Programming with Python

Hajer Mseddi

hajer.mseddi@datavora.com

Overview

Python is :

- Interpreted
- Object-Oriented
- Open source
- Easy-to-learn
- Easy-to-read
- Easy-to-maintain

History

- Developed by Guido van Rossum
- Python 1.0 : released in November 1994.
- Python 2.0 : released in 2000.
- Python 2.7.11 : latest edition of Python 2.
- Python 3.0 : released in 2008.
- Python 3.6 : latest stable version of Python 3.
- Python 3.7 : pre-release version
- Python 3.8 : in development

Install

- Windows : https://www.python.org/downloads/

  Add Python.exe to path

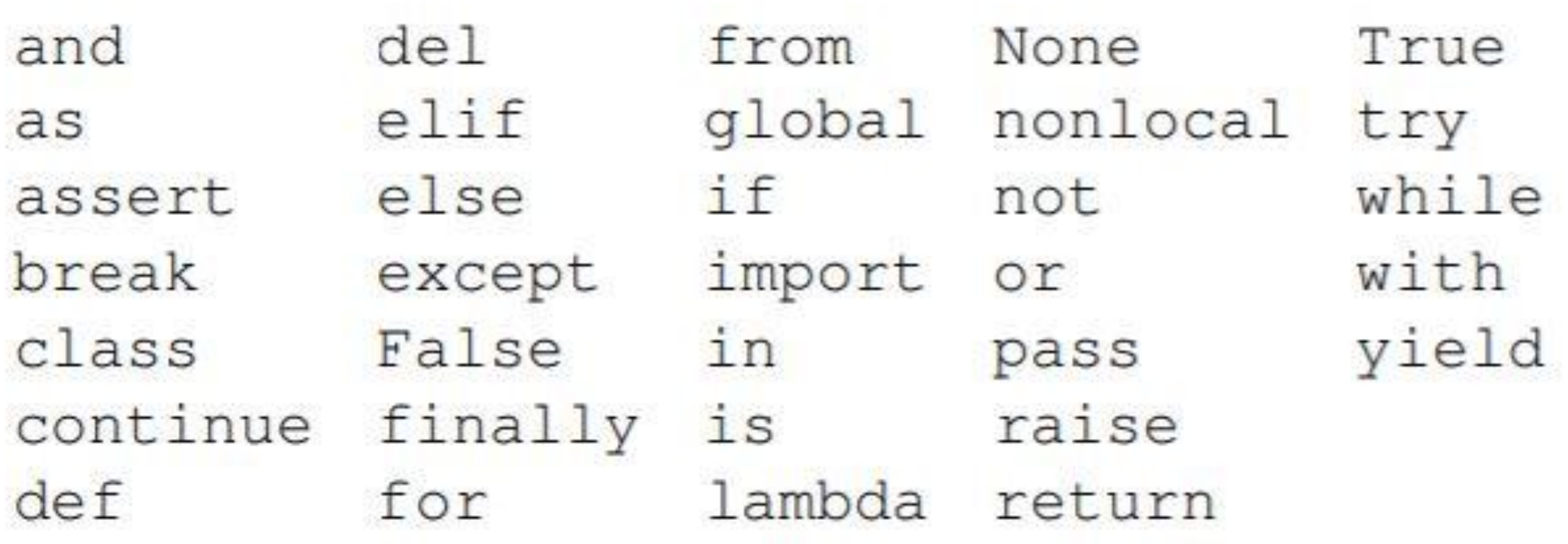
- Linux : ubuntu pre-installed

  Or $ sudo apt-get install python3-minimal

Naming Convention

- UPPERCASE or UPPER_CASE for constants
- TitleCase for classes
- camelCase for fonctions, methods and graphic interfaces
- AnExceptionError for exceptions
- amodule_m for modules
- lowercase ou lower_case for other identifiers

Python keywords: 33 reserved words



and	del	from	None	True
as	elif	global	nonlocal	try
assert	else	if	not	while
break	except	import	or	with
class	False	in	pass	yield
continue	finally	is	raise	
def	for	lambda	return	

Lines and Indentation

- No braces { }
- No semicolons ;
- One line = one statement
- One block = all statements must be **indented** the same amount

Multi-Line Statements

-
  ```
  total = item_one + \
          item_two + \
          item_three
  ```

Assigning Values to Variables

No need for explicit declaration :

- quantity = 10   # An integer assignment
- price = 9.2   # A floating point
- category = "Book"   # A string

Data types

- Numbers :
  - Int : n = 10
  - Float: n = 9.1
  - Complex: n = 3.14j
- String : s = 'Hello World!'
- Boolean : bf = False, bt = True
- List : abc = [‘a’, ‘b’, ‘c’, 1, 2, 3]
- Tuple : a read-only list : abc = (‘a’, ‘b’, ‘c’, 1, 2, 3)
- Dictionary : mydict = {‘name’: ‘Ahmed’, ‘age’: 10}

Basic Operators

- Arithmetic Operators : + - * / // % **
    - 9 / 2 = 4.5
    - 9 // 2 = 4
    - 9 % 2 = 1
    - 3 ** 2 = 9
- Comparison Operators : == != > < >= <=
- Assignment Operators : += -= *= /= ...
- Logical Operators : and or not
- Membership Operators : in not in
- Identity Operators: is not is

Strings

- s = ‘Hello World!’
  - s[0] → ‘H’
  - s[2:5] → ‘llo’
  - s[:4] → ‘Hell’
  - s[4:] → ’o World!’
- ‘H’ in s → True
- ‘H’ not in s → False
- ‘h’ in s → False
- len(s) → 12

- ‘Hello’ + ‘World’ → ‘HelloWorld’
- ‘Hello’ * 2 → ‘HelloHello’
- ‘Hello’.upper() → ‘HELLO’
- ‘Hello’.lower() → ‘hello’
- ‘hello world’.title() → ‘Hello World’
- ‘hello’.replace(‘o’, ‘a’) → ‘hella’
- ‘hello world’.split() → [‘hello’, ‘world’]
- ‘hello world’.split(‘o’) → [‘hell’, ‘ w’, ‘rld’]
-

Lists

- mylist = [‘a’, ‘b’, ‘c’, 1, 2, 3]
- mylist[0] → ‘a’
- mylist[2] → [‘c’]
- mylist[-1] → 3
- mylist[2:4] → [‘c’, 1]
- len(mylist) → 6
- [1, 2, 3] + [4, 5, 6] → [1, 2, 3, 4, 5, 6]
- ['Hi!'] * 4 → ['Hi!', 'Hi!', 'Hi!', 'Hi!']
- 3 in [1, 2, 3] → True

Lists

- mylist.append(‘a’) → [‘a’, ‘b’, ‘c’, 1, 2, 3, ‘a’]
- mylist.count(‘a’) → 1
- mylist.extend([‘x’, ‘y’]) → [‘a’, ‘b’, ‘c’, 1, 2, 3, ‘a’, ‘x’, ‘y’]
- mylist.index(‘a’) → 0 : lowest index in the list
- mylist.insert(2, ‘Z’) → [‘a’, ‘b’, ‘Z’, ‘c’, 1, 2, 3, ‘a’, ‘x’, ‘y’]
- mylist.pop() : Removes and returns last object or obj from list

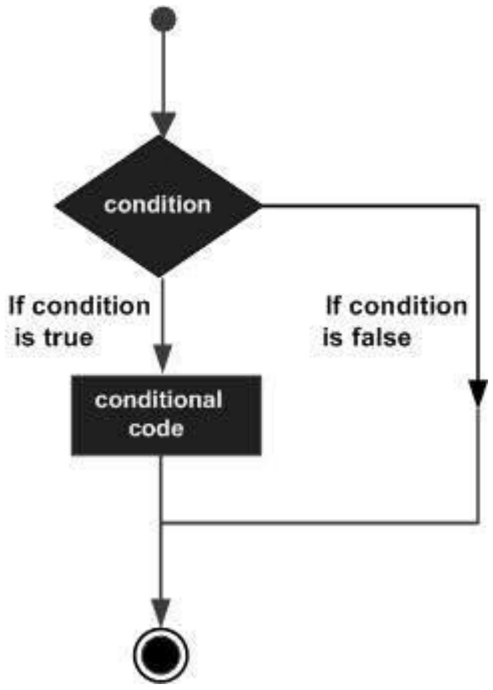
→ mylist.pop() → ‘y’ , mylist → [‘a’, ‘b’, ‘Z’, ‘c’, 1, 2, 3, ‘a’, ‘x’]

- mylist.remove(‘Z’) → [‘a’, ‘b’, ‘c’, 1, 2, 3, ‘a’, ‘x’]
- mylist.reverse() → [‘x’, ‘a’, 3, 2, 1, ‘c’, ‘b’, ‘a’]
- [‘c’, ‘b’, ‘d’, ‘a’].sort() → [‘a’, ‘b’, ‘c’, ‘d’]

Dictionaries

- mydict = {‘name’: ‘Iphone 7’, ‘price’: 2000, ‘category’: ‘Phone’}
- mydict['name'] → ‘Iphone 7’
- mydict[‘price’] → 2000
- mydict[‘color’] = ‘black’ → {‘name’: ‘Iphone 7’, ‘price’: 2000, ‘category’: ‘Phone’, ‘color’:’black’}
- mydict[‘color’] = ‘white’ → {‘name’: ‘Iphone 7’, ‘price’: 2000, ‘category’: ‘Phone’, ‘color’:’white’}
- del mydict[‘price’] → {‘name’: ‘Iphone 7’, ‘category’: ‘Phone’, ‘color’:’black’}

Decision Making



- **Non-zero** and **non-null** values are **True**
- Any **zero** or **null** values are **False**

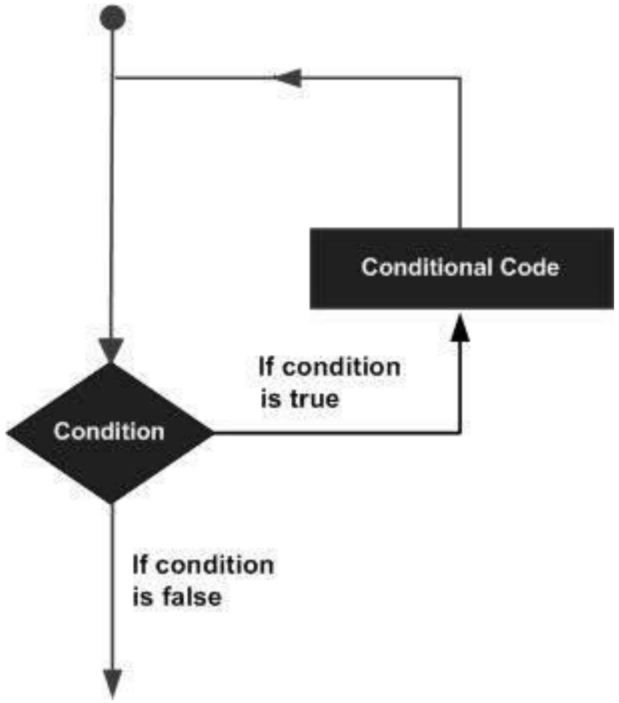
- if condition:

```
      statements
 else:
      statements
```

Decision Making

```
nb = 100

if ( nb == 100 ) :

    print ("Value of expression is 100")

else:

     print (“Value of expression is not 100)
```

Loops



- **While loop:** repeat while a condition is true
- **For loop:** loop in a list
- **Break** statement : stop the loop statements and execute to the following statement
- **Continue** statement : skip the remainder of the statements and retest condition
- **Pass** statement : statement is required syntactically but you do not want any command or code to execute

Loops

- **While**

```
count = 0
while (count < 5):
   print ('The count is:', count)
   count = count + 1
print ("END")
```

→

```
The count is: 0
The count is: 1
The count is: 2
The count is: 3
The count is: 4
END
```

Loops

- **For**

```
fruits = ['banana', 'apple',  'mango']
for fruit in fruits:
   print ('Current fruit :', fruit)
print ("END")
```

→

```
Current fruit : banana
Current fruit : apple
Current fruit : mango
END
```

Functions

```
def functionName( parameters ):
   function_suite
   return [expression]
```

-
  ```
  def sum(a , b ):
      # Add both parameters
      s = a + b
      return s
  ```

- sum(2, 5) → 7

Functions

- Let’s sort this list : [125, 65, 78, 1, 24, 44]

→ [125, 65, 78, 1, 24, 44].sort()

- [125, 65, 78, 1, 24, 44].sort(reverse = True)

Thanks for your attention