Python Intro

January 13, 2021

1 Python Intro

Notebook containing examples of the different tools Python programming language provides, following the tutorial of the web W3schools.com

1.0.1 Table of Contents

- While Loops
- For Loops
- Functions
- Lambda
- Classes and Objects
- Inheritance
- User Inputs
- Try Except

1.0.2 Python Syntax

[1]:
```
print ("Hello World!")
```

```
Hello World!
```

[2]:
```
## Indentation is important
if 5 > 2:
    print ("Five is greater than two!")
```

```
Five is greater than two!
```

- `help()` is very usefull command to progress in Python
- `Tab` is a very powerfull tool to autocomplete our code

1.0.3 Python Comments

[3]:
```
## This is a comment
```

[4]:
```
"""
This is a comment
written in
more that just one line
"""

x = str("If nothing is stated in the cell it will return the comment, which is␣
 ↪actually an string.")
print (x)
```

```
If nothing is stated in the cell it will return the comment, which is actually
an string.
```

1.0.4 Python Variables

[5]:
```
## Variable naming
myCamelCase = str("camel")
MyPascalCase = str("pascal")
my_snakeCase = str("snake")
```

[6]:
```
## Pyhton is an intelligent language
carname = "Volvo"
type (carname)
```

[6]:
```
str
```

[7]:
```
## Multiple naming
x , y = 5 , 10
z = x + y
print(z)
```

```
15
```

[8]:
```
## Variables named on the Main are global
x = "Global"

## Variables named in functions are local
def fun():
    x = "Local"

print (x)
```

```
Global
```

1.0.5 Python Data Types

- str `x = "Hello World"`
- int `x = 20`
- float `x = 20.5`
- complex `x = 1j`
- list `x = ["apple", "banana", "cherry"]`
- tuple `x = ("apple", "banana", "cherry")`
- range `x = range(6)`
- dict `x = {"name" : "John", "age" : 36}`
- set `x = {"apple", "banana", "cherry"}`
- frozenset `x = frozenset({"apple", "banana", "cherry"})`
- bool `x = True`

[9]:
```
## Check the data type
is_male = True
type (is_male)
```

[9]:
```
bool
```

1.0.6 Python Numbers

[10]:
```
## Convert to float
x = 10; x = float(x)
print (x , type(x))
```

```
10.0 <class 'float'>
```

[11]:
```
## Convert to int
x = 10.8; x = int(x)
print (x , type(x))
```

```
10 <class 'int'>
```

[12]:
```
## Convert to complex
x = 10.8; x = complex(x)
print (x , type(x))
```

```
(10.8+0j) <class 'complex'>
```

1.0.7 Python Strings

[13]:
```
## Str in different lines
a = """Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua."""
print(a)
```

```
Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua.
```

[14]:
```
## Strings are arrays
for x in "Luis Martin":
    print(x, end = "")
```

```
Luis Martin
```

[15]:
```
## Length of a string
a = "ItAcademy"
print (len(a))
```

```
9
```

[16]:
```
## Check String with bool
txt = "The best things in life are free!"
```

```
if "free" not in txt:
    print ("No, 'free' is not present.")
else:
    print ("Yes, 'free' is present.")
```

```
Yes, 'free' is present.
```

[17]:
```
## Indexing
txt = "The best things in life are free!"
print (txt[:3] + " " + txt[-5:])
```

```
The free!
```

[18]:
```
## Upper case
a = "Luis Martin"
print (a.upper())
```

```
LUIS MARTIN
```

[19]:
```
## Lower case
a = "Luis Martin"
print (a.lower())
```

```
luis martin
```

[20]:
```
## Replace string
a = "Hello, World!"
print (a.replace("l", "L"))
```

```
HeLLo, WorLd!
```

[21]:
```
## Split string
a = "Luis Martin"
print (a.split(" "))
```

```
['Luis', 'Martin']
```

1.0.8 Python Booleans

Booleans represent one of two values: `True` or `False`

[22]:
```
## Comparisson Operators
print (10 > 9)
print (10 == 9)
print (10 < 9)
```

```
True
False
False
```

[23]:
```
## Evaluate Values. Empty values are false
z = [3.2, 0, "Hola", "", None]
print ([bool(x) for x in z])
```

```
[True, False, True, False, False]
```

1.0.9 Python Operators

[24]:
```
## Modulus returns the reminder
x = 4 % 3
print (x)
```

```
1
```

[25]:
```
## Floor division strips the decimal part
x = 10 // 3
print (x)
```

```
3
```

[26]:
```
## Add Asignment works also with -, *, /, ...
x = 10; x += 1;
print (x)
```

```
11
```

[27]:
```
## Comparisson ==, !=, <, <=, >, >=
print (10 != 10)
```

```
False
```

[28]:
```
## Logical (and, or, not)
10 > 8 and 8 > 5
```

[28]:
```
True
```

1.0.10 Python Lists

- List is changeable, meaning that we can change, add, and remove items in a list after it has been created.
- Since lists are indexed, lists can have items with the same value.
- You cannot copy a `list` simply by typing `list2 = list1`, because `list2` will only be a reference to `list1`, and changes made in `list1` will automatically also be made in `list2`. There are ways to make a copy, one way is to use the built-in List method `copy()`.

[29]:
```
## Chek item on a List
thislist = ["apple", "banana", "cherry"]
if "apple" in thislist:
```

```
    print ("Yes, 'apple' is in the fruits list")
```

```
Yes, 'apple' is in the fruits list
```

[30]:
```
## Lists are changeable
thislist = ["apple", "banana", "cherry"]
thislist[1:2] = ["blackcurrant", "watermelon"]
print(thislist)
```

```
['apple', 'blackcurrant', 'watermelon', 'cherry']
```

[31]:
```
## Insert item in agiven position
Colors = ["Rosa", "Blau", "Groc"]
Colors.insert(2, "??????")
print(Colors)
```

```
['Rosa', 'Blau', '??????', 'Groc']
```

[32]:
```
## Append inserts at the end
Colors = ["Rosa", "Blau", "Groc"]
Colors.append("??????")
print(Colors)
```

```
['Rosa', 'Blau', 'Groc', '??????']
```

[33]:
```
## Extend insert items into the list, not necessarily from another list
Colors = list(("Rosa", "Blau", "Groc"))
Colors_tuple = ("Rosa", "Blau", "Groc")
Colors.extend(Colors_tuple)
print (Colors)
```

```
['Rosa', 'Blau', 'Groc', 'Rosa', 'Blau', 'Groc']
```

[34]:
```
## Remove an item by label
numbers = ["one", "two", "three", "four"]
numbers.remove("two")
print (numbers)
```

```
['one', 'three', 'four']
```

[35]:
```
## Remove an item by index
numbers = ["one", "two", "three", "four"]
numbers.pop(1) #removes last if not specified
del numbers[-1] #can del the whole list
print (numbers)
```

```
['one', 'three']
```

[36]:
```
## List from other list
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]
newlist = [x for x in fruits if "a" in x]
print(newlist)
```

```
['apple', 'banana', 'mango']
```

[37]:
```
## Sort List
names = ["Maria", "Joan", "Lidia", "Judit", "Pau"]
names.sort(reverse = False)
print (names)
```

```
['Joan', 'Judit', 'Lidia', 'Maria', 'Pau']
```

1.0.11 Python Tuples

- A tuple is a collection which is ordered and **unchangeable**.
- Tuple items are ordered, unchangeable, and allow duplicate values.

[38]:
```
## Change item converting into a List
x = ("apple", "banana", "cherry")
y = list(x); y[1] = "kiwi"
x = tuple(y)
print(x)
```

```
('apple', 'kiwi', 'cherry')
```

[39]:
```
## Delete the whole tuple is possible
thistuple = ("apple", "banana", "cherry")
del thistuple
print(thistuple) #this will raise an error because the tuple no longer exists
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-39-98be15cd4b09> in <module>
      2 thistuple = ("apple", "banana", "cherry")
      3 del thistuple
----> 4 print(thistuple) #this will raise an error because the tuple no longer␣
 ↪exists

NameError: name 'thistuple' is not defined
```

[40]:
```
## Unpack tupple
Colors = ("Red", "Blue", "Green", "Black")
Red, Blue, *Other = Colors
print(Other)
```

```
['Green', 'Black']
```

[41]:
```
## Count items
x = (1,2,3,4,1,3,1,5,1,6,7)
x.count(1)
```

[41]: 4

1.0.12 Python Sets

- Sets are a collection which is both **unordered** and **unindexed**.
- Sets are **unchangeable**.
- Sets cannot have two items with the same value. **Do not allow duplicates**.

[42]:
```
## Define Set
set1 = {"apple", "banana", "cherry"}
print(set1)
```

```
{'apple', 'banana', 'cherry'}
```

[43]:
```
## Add to a set
names = {"Maria", "Joan", "Lidia", "Judit", "Pau"}
names.add("?????")
print (names)
```

```
{'Judit', 'Pau', 'Lidia', 'Maria', 'Joan', '?????'}
```

[44]:
```
## Update from another iterables
names = {"Maria", "Joan", "Lidia", "Judit", "Pau"}
numbers = range(3)
names.update(numbers)
print (names)
```

```
{0, 1, 2, 'Judit', 'Pau', 'Lidia', 'Maria', 'Joan'}
```

[45]:
```
## Remove and Discard
names = {"Maria", "Joan", "Lidia", "Judit", "Pau"}
names.remove("Pau") #if Pau not in names it raise an Error
names.discard("Elisabeth") #Does not raise an Error
print (names)
```

```
{'Judit', 'Lidia', 'Maria', 'Joan'}
```

[46]:
```
## Clear the Set
names = {"Maria", "Joan", "Lidia", "Judit", "Pau"}
names.clear()
print (names)
```

```
set()
```

[47]:
```
## Keep the duplicates
names1 = {"Maria", "Joan", "Lidia", "Judit", "Pau"}
names2 = {"Maria", "Ana", "Enric", "Joan"}
names1.intersection_update(names2)
print (names1)
```

```
{'Maria', 'Joan'}
```

1.0.13 Python Dictionaries

- `Dictionaries` are used to store data values in key: value pairs.
- `Dictionaries` are a collection which is **unordered**, **changeable** and **does not allow duplicates**.

[48]:
```
## Define a Dictionary
dict1 = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(dict1)
```

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
```

[49]:
```
## Acces to Dict values through Key
dict1 = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(dict1["brand"])
```

```
Ford
```

[50]:
```
## Store into variables
dict1 = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
x = dict1.get("model")
print (x)
```

```
Mustang
```

[51]:
```
## Add new pairs
car = {
"brand": "Ford",
```

```
"model": "Mustang",
"year": 1964
}

x = car.keys() #get keys
print(x)

car["color"] = "white"
print(x)
```

```
dict_keys(['brand', 'model', 'year'])
dict_keys(['brand', 'model', 'year', 'color'])
```

[52]:
```
## Get values
car = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}
x = car.values()
print (x)
```

```
dict_values(['Ford', 'Mustang', 1964])
```

[53]:
```
## Update dictionary
car = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
car.update({"year": 2020})
print (car)
```

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 2020}
```

1.0.14 Python If ... Else

[54]:
```
## Exemple
a = 200
b = 33
if b > a:
    print("b is greater than a")
elif a == b:
    print("a and b are equal")
else:
    print("a is greater than b")
```

```
a is greater than b
```

[55]:
```
## Simple If in one line
a = 200
b = 33
if a > b: print("a is greater than b")
```

```
a is greater than b
```

[56]:
```
## Complex If in one line
a = 330
b = 330
print("A greater than B") if a > b else print("They are equal") if a == b else␣
 ↪print("B greater than A")
```

```
They are equal
```

1.0.15 Python While Loops

[57]:
```
## Executes while the statement holds True
i = 0
while i < 6:
    print(i)
    if i == 3:
        break #Stops the while at i==3
    i += 1
```

```
0
1
2
3
```

[58]:
```
## Continue jumps to the top
i = 0
while i < 6:
    i += 1
    if i == 3:
        continue #3 not printed
    print(str(i) + ", ", end="")
else:
    print ("End")
```

```
1, 2, 4, 5, 6, End
```

1.0.16 Python For Loops

[59]:
```
## Exemple: For runs among iterable objects
for x in range(0, 10, 2):
    if x == 2: continue
    print(x)
```

```
    if x == 6: break
```

```
0
4
6
```

[60]:
```
## Exemple: Lists and else statement
for x in ["Apple", "Microsoft", "Sony", "Google"]:
    print (x)
else: print ("Finished!")
```

```
Apple
Microsoft
Sony
Google
Finished!
```

1.0.17 Python Functions

[61]:
```
## Defining a function
def fun(name, lastname):
        print ("My name is " + name + " " + lastname)

fun("Luis", "Martin")
```

```
My name is Luis Martin
```

[62]:
```
## Unknown number of parameters
def fun(*Colors):
    print (Colors[0] + " and " + Colors[1] + " are my favorite colors!")
fun("Red", "Blue")
```

```
Red and Blue are my favorite colors!
```

[63]:
```
## Parameters using keys and default values
def fun(teacher = "Ana", **names):
    print (names["name_1"] + ", " + names["name_2"] + " and " + names["name_3"]␣
 ↪+ " are my best friends. "  +
          teacher + " is our teacher.")
fun(name_3 = "Joan", name_1 = "Arnau", name_2 = "Tomas")
fun("Julia", name_3 = "Joan", name_1 = "Arnau", name_2 = "Tomas")
```

```
Arnau, Tomas and Joan are my best friends. Ana is our teacher.
Arnau, Tomas and Joan are my best friends. Julia is our teacher.
```

1.0.18 Python Lambda

[64]:
```
## Exemple
x = lambda a, b : a * b
print (x(5, 6))
```

```
30
```

[65]:
```
## Inside other functions
def myfunc(n):
    return lambda a : a * n
mytripler = myfunc(3)
print(mytripler(11))
```

```
33
```

1.0.19 Python Classes and Objects

- Python is an object oriented programming language.
- Almost everything in Python is an object, with its properties and methods.
- A Class is like an object constructor, or a "blueprint" for creating objects.

[66]:
```
## Defining classes and assigning objects
class MyClass:
    x = 5
MyObject = MyClass()
print (MyObject.x)
```

```
5
```

[67]:
```
## __init__ function
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

Manager = Person("Alex", 45)

print (Manager.name)
print (Manager.age)
```

```
Alex
45
```

[68]:
```
## Methods inside classes
class Person:
    def __init__(self, name, age, position):
        self.name = name
```

```
        self.age = age
        self.position = position

    def Hello(self):
        print ("Hello my name is " + self.name + ". I'm the " + self.position +␣
↪".")

Manager = Person("John", 36, "Boss")
Manager.Hello()
```

```
Hello my name is John. I'm the Boss.
```

1.0.20 Python Inheritance

- Inheritance allows us to define a class that inherits all the methods and properties from another class.

[69]:
```
## Parent Class
class Person:
    def __init__(self, fname, lname):
        self.firstname = fname
        self.lastname = lname

    def printname(self):
        print(self.firstname, self.lastname)

Unknown = Person("John", "Doe")
Unknown.printname()

## Child Class
class Teacher(Person):
    def __init__(self, fname, lname, subject):
        Person.__init__(self, fname, lname)
        self.subject = subject

    def printname(self):
        print (self.firstname, self.lastname + ",", "Subject:", self.subject)

Math_Teacher = Teacher("Carles", "Cordon", "Maths")
Math_Teacher.printname()
```

```
John Doe
Carles Cordon, Subject: Maths
```

1.0.21 Python User Input

[70]:
```
## Input gives feedback from user
name = input ("Enter your name: ")
print("Hello " + name)
```

```
Enter your name: Luis
Hello Luis
```

1.0.22 Python Try Except

The `tr` block lets you test a block of code for errors.

The `excep` block lets you handle the error.

The `finall` block lets you execute code, regardless of the result of the try- and except blocks.

[71]:
```
## Exemple. Does not raise an error
try:
    print(xx)
except:
    print("xx is not defined")
```

```
xx is not defined
```

[72]:
```
## Look for given errors
try:
    print(xx)
except NameError:
    print("Variable xx is not defined")
except:
    print("Something else went wrong")
```

```
Variable xx is not defined
```

[73]:
```
## Finally executes regardles of the exceptions
try:
    print(xx)
except:
    print("Something went wrong")
finally:
    print("The 'try except' is finished")
```

```
Something went wrong
The 'try except' is finished
```

[74]:
```
## Raise your own exceptions
x = 3.32
if type(x) is not int:
    raise TypeError("Only integers allowed")
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-74-64e79bf93f0e> in <module>
      2 x = 3.32
      3 if type(x) is not int:
----> 4     raise TypeError("Only integers allowed")

TypeError: Only integers allowed
```