



Python: **Basics (in Python)**

AAA-Python Edition



Plan

- 1- Basic Operations and variables
- 2- Basic Types
- 3- Functions and white space formatting
- 4- Modules and Libraries
- 5- Examples of some Libraries
- 6- Installing Libraries in Google Colab



1- Basic Operations And variables

- To perform simple operations, you just have to type them:

```
[6] 1 | 11 + 8
```

```
↳ 19
```

```
[8] 1 | 25 / 2
```

```
↳ 12.5
```

Integer division/floored quotient

```
[4] 1 | 25 // 2
```

```
↳ 12
```

Modulus (remainder)

```
[3] 1 | 10 % 4
```

```
↳ 2
```

Exponent

```
[13] 1 | 2 ** 3
```

```
↳ 8
```



1- Basic Operations And variables

- Other type of operations:

```
[24] 1 | 5 > 3
```

☐ True

If 5 wasn't greater than 3, it would returned False

```
[25] 1 | 5 != 3
```

☐ True

Is 5 different from 3

```
[29] 1 | 5 == 3
```

☐ False

Is 5 equal to 3

The only case this expression is evaluated to True, it's when the two operands are evaluated to True

```
[30] 1 | not (5==3)
```

☐ True

```
[31] 1 | (5>3) and (2 >3)
```

☐ False

```
[32] 1 | (5>3) or (2 >3)
```

☐ True

The only case this expression is evaluated to False, it's when the two operands are evaluated to False



1- Basic Operations And variables

To evaluate an expression with multiple operators, the “**precedence**” rule apply

Expression between parentheses is evaluated first: $(-1+1)=0$

Then the exponentiation is evaluated: $6**0=1$

Then the multiplication is evaluated: $3*1=3$

Then the addition is evaluated: $5+3=8$

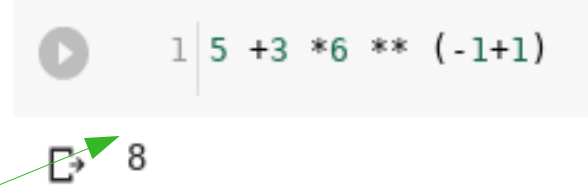


Table of precedence of some operators (increasing order)

Operators in the same box
have same precedence

Operators in the same box group from left to right
(except for exponentiation $**$). For example, to evaluate $5 / 4 * 2$:

We start by: $5 / 4 = 1.25$ (the most left operator)

Then: $1.25 * 2 = 2.5$ (we continue with the following one)

Highest precedence

or
and
not
< , <= , > , >= , != , ==
+, -
*, / , // , %
**
()

The full table can be seen at:

<https://docs.python.org/3.6/reference/expressions.html#operator-precedence>

[By Amina Delali]



1- Basic Operations And variables

- We can store values of expressions in “**variables**” with the “**assignment**” statement:

```
[ ] 1 a = 3
     2 b = 5
     3 c = a - b
     4 c
[-2]
```

Assignment statement

The value of “a” is 3,
and the value of “b” is 5

Variable name

- Variable names have some mandatory characteristics
 - Composed of **1 word**
 - Composed **only** by: **letters**, **number** or the underscore character (**_**)
 - **Can not start** with a **number**



1- Basic Operations And variables

- We can assign **one value** to **multiple variables**

```
[34]  1 | g = h = j = 5.31  
      2 |  
      3 | h
```

```
↳ 5.31
```

- We can assign **multiple values** to **multiple variables**:

```
[35]  1 | str1 , str2 , num1 = "Hello" , "World!" , -3  
      2 | str2
```

```
↳ 'World!'
```



2- Basic Types

-

Numbers

- A number can be:

- **Integer**

5
-3
0
1000

- **Float**

7.8
-3.156
0.0

- **Complex**

5.3+2j
10+1j

Real part

Imaginary part

```
[37] 1 | a = 4.3 + 5j  
      2 | a
```

```
↳ (4.3+5j)
```

```
[38] 1 | a.real
```

```
↳ 4.3
```

```
[39] 1 | a.imag
```

```
↳ 5.0
```




2- Basic Types

Strings

- String are text values written between **quotes**:

```
1 S1='Simple Quoted String'  
2 S2="Double Quoted String"  
3 S3='''Triple Quoted String with a simple quote.  
4 Can be written with double quotes.  
5 Contatins multiple lines'''  
6 S1
```

↳ 'Simple Quoted String'

```
[54] 1 S2
```

↳ 'Double Quoted String'

```
[58] 1 S3
```

↳ 'Triple Quoted String with a simple quote.\nCan be written with double quotes.\nContatins multiple lines'

```
[59] 1 print(S3)
```

↳ Triple Quoted String with a simple quote.
Can be written with double quotes.
Contatins multiple lines



Strings

- With Strings, we can perform **Concatenation** and **Replication** operations:

```
[62] 1 | S1 +" and a " + S2
```

↳ 'Simple Quoted String and a Double Quoted String'

```
[63] 1 | S2 * 2
```

↳ 'Double Quoted StringDouble Quoted String'



Boolean

- They have only two values: **True** and **False**
- In a numeric context: **True** behaves like **1** and **False** like **0**

```
[66] 1 | True + 3
```

```
↳ 4
```

```
[65] 1 | False + 3
```

```
↳ 3
```

- The Boolean operators are : **and, or, not**



3- Functions and white space formatting

Functions

- Functions are a “**reusable**” block of code.
- They can be “**built-in**” functions: already defined
- They can be also “**user-defined**”: you can define your own functions.
- Example of **built-in** functions:

Number of character of string S1

```
[68] 1 print(s1, "A simple string")
```

```
↳ Simple Quoted String A simple string
```

```
[74] 1 len(s1)
```

```
↳ 20
```



3- Functions and white space formatting

Functions

```
[82] 1 j= input("Give je value of j:\n")  
      2 print ("j=",j)  
      3
```

```
☞ Give je value of j:  
12  
j= 12
```

New line character

```
[80] 1 float(j)
```

```
☞ 12.0
```

Convert j into a float



3- Functions and white space formatting

White space formatting

- Python uses **indentation** to define blocks of code
- Blocks begin when the indentation **increases**
- Blocks end when the indentation **decreases**
- Whitespace is **ignored** inside parentheses and brackets

User defined functions

The block of code is marked by a colon(:) and its indentation (the space before print)

```
[88] 1 def function_name(param):  
      2     print("This is the parameter of the function: "+ param)  
      3 function_name("here")
```

☞ This is the parameter of the function: here

Calling the function
(decreasing the indentation to terminate
The function definition block)



3- Functions and white space formatting

Return statement

- A function can **return** a value using the keyword “**return**”

```
[95] 1 def add(a, b):  
      2     c= a+b  
      3     return c  
      4  
      5 print(add(5,3))
```

The function arguments

The function returns the value of a+b

8

Keyword and default arguments

- In a function **call**, we can **identify** the arguments by their **name**.
- In a function **definition**, the arguments can have a **default value**
→ they will be **optional**



3- Functions and white space formatting

- [97]

```
1 def printAnyway(a,b,toprint="No Given third argument"):
2     d=a/b
3     print("The result=",d)
4     print(toprint)
5     return d
6 printAnyway(b=2,a=4)
7
```

☞ The result= 2.0
No Given third argument
2.0

The default value of a third argument
So the argument is optional

The order of the arguments a and b
doesn't matter, since they are identified
by their names



4- Modules and Libraries

Module and Library

- A module is a program that contains a related **group of functions** that can be **embedded** in your programs
- To use the functions module you have to use the “**import**” statement.
- Other statement with **import** like “**from**” and “**as**” can be used
- A set of modules define a **Library**
- Python comes with a **library** called the **standard library**
- To use an other library modules, you have to **install** the corresponding library : a **third-party library**



4- Modules and Libraries

```
[100] 1 import random  
      2 print(random.randint(1,100))
```

☞ 93

Function “randint” from
module random

```
[101] 1 from random import randint  
      2 print(randint(1,100))
```

☞ 22

Only “randint” was
imported

```
[102] 1 from random import randint as ri  
      2 print(ri(1,100))
```

☞ 14

The name of “randint” was replaced by “ri”



5- Examples of some Libraries

Third-party libraries

- **Numpy:** is the fundamental package for scientific computing with Python
- **Pandas:** is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.
- **Matplotlib:** is a Python 2D plotting library
- **Tensorflow:** An open source machine learning framework for everyone. It is a software library for high performance numerical computation.



5- Examples of some Libraries

Third-party libraries

- **Numpy:**

```
[5] import numpy as np
    # create an array with a range of integers from 0 to 5
    a = np.arange(6)
    print(a)

    # transform the array into a (2,3) dimension array
    a=a.reshape(2,3)
    print(a)
```

```
↳ [0 1 2 3 4 5]
   [[0 1 2]
    [3 4 5]]
```

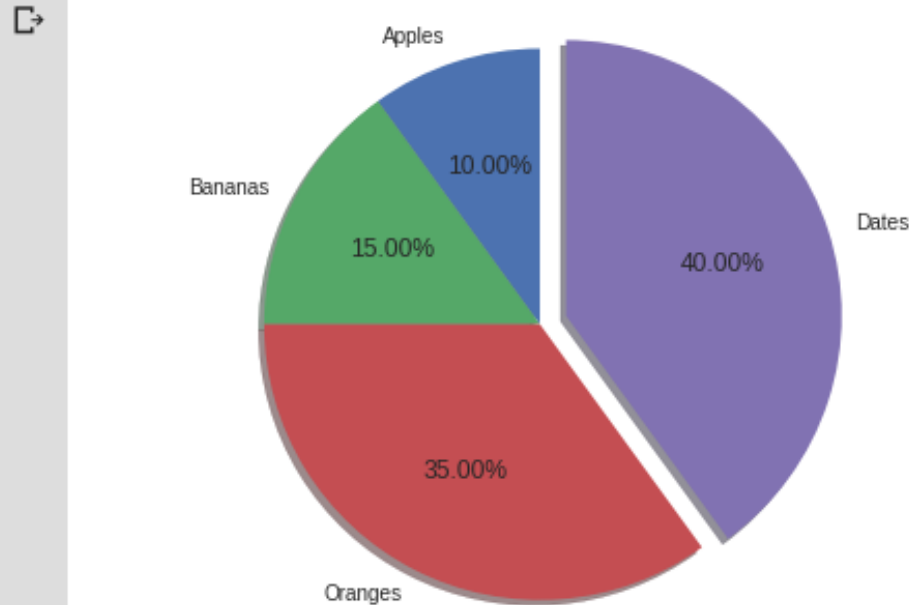


5- Examples of some Libraries

Third-party libraries

- **Matplotlib:**

```
import matplotlib.pyplot as plt
# Pie chart
labels = 'Apples', 'Bananas', 'Oranges', 'Dates' #names of the pie slices
sizes = [10, 15, 35, 40] # size of each slice
explode = (0,0,0,0.1) # explode the biggest slice (with 0.1 value)
fig, ax = plt.subplots()
ax.pie(sizes, explode=explode, labels=labels, autopct='%1.2f%%', shadow=True, startangle=90)
ax.axis('equal') # The pie has a circle form
plt.show()
```



[By Amina Delali]



6- Installing Libraries in Google Colab

! pip install

```
[ ] 1 # To determine which version you're using:  
2 !pip show tensorflow  
3  
4 # For the current version:  
5 !pip install --upgrade tensorflow  
6  
7 # For a specific version:  
8 !pip install tensorflow==1.2  
9  
10 # For the latest nightly build:  
11 !pip install tf-nightly
```

From:(https://colab.research.google.com/notebooks/snippets/importing_libraries.ipynb)

apt-get

```
[ ] 1 !apt-get install r-base
```

After !apt-get update



References

- Duchesnay Edouard and Löfstedt Tommy. Statistics and machine learning in python release 0.2. On-line at ftp: [//ftp.cea.fr/pub/unati/people/educhesnay/pystatml/M1_IMSV/StatisticsMachineLearningPythonDraft.pdf](ftp://ftp.cea.fr/pub/unati/people/educhesnay/pystatml/M1_IMSV/StatisticsMachineLearningPythonDraft.pdf). Accessed on 23-09-2018.
- Python Software Foundation. The python language reference. On-line at <https://docs.python.org/3.6/reference/index.html>. Accessed on 23-09-2018.
- Python Software Foundation. The python standard library. On-line at <https://docs.python.org/3.6/library/index.html>. Accessed on 23-09-2018.
- Joel Grus. Data science from scratch: first principles with python. O'Reilly Media, Inc, 2015.
- J. D. Hunter. Matplotlib: A 2d graphics environment. Computing In Science & Engineering, 9(3):90–95, 2007.
- NumPy. Numpy. On-line at <http://www.numpy.org/>. Accessed on 23-09-2018.



References

- pandas. pandas. On-line at <https://pandas.pydata.org/>. Accessed on 23-09-2018.
- Chandat Sunny. Learn Python in 24 Hours. CreateSpace Independent Publishing Platform, 2016.
- Al Sweigart. Automate the boring stuff with Python: practical programming for total beginners. No Starch Press, 2015.
- TensorFlow. About tensorflow. On-line at <https://www.tensorflow.org/>. Accessed on 23-09-2018.



Thank you!

FOR ALL YOUR TIME