Python For Data Analysis

Introduction to Python

Imen Ouled Dlala

imen.ouled_dlala@devinci.fr

September 8, 2022

General Plan

1. Introduction
2. Numpy library
3. Pandas library
4. Data analysis and visualization
   - Seaborn, Matplotlib, Bokeh
5. Webscrapping
6. Machine learning and Datasets
   - Scikit-learn
7. API Django / Flask

Overview

Python

Getting Started with Python

- Python is both an interpreted and a compiled language.
- Python variables are references to objects.
- Instructions are executed one after the other.

Indenting Code

- Code blocks are defined by their indentation (this allows to define the loops and functions) .
- No semicolons and braces (but we can put several expressions in a row separated by ";") .
- Loops and Conditional statements end with a colon " : ".

Python

Getting Started with Python

```
import math
n=5
for a in range(1,n):
  for b in range(a,n):
    c_square = a**2 + b**2
    c = int(math.sqrt(c_square))
    if ((c_square - c**2) == 0):
      print(a, b, c)
```

Python

Getting Started with Python

The scope of a variable in python is that part of the code where it is visible.

- Local scope
  - A variable created inside a function belongs to the local scope of that function, and can only be used inside that function.
  - The local variable can be accessed from a function within the function.
- Global scope
  - A variable declared outside of the function or in global scope.
  - We can read it anywhere in the program.
  - Global variables are available from within any scope, global and local.
- Built-in
  - The built-in scope has all the names that are loaded into python variable scope when we start the interpreter.
  - We never need to import any module to access.

Python

Getting Started with Python

Semantics

Everything is an object

The methods are called via dots [.]
The type of the object is stored in the object itself.

Variables

They are pointers to objects.
They can refer to different objects from one moment to another.
Objects are typed, but not the variables!

```
x = "hello world !"
x.capitalize()
```

Hello world !

Python

Conditional statements and loops

```
if age > 18:
    adult =True
else:
    adult =False
```

```
if age < 10:
    print("child")
elif age < 18
    print("teenager")
else:
    print("adult")
```

```
for age in range(9,19):
    if age < 10:
        print("child")
    elif age <:18
        print("teenager")
    else:
        print("adult")
```

Python

Conditional statements and loops

```
iteration = 0
while iteration < 10:
  iteration+= 1
   print (iteration)
else:
   print ("fin")
```

```
import random
max = 100
value = random.randint(0, max)
iteration= 0
while iteration < max:
   if iteration == value :
      break
   iteration= iteration+ 1
print ("the value of random was:", iteration)
```

Python

String

```
name = "Antoine"
message = "Hello {name}!"
message.format(**locals())
```

'Hello Antoine!'

```
for name in ["a", "b"]:
    print (message.format(**locals()))
```

Hello a !
Hello b !

```
"Hello %s!"%(name)
```

'Hello b!'

Python

String

```
value_float=2.786543
"The value is %.2f!" %(value_float)
```

'The value is 2.79!'

⇒ 2 digits after the comma

```
"The value is %10.2f!" %(value_float)
```

⇒ 10 (blank) columns

```
"The value is %010.2f!" %(value_float)
```

⇒ 10 columns filled with (zeros)

Python

List

```
lst_1 = [1, 4, 77, 43, 100]
lst_2 = ['Paris','Berlin','Amsterdam']
lst_3 =[lst_1,lst_2]
lst_3 =[ ]
```

lst_1 =

1	4	77	43	100
#0	#1	#2	#3	#4

#index

#Indexing

- lst_1 [#index] ← element
- lst_1 [0] ← first element
- lst_1 [-1] ← last element

Python

List

#Slicing
position = 0
step = 1

- lst_1 [postion : position+1 : step] [1]
- lst_1 [postion : position+2 : step] [1, 4]
- lst_1 [postion : position+4 : step+1] [1, 77]
- lst_1 [-1:] [100]
- lst_1 [-2:] [43, 100]
- lst_1 [1:3] [4, 77]

Python

List

```
if 'Barcelona' in lst_2:
    print("yes")
else:
    print("no")
```

Output:

no

```
for i in lst_2:
    print(i)
```

Paris
Berlin
Amesterdam

Python

List

```
for index, valeur in enumerate(lst_2):
    print(index,valeur)
```

```
0 Paris
1 Berlin
2 Amesterdam
```

```
for x,y in zip(lst_2,lst_1):
    print(x,y)
# enumeration stops when the shortest list ends
```

```
Paris 1
Berlin 4
Amesterdam 77
```

Python

List

Changing lists

append	lst_1.append(8)	[1, 4, 77, 43, 100, 8]
insert	position= 2 value= 5 lst_1.insert(2, 5)	[1, 4, 5, 77, 43, 100]
+	lst_1+lst_1	[1, 4, 77, 43, 100, 1, 4, 77, 43, 100]
extend	lst_1.extend(lst_1)	[1, 4, 77, 43, 100, 1, 4, 77, 43, 100]

Python

List

Changing lists

pop	position = 4 list_1.pop(position)	[1, 4, 77, 43]
remove	value = 77 lst_1.remove(value)	[1, 4, 43, 100]
sort	lst_1.sort()	[1, 4, 43, 77, 100]
bisect	import bisect bisect.bisect(lst_1, 5)	2 (index of the first element $>$ 5)

Python

List Comprehensions

```
list=[ ]
for i in range (10):
    list.append(i**2)
print(list)
```

```
import time
start=time.time()
list=[ ]
for i in range(10):
    list.append(i**2)
end=time.time()
print(end-start)
```

```
list_comp=[i**2 for i in range (10)]
print(list_comp)
```

```
import time
start=time.time()
list=[i**2 for i in range(10)]
end=time.time()
print(end-start)
```

Python

Dictionaries

```
dictionary_1 = {
"un":"one",
"deux":"two",
}
```

```
dictionary_2={
"girls":40,
"boys":30
}
```

- key, value
  - dictionary_1.values()
  - dictionary_1.keys()
- Retrieve a value
  - dictionary_1.get("cinq","dont exist")
  - dictionary_1["deux"]
- Test the existence of a key
  - "un" in dictionary_1
  - "two" in dictionary_1

Python

Dictionaries

Display dictionary keys

```
for i in dictionary_1:
    print(i)
```

Display dictionary values

```
for i in dictionary_1.values():
    print(i)
```

Display dictionary keys and values

```
for k,v in dictionary_1.items():
    print(k,v)
```

Python

Dictionaries

Changing dictionaries

Insertion

key= "trois"
value= "three"
dictionary_1[key] = value

{"un" : "one", "deux" : "two", "trois" : "three" }

Concatenation

dictionary_3={ "quatre": "four" }
dictionary_1.update(dictionary_3)

{"un" : "one", "deux" : "two", "trois" : "three" ,"quatre" :"four"}

Python

Dictionaries

Changing dictionaries

Deletion

del $\Longrightarrow$ del dictionary_1["quatre"] {"un" : "one", "deux" : "two", "trois" : "three"}

pop $\Longrightarrow$ dictionary_1.pop("trois") {"un" : "one", "deux" : "two"}

Sort

soted $\Longrightarrow$ sorted(dictionary_1) returns a sorted list of dictionary keys

Python

Dictionary Comprehension

```
age = [22, 1, 40, 33, 55, 10]
first_name = ["Antoine", "Sarra", "Luc", "Julien"]
```

```
dict_comp_1 = { k:v for k,v in enumerate (first_name)}
```

```
dict_comp_1 = { 0: "Antoine" , 1: "Sarra", 2: "Luc", 3:"Julien" }
```

```
dict_comp_2 = { name:age for name,age in zip ( first_name,age)}
```

```
dict_comp_2 = { "Antoine": 22, "Sarra": 1, "Luc": 40, "Julien": 33}
```

```
dict_comp_3 = { name :age for name,age in zip (first_name,age) if age>22 }
```

```
dict_comp_3 = { "Luc": 40, "Julien": 33 }
```

Python

Tuples ans Sets

Tuples

tuple_int = (1, 2, 3, 4)
lst = list(tuple_int)
tuple_int = tuple(lst)

Sets

set_int = {1, 2, 3, 4}

set_int.add(value)
set_int.remove(value)
set_int.discard(value)
set_int.clear()

set_int.issubset(another_set)
set_int.issuperset(another_set)

set_int | another_set union
set_int & another_set intersection
set_int - another_set difference(elements in the first set, not in the second)
set_int ∧ another_set symmetrical difference

Python

Functions

Definition

def noun (param1, parm2)

```
def square (x):
    return x*x
```

```
square(3)
```

Output : 9

```
path_to_square = square
```

```
def apply (function, value) :
    return function(value)
```

```
apply(path_to_square, 3)
```

Output : 9

Python

Functions

map/filter

map(function_to_apply, list_of_inputs): applies a function to all the items in an input_list.

```
def square (x):
    return x*x
```

```
list_int = [0, 4, -5, 7, -100]
list_square =map(square,list_integer)
list_square
```

Output: [0, 16, 25, 49, 1000]

Python

Functions

map/filter

filter(function_to_apply, list_of_inputs): creates a list of elements for which a function returns true.

```
def is_positive (x):
    return x > 0
```

```
ist_int = [0, 4, -5, 7, -100]
list_positive= filter(is_positive, list_int)
list_positive
```

Output: [4, 7]

Python

Functions

```
square = lambda x: x**2
print(square(3))
```

```
function =lambda x,y: x**2+y
print(function(3,2))
```

```
list_square = map(lambda x: x**2,list_int)
list_square
```

Output: [0, 16, 25, 49, 1000]

Python

Functions

```
list(map(lambda x: x > 0,list_int))
```

Output: [False, True, False, True, False]

```
list(filter(lambda x: x > 0,list_int))
```

Output: [4, 7]

Python

Functions

*args and **kwargs

cf: `http://deusyss.developpez.com/tutoriels/Python/args_kwargs/`

Definition

def noun (*args) List of parameters (variable length)
def noun (**kwargs) A dictionary as a parameter

A function as a parameter

```
def apply( function, liste) :
    for index, item in enumerate(liste): #enumerate returns the index and the value
        liste[index] = function(item) #evaluation of the function
```

```
list_int =[2,3]
apply(path_to_square, liste_int)
```

Python

Dates

The datetime module supplies classes for manipulating dates and times.
It offers three types of objects:

- date, time and datetime (contains a date and a time)

Import
Utilization

```
from datetime import datetime, date, time
dt = datetime(2011,10,29,20,30,21)
dt.day
    29
dt.minute
    30
dt.date()
    datetime.date(2011,1,0,29)
dt.time()
    datetime.time(20,30,21)
```

Python

Dates

Conversion	dt.strftime("%m/%d/%Y %H:%m/%d") '10/29/2011 20:10/29' datetime.strptime('20091031', '%Y/%m/%d') datetime.datetime(2009,10,31,0,0)
Replace	dt.replace(minute=0, second=0)
Timedelta	datetime.datetime(2011,10,29,20,0) dt2 = datetime.datetime(2011,10,29,20,30,21) delta = dt2 - dt dt + delta

Python

Dates

Dates – format iso
source : `https://docs.python.org/2/library/datetime.html`

Python

Files and Folders

File

```
path_file ="C:/.../file.csv"
```

Read

```
with open(path_file, 'r') as file:
  for line in file:
    print line
```

Add

```
with open(path_file, 'a') as file:
  file.write("hello world!")
```

Read

```
content = open(path_file).readlines()
```

Write many lines

```
open(path_file).writelines(content)
```

Python

Files and Folders

Folder

path_dir = "C:/folder/"

Import os

Verify the existence

os.path.exists(path_dir)

Create a folder

os.makedirs(path_dir)

Useful function

```
import os
if not colorredos. path. exists (path_dir):
  os. makedirs (path_dir)
  print " {0}created". format(path_dir)
```

Join a file path and a folder path

os.join(path_dir, path_file)

End

Good Lecture!