False
     b = True
     c = True
     (a and b) or c
```

```
[8]: False
```

[7]:
```
(a and b)
```

[7]:
```
False
```

[ ]:
```
a and (b or c)
```

## 17 Strings

- You can use any kind of quotes to define a string.
- ‘This is a string’
- ” Also this! ”
- “ ” “And this too!” “ ”
- ’ ’ ‘And yet another!” ’

[9]:
```
a  = ' This is a string '
print(a)
```

```
 This is a string
```

[ ]:
```
b = " Also this! "
print(b)
```

[ ]:
```
c = """ And this too! """
print(c)
```

[ ]:
```
d = ''' And yet another! '''
print(d)
```

## 18 Strings (2)

- We must distinguish between quotation marks that are used by Python to define a string variable and quotation marks that define a text or a text.
- ‘ “Go and do this!” Carlos said.’
- If there is an apostrophe in the text, use the definition with “ “.
- Triple quotes form “split” text into lines:
- `python     """ En un lugar de la Mancha,      de cuyo nombre no quiero acordarme,      no ha mucho tiempo que vivía un hidalgo....     """`

[2]:
```
a = ' "Go and do this! "Said Carlos.'
print(a)
```

```
 "Go and do this! "Said Carlos.
```

[4]:
```python
a = " l'apostrophe (en Francaise)"
print(a)
```

```
 l'apostrophe (en Francaise)
```

[5]:
```python
a = """ En un lugar de la Mancha,
  de cuyo nombre no quiero acordarme,
  no ha mucho tiempo que vivía un hidalgo....
    """
print(a)
```

```
 En un lugar de la Mancha,
  de cuyo nombre no quiero acordarme,
  no ha mucho tiempo que vivía un hidalgo…
```

## 19 Strings

- `In []: w = "ciao"`
- `In []: print(w[0], w[1], w[-1], w[1:3])`
- `In []: len(w)`
- `Out[]: 4`

[1]:
```python
w = "ciao"
print (w[0], w[1], w[-1], w[1:3])
```

```
c i o ia
```

[ ]:
```python
len(w)
```

## 20 Strings (4)

- Once a string has been assigned, it can no longer change.
- `In []: w[0] = 'C'`
- `python ---------------------------------------------- TypeError Traceback (most recent call last) <ipython console> in <module>() TypeError: 'str' object does not support item assignment`

[2]:
```python
w[0] = 'C'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-2-0076bfcda6b2> in <module>
----> 1 w[0] = 'C'
```

```
TypeError: 'str' object does not support item assignment
```

## 21 The type() function: discovering data types

- `In []: a = 1.0`
- `In []: type(a)`
- `Out[]: float`
- `In []: type(1)`
- `Out[]: int`
- `In []: type(1+1j)`
- `Out[]: complex`
- `In []: type('ciao')`
- `Out[]: str`

[3]:
```
a = 1.0
type(a)
```

[3]:
```
float
```

[4]:
```
type(1)
```

[4]:
```
int
```

[5]:
```
type(1+1j)
```

[5]:
```
complex
```

[6]:
```
type('ciao')
```

[6]:
```
str
```

## 22 Arithmetic operator (I)

- % is the modulus:
- `In []: 77 % 2`
- `Out[]: 1`
- `In []: 1886 % 12`
- `Out[]: 2`
- `In []: 1884 % 12`

- `Out[]: 0`
- `In []: 864675 % 10`
- `Out[]: 5`
- `In []: 31245 * 126789`
- `Out[]: 3961522305`
- `**` is the power operator:
- `In []: big = 12345678912345678 90 ** 3`
- `In []: verybig = big * big * big * big`

```
[7]: 77 % 2
```

```
[7]: 1
```

```
[8]: 1886 % 12
```

```
[8]: 2
```

```
[9]: 1884 % 12
```

```
[9]: 0
```

```
[10]: big = 1234567891234567890 ** 3
      print(big)
```

```
1881676377434183981909562699940347954480361860897069000
```

```
[11]: verybig = big * big * big * big
      print(verybig)
```

```
12536598903329366187366602453637834523513900340514681990694177938853001286963089
59751318632827038393929892364111034045644705822191276748064162089769506596140320
712943567700406694501848259452923449436695912100000000000000
```

## 23 Arithmetic operators (II)

- `In []: 18 / 3`
- `Out[]: 6`
- `In []: 18 / 3.0`
- `Out[]: 6.0`
- `In []: 18.0 / 3`
- `Out[]: 6`

- `In []: 18.0 / 9.5`
- `Out[]: 1.894736842105263`

## 24 Arithmetic operators (III): the 'floor' operator

- `In []: 18 // 4`
- `Out[]: 4`
- `In []: 18 // 4.0`
- `Out[]: 4.0`
- `In []: 18.0 // 4.0`
- `Out[]: 4.0`
- `In []: 18.0 // 9.6`
- `Out[]: 1.0`

```
[13]: 18.0 // 9.6
```

```
[13]: 1.0
```

## 25 Arithmetic operators (IV)

- Counters (used in *loop* clauses):
- `In []: a = 8000`
- `In []: a += 1`
- `In []: a`
- `Out[]: 8001`
- `In []: a -= 7`
- `In []: a`
- `Out[]: 7994`
- `In []: a *= 2`
- `In []: a`
- `Out[]: 15988`
- `In []: a /= 5`
- `In []: a`
- `Out[]: 3197.6`

[ ]:
```
a = 8000
a += 2
a += 4
a
```

## 26 String operators (I)

- `In []: s = 'En un lugar '`
- `In []: p = 'de la Mancha,    '`
- `In []: q = ' de cuyo nombre no quiero acordarme, '`
- `In []: s + p + q`
- `Out[]: 'En un lugar de la Mancha, de cuyo nombre no quiero acordarme,'`
- `In []: s * 4`
- `Out[]: 'En un lugar En un lugar En un lugar En un lugar '`

[3]:
```
s = 'En un lugar '
p = 'de la Mancha, '
q = 'de cuyo nombre no quiero acordarme,'
s + p + q
```

[3]:
```
'En un lugar de la Mancha, de cuyo nombre no quiero acordarme,'
```

[4]:
```
s = 'En un lugar '
s * 4
```

[4]:
```
'En un lugar En un lugar En un lugar En un lugar '
```

## 27 String operators (II): errors

- 
- `In []: s * s`
- – ---------------------------------------------

  ```
  TypeError  Traceback (most recent call last)
  <ipython console> in <module>()

  TypeError: can`t multiply sequence by
             non-int of type `str`
  ```

[6]:
```
a = 'En un lugar'
a.startswith('En')
```

[6]:
```
False
```

## 28 Methods applied to strings (I)

- `In []: a = 'En un lugar de la Mancha'`
- `In []: a.startswith('En')`
- `Out[]: True`
- `In []: a.startswith('hidalgo')`
- `Out[]: False`
- `In []: a.endswith('Mancha')`
- `Out[]: True`
- `In []: a.upper()`
- `Out[]: 'EN UN LUGAR DE LA MANCHA'`
- `In []: a.lower()`
- `Out[]: 'en un lugar de la mancha'`

## 29 Methods applied to strings (II)

- `In []: a = '  Hola todos  '`
- `In []: b = a.strip()`
- `In []: b`
- `Out[]: 'Hola todos'`
- `In []: b.index('od')`
- `Out[]: s`
- `In []: b.replace('Hola', 'Buenas noches')`
- `Out[]: 'Buenas noches todos'`

[10]:
```
a = '        Hola todos   '
e,f = "En un lugar", "de la Mancha"
f
```

[10]:
```
'de la Mancha'
```

[11]:
```
a.index('od')
```

[11]:
```
18
```

[15]:
```
a = "Hola todos"
a.replace('Hola', 'Buenas noches')
a
```

```
[15]: 'Hola todos'
```

## 30 Strings: split and join

- `In []: chars = 'a b c d'`
- `In []: chars.split()`
- `Out[]: ['a', 'b', 'c', 'd']`
- `In []: ' '.join(['a', 'b', 'c', 'd'])`
- `Out[]: 'a b c d'`
- `In []: alpha = ', '.join(['a', 'b', 'c'])`
- `In []: alpha`
- `Out[]: 'a, b, c'`
- `In []: alpha.split(', ')`
- `Out[]: ['a', 'b', 'c']`

```
[16]: alpha = ', '.join(['a', 'b', 'c'])
      alpha
```

```
[16]: 'a, b, c'
```

```
[17]: alpha.split()
```

```
[17]: ['a,', 'b,', 'c']
```

## 31 String formatting and visualization of numbers in strings

- `In []: x, y = 1, 1.234`
- `In []: x`
- `Out[]: 1`
- `In []: y`
- `Out[]: 1.234`
- `In []: 'x is %s, y is %s' %(x, y)`
- `Out[]: 'x is 1, y is 1.234'`
- You can also use `%d` or `%f` (see http://docs.python.org/library/stdtypes.html)

```
[18]: x, y = 1, 1.234
      'x is %s, y is %s' %(x, y)
```

```
[18]: 'x is 1, y is 1.234'
```

## 32 Relational and logic operators

- `In []: p, z, n = 1, 0, -1`
- `In []: p == z`
- `Out[]: False`
- `In []: p >= n`
- `Out[]: True`
- `In []: n < z < p`
- `Out[]: True`
- `In []: p + n != z`
- `Out[]: False`
- `n []: p + n == z`
- `Out[]: True`

## 33 The *assertion handling*

- `In []: p, z, n = 1, 0, -1`
- `In []: assert p != n`
- `In []: assert p == n`
- 
```
--------------------------------------------
AssertionError      Traceback (most recent call last)
----> 1 assert p == n

AssertionError:
```
- The error is not reported if the condition is true.
- The error is reported if the condition is false.

```
[13]: print(p)
      n
```

```
1
```

```
[13]: -1
```

```
[19]: p, z, n = 1, 0, -1
      assert p == n, "Error: condition is not verified"
      assert p != n
```

```
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
<ipython-input-19-8df85f50b0ea> in <module>
      1 p, z, n = 1, 0, -1
----> 2 assert p == n, "Error: condition is not verified"
      3 assert p != n
      4

AssertionError: Error: condition is not verified
```

[20]:
```
p, z, n = 1, 0, -1
assert p != n
```

[11]:
```
assert p == n
```

```
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
<ipython-input-11-9363016b14fc> in <module>
----> 1 assert p == n

AssertionError:
```

## 34 Examples of error handling

- `In []: assert p == n, "Error: condition is not verified"`
- `In []: assert p != n`
- `In []: assert p == n`

-
  ```
  ---------------------------------------------
  AssertionError      Traceback (most recent call last)
  ----> 1 assert p == n

  AssertionError: Error: condition is not verified
  ```
- It handles the error and provides an error message.

## 35 String containers

- `In []: fruta = 'manzana, banana, pera'`
- `In []: 'manzana' in fruta`
- `Out[]: True`
- `In []: 'patata' in fruta`

- `Out[]: False`

## 36 Keyboard IO

- “ input () ’ ’ is the command to take keyboard input.
- “ In []: a = input (‘Please enter a value:’) ’ ’ “Please enter a value: 10”
- `input () '' generates strings; so to use the entered value as input you have to transform it into an integer with` int () “.

[21]:
```
input('Escriba un valor: ')
```

```
Escriba un valor: 6
```

[21]:
```
'6'
```

## 37 The conditional *flow* `if…elif…else`

- `python    x = int(input("Introduce un número entero: ")) if x < 0:     print('Negativo') elif x == 0:     print('Cero') else:     print('Positivo')`

[24]:
```
x = int(input("Introduce un número entero: "))

if x < 0:
    print('Negativo')
elif x == 0:
    print('Cero')
else:
    print('Positivo')
```

```
Introduce un número entero: 2
Positivo
```

## 38 The `while` *loop* : example with the Fibonacci sequence

- `python     a, b = 0, 1 while b < 1000:     print(b, end=' ')     next = a + b     a = b     b = next`

[25]:
```
a, b = 0, 1
while b < 1000:
    print(b, end=' ')
    next = a + b
    a = b
    b = next
```

```
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
```

## 39 The `for` *loop* with the command `range`

Example: squares of the first 11 natural numbers:

- `python     for i in range(0, 11):         print(i, i * i)`

[27]:
```
for i in range(0, 11, 2):
        print(i, i * i)
```

```
0 0
2 4
4 16
6 36
8 64
10 100
```

## 40 The `break` clause: exiting from the `while` *loop*

Example: find the first number less than 100 of the Fibonacci series divisible by 4

- `python     a, b = 0, 1     while b < 100:         if b % 4 == 0:             print(b)             break         a, b = b, a + b`
- Beware of "indentation": it is important to define the `while` loop and the `if` condition.

[20]:
```
a, b = 0, 1
while b < 100:
    if b % 4 == 0:
        print(b)
        break
    a, b = b, a + b
```

```
8
```

## 41 The "pass clause": do nothing and move to the next iteration in the *loop*

- `python    for i in range(5):        if i % 2 ==  0:             pass     else:             print(i, 'dispar')     1 dispar     3 dispar`
- Beware of "indentation": it is important to define the `while` loop and the `if` condition.

[ ]:
```
for i in range(5):
    if i % 2 ==  0:
        pass
    else:
        print(i, 'dispar')
```

## 42 Data structures: *lists*

- Lists contain different types of data …
- … also the other lists that are therefore nested in another list ….
- Example:

```
# Esta es una lista:

In []: x = ['manzana', 1, 'banana', 2.5, 't-shirt', [1,2,3]]

In []: len(x)

Out []: 6
```

[18]:
```
x = ['manzana', 1, 'banana', 2.5, 't-shirt', [1,2,3]]
len(x)
len(x[5])
```

[18]:
```
3
```

[21]:
```
#Subsetting a list
x[5][2]
```

[21]:
```
3
```

## 43 Methods in lists: the "append" clause

```
In []: num = [9, 8, 2, 3, 7]

In []: num + [4, 5, 6]

Out []: [9, 8, 2, 3, 7, 4, 5, 6]

In []: num.append([4, 5, 6])

In []: num

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

[22]:
```
num = [9, 8, 2, 3, 7]
num + [4, 5, 6]
```

[22]:
```
[9, 8, 2, 3, 7, 4, 5, 6]
```

[23]:
```
num.append(4)
num
```

[23]:
```
[9, 8, 2, 3, 7, 4]
```

[26]:
```
num = [9, 8, 2, 3, 7]
num1 = num + [10,11,12]
```

```
num1.append([4, 5, 6])
num1
```

```
[26]: [9, 8, 2, 3, 7, 10, 11, 12, [4, 5, 6]]
```

## 44 List methods: `extend`, `reverse` and `remove`

- Lists contain different types of data …
- … also the other lists that are therefore nested in another list ….
- Example:

```
# This is a list:

In []: num = [9, 8, 2, 3, 7]

In []: num.extend([4, 5, 6])

In []: num

Out[]: [9, 8, 2, 3, 7, 4, 5, 6]

In []: num.reverse()

In []: num

Out[]: [6, 5, 4, 7, 3, 2, 8, 9]

In []: num.remove(6)

In []: num
```

```
[27]: num = [9, 8, 2, 3, 7]
      num.extend([4, 5, 6])
      num
```

```
[27]: [9, 8, 2, 3, 7, 4, 5, 6]
```

```
[17]: num.reverse()
      num
```

```
[17]: [6, 5, 4, 7, 3, 2, 8, 9]
```

```
[18]: num.remove(6)
      num
```

```
[18]: [5, 4, 7, 3, 2, 8, 9]
```

## 45 List methods: `count`, `index` and `pop`

```
In []: num = [9, 8, 2, 3, 1, 2, 3, 4]
In []: num.count(2) #How many times is number 2 in the list?

Out[]: 2

In []: num.index(2) #What is the index of the first number 2 on the list?

Out[]: 2

In []: num.pop(4) #Returns the element present in correspondence of the argument
and deletes it from the list

Out[]: 1

In []: num

[9, 8, 2, 3, 2, 3, 4]
```

[28]:
```
num = [9, 8, 2, 3, 1, 2, 3, 4]
num.count(2)
```

[28]:
```
2
```

[29]:
```
num.index(2)
```

[29]:
```
2
```

[30]:
```
num.pop(4)
```

[30]:
```
1
```

[32]:
```
num
```

[32]:
```
[9, 8, 2, 3, 2, 3, 4]
```

## 46 Removing elements from a list

- List elements can be removed based on their index or ….
- … based on their value.

" In []: del num [1] ''

" In []: num.remove (3) ''

- – When removing by value the first element in the list containing that value is removed *.

[ ]:
```
del num[1]
num.remove(3)
num
```

## 47 Ordering with `sort()`

```
In []: a = [5, 1, 6, 7, 7, 10]
```

```
In []: a.sort()
```

```
In []: a
```

- The method `sort` orders a list.
- If you may want a new sorted list you have to use `sorted`.
- Use `reverse=True` for an inverse order

```
In []: a = [5, 1, 6, 7, 7, 10]
```

```
In []: b = sorted(a, reverse=True)
```

```
In []: b
```

```
[23]: a = [5, 1, 6, 7, 7, 10]
      a.sort()
      a
```

```
[23]: [1, 5, 6, 7, 7, 10]
```

```
[24]: a = [5, 1, 6, 7, 7, 10]
      b = sorted(a, reverse=True)
      b
```

```
[24]: [10, 7, 7, 6, 5, 1]
```

```
[ ]: a
```

```
[25]: 110/30
```

```
[25]: 3.6666666666666665
```

## 48 The list container

```
In []:  num = [9, 8, 2, 3, 7]
```

```
In []: 4 in num
```

```
Out[]: False
```

```
In []: b = 8
```

```
In []: b in num
```

```
Out[]: True
```

```
In []: b not in num
```

```
Out[]: False
```

```
[26]: num = [9, 8, 2, 3, 7]
      4 in num
```

```
[26]: False
```

```
[27]: b = 8
      b in num
```

```
[27]: True
```

```
[28]: b not in num
```

```
[28]: False
```

## 49 Tuple: immutable sequences

- Tuples cannot be modified and does not allow for assignment.

```
In []: x = 1, 2, 3

In []: x

Out[]: (1, 2, 3)

In []: t = (1, 2, 3, 4, 5, 6, 7, 8)

In []: t[0] + t[3] + t[-1]

Out[]: 13

In []: t[4] = 5
```

```
---------------------------------------------------------
TypeError                 Traceback (most recent call last)
<ipython-input-36-fa7207573daf> in <module>
----> 1 t[4] = 5

TypeError: 'tuple' object does not support item assignment
```

```
[ ]: x = 1, 2, 3
     x
```

```
[ ]: t = (1, 2, 3, 4, 5, 6, 7, 8)
     t[0] + t[3] + t[-1]
```

```
[ ]: t[4] = 5
```

```
[22]: num_prime = [2, 3, 5, 7, 11, 13, 17]
      num_prime[2:6]
      num_prime[2:-1]
      numeros = list(range(14))
```

```
numeros
```

```
[22]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
```

```
[25]: num_prime[2:-2]
```

```
[25]: [5, 7, 11]
```

```
[27]: # the last argument (2) stands for the step
      numeros[1:10:2]
```

```
[27]: [1, 3, 5, 7, 9]
```

```
[14]: numeros[:10]
```

```
[14]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
[15]: numeros[::2]
```

```
[15]: [0, 2, 4, 6, 8, 10, 12]
```

```
[16]: numeros[:: -1]
```

```
[16]: [13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
```

```
[17]: a , b = 5, 7
```

```
[18]: a,b = b,a
```

```
[19]: a
```

```
[19]: 7
```

```
[20]: b
```

```
[20]: 5
```

```
[21]: x = 1, 2, 3
```

```
[22]: x
```

```
[22]: (1, 2, 3)
```

```
[33]: x[0]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
```

```
<ipython-input-33-2f755f117ac9> in <module>
----> 1 x[0]

TypeError: 'int' object is not subscriptable
```

```
[23]: y = list(x)
```

```
[23]: [1, 2, 3]
```

```
[24]: x
```

```
[24]: (1, 2, 3)
```

```
[25]: y = list(x)
```

```
[26]: y
```

```
[26]: [1, 2, 3]
```

```
[27]: tuple(y)
```

```
[27]: (1, 2, 3)
```

```
[28]: tuple('buenas dias')
```

```
[28]: ('b', 'u', 'e', 'n', 'a', 's', ' ', 'd', 'i', 'a', 's')
```

```
[29]: #Block indexing in Python
      #Syntax is the following:
      #sequence[intitial point: final point: step]
      num_primos= [1,3,5,7,11,13,17,19,23]
```

```
[ ]: num_primos[2:6]
```

```
[ ]: num_primos[:4]
     numeros = list(range(1:14))
```

```
[ ]: stringa = ['hola','a todos']
     stringa[:2]
```

```
[ ]: numeros = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16, 18 ,20]
     numeros[0:16:2]
```

```
[ ]: numeros
```

```
[ ]: numeros[10:]
[ ]: numeros[::2]
```

```
[ ]: numeros[::-2]
```

```
[ ]: numeros[::-1]
```

```
[ ]: semana = 'lun mar mie jue vie sab dom'
```

```
[ ]: semana
```

```
[ ]: semana.split()
```

```
[ ]: semana.split()[2]
```

```
[ ]: stri = 'Domingo'
```

```
[ ]: if stri.lower()[:3] not in semana:
         print('Ingrese el día de la semana correctamente')
     else:
         print('Día de la semana correcto')
```

```
[39]: #DICTIONARIES: The starting point of data frames....
      d = {'png ': 'image file','txt' : 'text file','ipynb' : 'notebook file','java' :
       ↪ 'java file','pdf' : 'Acrobat file'
      }
```

```
[36]: d['txt']
```

```
[36]: 'text file'
```

```
[37]: 'png' in d
```

```
[37]: True
```

```
[38]: 'jpg' in d
```

```
[38]: False
```

```
[34]: list(d.keys())
```

```
[34]: ['png', 'txt', 'ipynb', 'java', 'pdf']
```

```
[35]: list(d.values())
```

```
[35]: ['image file', 'text file', 'notebook file', 'java file', 'Acrobat file']
```

```
[36]: d['bin'] = 'binary file'
```

```
[37]: d
```

```
[37]: {'png': 'image file',
       'txt': 'text file',
       'ipynb': 'notebook file',
       'java': 'java file',
       'pdf': 'Acrobat file',
       'bin': 'binary file'}
```

```
[42]: d['pdf'] = 'Portable Document Format'
```

```
[43]: d
```

```
[43]: {'png ': 'image file',
       'txt': 'text file',
       'ipynb': 'notebook file',
       'java': 'java file',
       'pdf': 'Portable Document Format'}
```

```
[44]: del d['java']
```

```
[45]: d
```

```
[45]: {'png ': 'image file',
       'txt': 'text file',
       'ipynb': 'notebook file',
       'pdf': 'Portable Document Format'}
```

```
[ ]: d.del('bin')
```

```
[ ]: d.pop('bin')
```

```
[ ]: d
```

```
[ ]: d.get('txt')
```

```
[41]: d.update({'png': 'image file', 'exe': 'executable file'})
```

```
[42]: d
```

```
[42]: {'png': 'image file',
       'txt': 'text file',
       'ipynb': 'notebook file',
       'java': 'java file',
       'pdf': 'Portable Document Format',
       'exe': 'executable file'}
```

```
[43]: 'bin' in d
```

```
[43]: False
```

```
[44]: 'exe' in d
```

```
[44]: True
```

```
[ ]: 'executable file' in d
```

```
[ ]: d1 = dict(png='image file', txt='text file')
```

```
[ ]: d1
```

```
[ ]: a_list=[1,2,3,4,5,6,7,8]
```

```
[ ]: a_juntos = set(a_list)
```

```
[ ]: a_juntos
```

```
[45]: # (Mathematical) sets
      Juntos1 = set([1,2,3,5,8])
      Juntos2 = set([2,3,5,7])
```

```
[46]: Juntos1 | Juntos2
```

```
[46]: {1, 2, 3, 5, 7, 8}
```

```
[47]: Juntos1 & Juntos2
```

```
[47]: {2, 3, 5}
```

```
[48]: Juntos1 - Juntos2
```

```
[48]: {1, 8}
```

```
[50]: Juntos1 ^ Juntos2
```

```
[50]: {1, 7, 8}
```

```
[55]: Juntos3 = set([1,2,3,4,5,6,7,8,9,10,11])
```

```
[56]: Juntos3 < Juntos1
```

```
[56]: False
```

```
[57]: Juntos1 < Juntos1
```

```
[57]: False
```

```
[58]: Juntos1 <= Juntos1
```

```
[58]: True
```

```
[59]: 1 in Juntos1
```

```
[59]: True
```

```
[60]: for i in Juntos1:
          print(i)
```

```
1
2
3
5
8
```

```
[61]: #Functions
      def fun(x):
          return (x*x)
```

```
[62]: fun(4)
```

```
[62]: 16
```

```
[63]: def Cervantes():
          print('En un lugar de la Mancha')
```

```
[64]: Cervantes()
```

```
En un lugar de la Mancha
```

```
[ ]: def promedio(a,b):
         return (a+b)/2
```

```
[ ]:  promedio(3,5)
```

```
[ ]: valores = [27,28,30,30,18,23,29,30]
     longitud = len(valori)
```

```
[ ]: abs(-5)
```

```
[65]: documento = open('DonQuixote.txt', encoding = 'mac roman', mode = 'r')
```

```
[66]: leggi = documento.read()
```

```
[67]: leggi
```

```
[67]: 'En un lugar de la Mancha, \nde cuyo nombre no quiero acordarme, \nno ha mucho
      tiempo que vivía un hidalgo de los de lanza en astillero, \nadarga antigua,
      \nrocín flaco y galgo corredor'
```

```
[68]: leggi_list = leggi.splitlines()
```

```
[69]: leggi_list
```

```
[69]: ['En un lugar de la Mancha, ',
       'de cuyo nombre no quiero acordarme, ',
       'no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero, ',
       'adarga antigua, ',
       'rocín flaco y galgo corredor']
```

```
[70]: type(leggi_list)
```

```
[70]: list
```

```
[71]: documento.close()
```

```
[1]: for riga in open('DonQuixote.txt', encoding = 'mac roman', mode = 'r'):
         print(riga)
```

```
En un lugar de la Mancha,

de cuyo nombre no quiero acordarme,

no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero,

adarga antigua,

rocín flaco y galgo corredor
```

## 50 High-level introduction to pandas

For a more complete introduction to pandas, see `https://pandas.pydata.org/`.

```
[46]: pip install pandas
```

```
Requirement already satisfied: pandas in /opt/anaconda3/lib/python3.8/site-
packages (1.3.4)
Requirement already satisfied: pytz>=2017.3 in
/opt/anaconda3/lib/python3.8/site-packages (from pandas) (2021.1)
Requirement already satisfied: python-dateutil>=2.7.3 in
/opt/anaconda3/lib/python3.8/site-packages (from pandas) (2.8.1)
```

```
Requirement already satisfied: numpy>=1.17.3 in
/opt/anaconda3/lib/python3.8/site-packages (from pandas) (1.22.1)
Requirement already satisfied: six>=1.5 in /opt/anaconda3/lib/python3.8/site-
packages (from python-dateutil>=2.7.3->pandas) (1.15.0)
WARNING: You are using pip version 21.2.2; however, version 22.1.2 is
available.
You should consider upgrading via the '/opt/anaconda3/bin/python -m pip install
--upgrade pip' command.
Note: you may need to restart the kernel to use updated packages.
```

[34]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[132]:
```python
#This is to let you have larger fonts...
from IPython.core.display import HTML
HTML("""
<style>

div.cell { /* Tunes the space between cells */
margin-top:1em;
margin-bottom:1em;
}

div.text_cell_render h1 { /* Main titles bigger, centered */
font-size: 2.2em;
line-height:1.4em;
text-align:center;
}

div.text_cell_render h2 { /*  Parts names nearer from text */
margin-bottom: -0.4em;
}


div.text_cell_render { /* Customize text cells */
font-family: 'Times New Roman';
font-size:1.5em;
line-height:1.4em;
padding-left:3em;
padding-right:3em;
}
</style>
""")
```

[132]:
```
<IPython.core.display.HTML object>
```

```
[ ]:
```