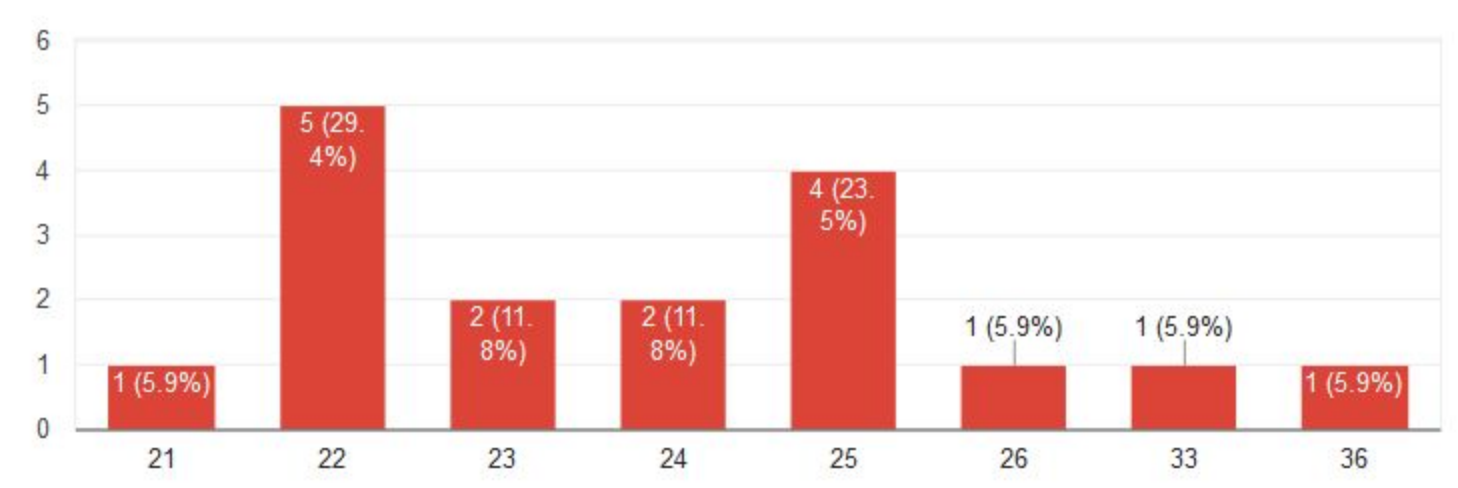
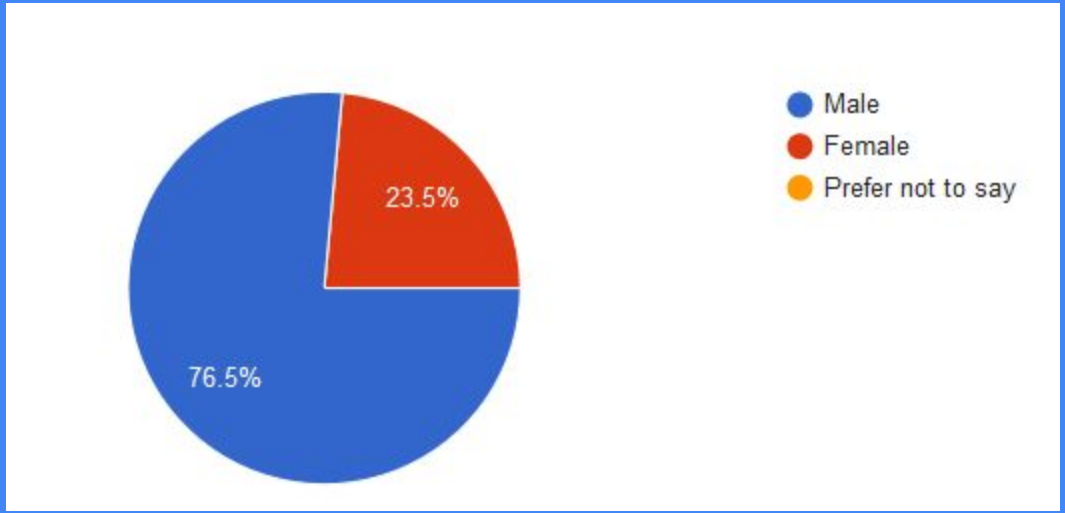
Lesson 2 - Meet your new best friend

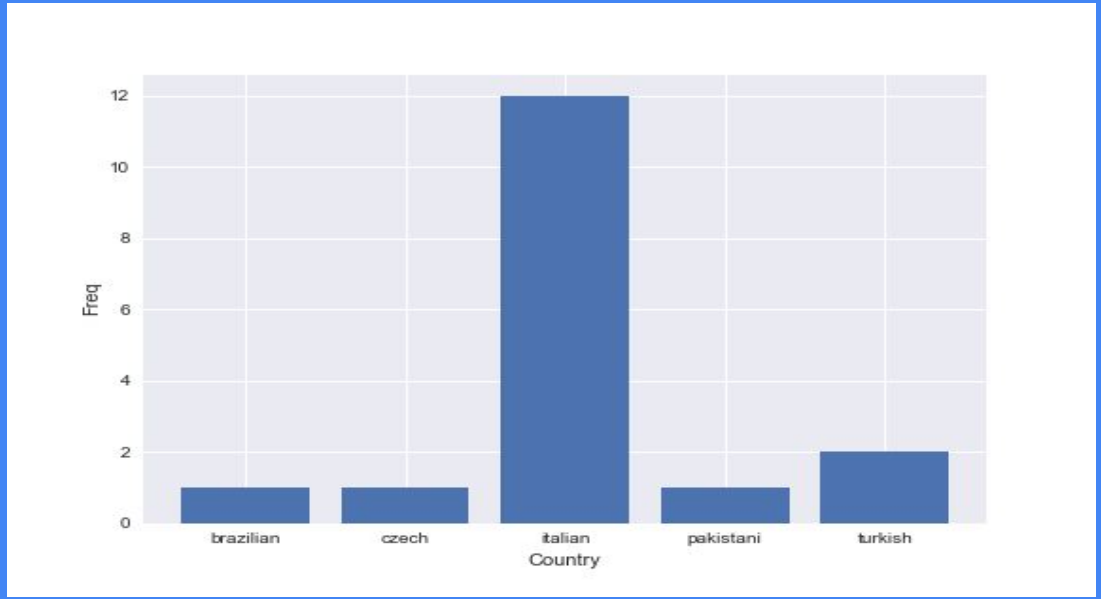
Who are you?



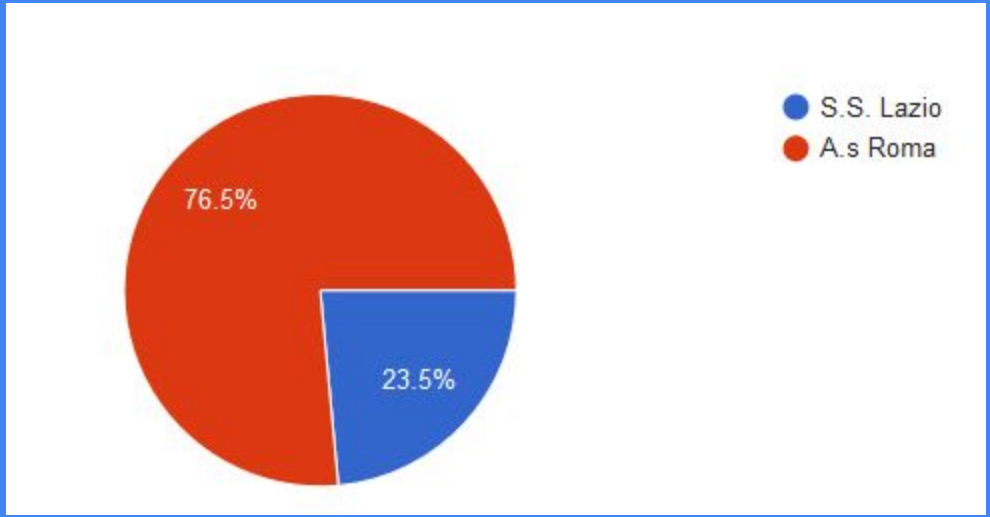
How old are you?



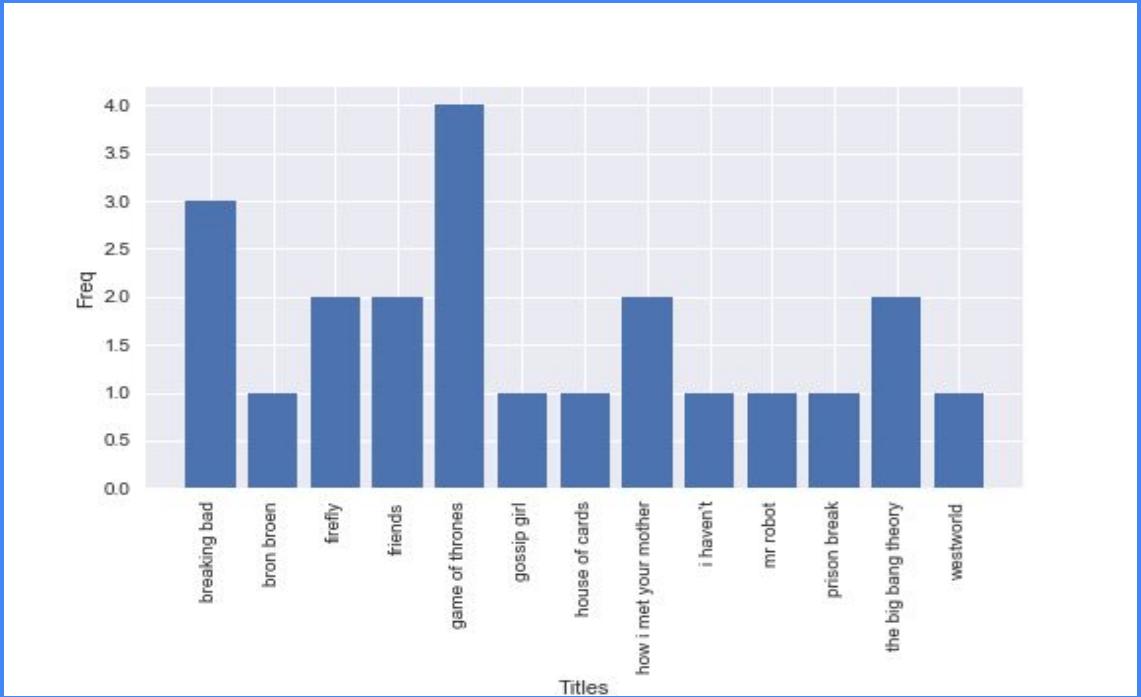
The pink quota is growing fast!



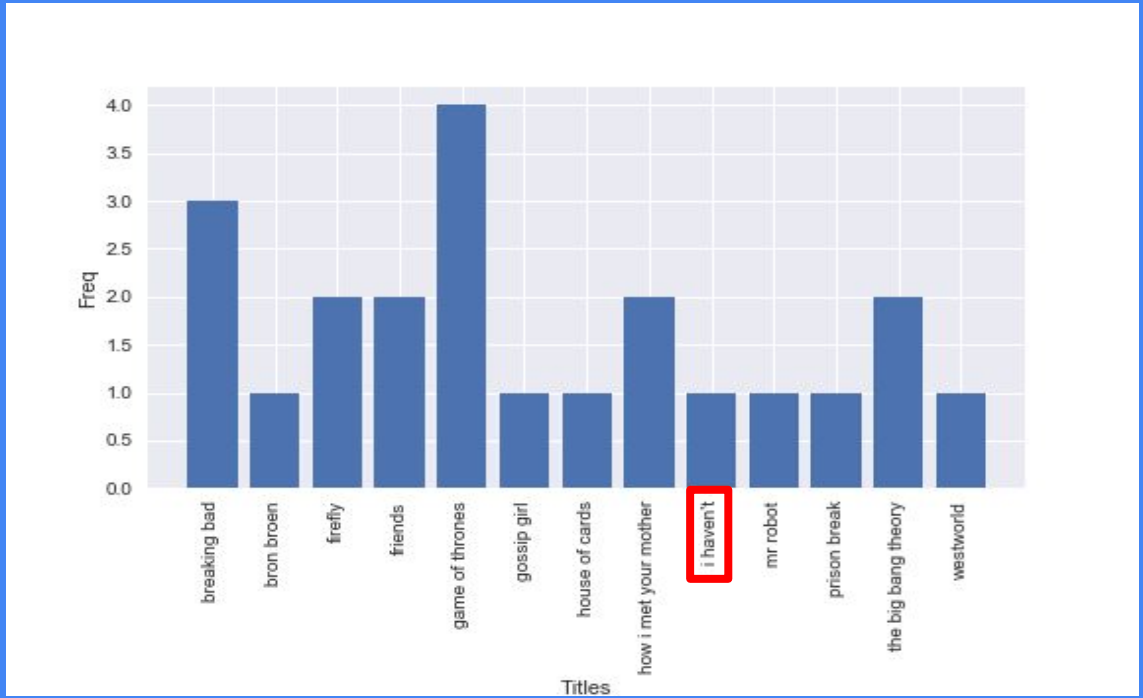
I guess this graph will be different in the Master...



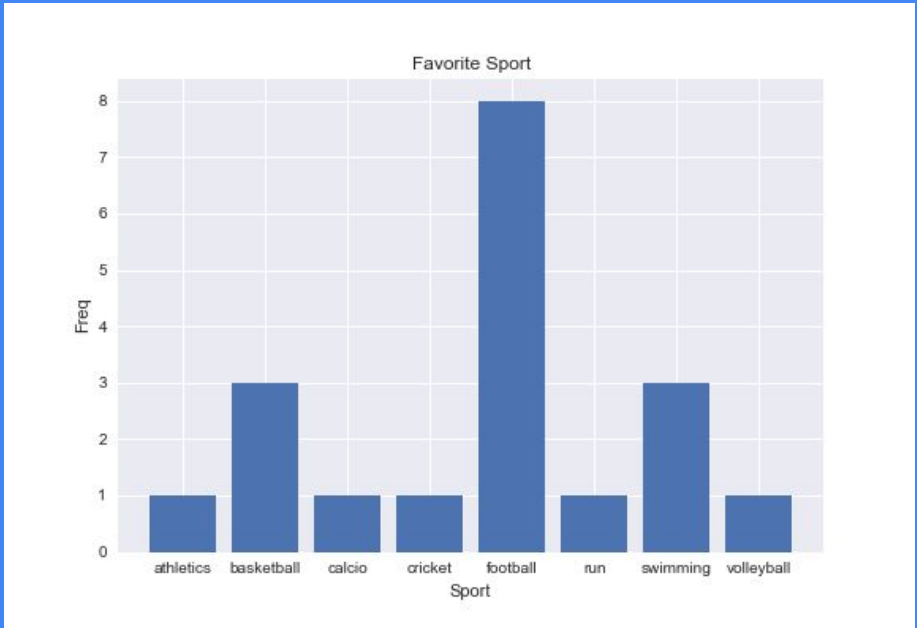
Here, we have a problem!



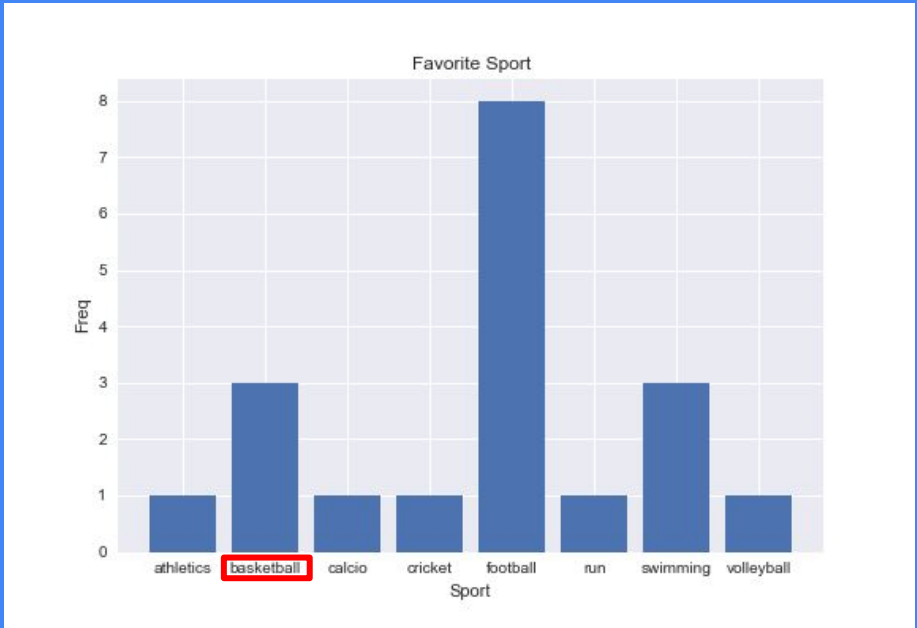
And the winner is...



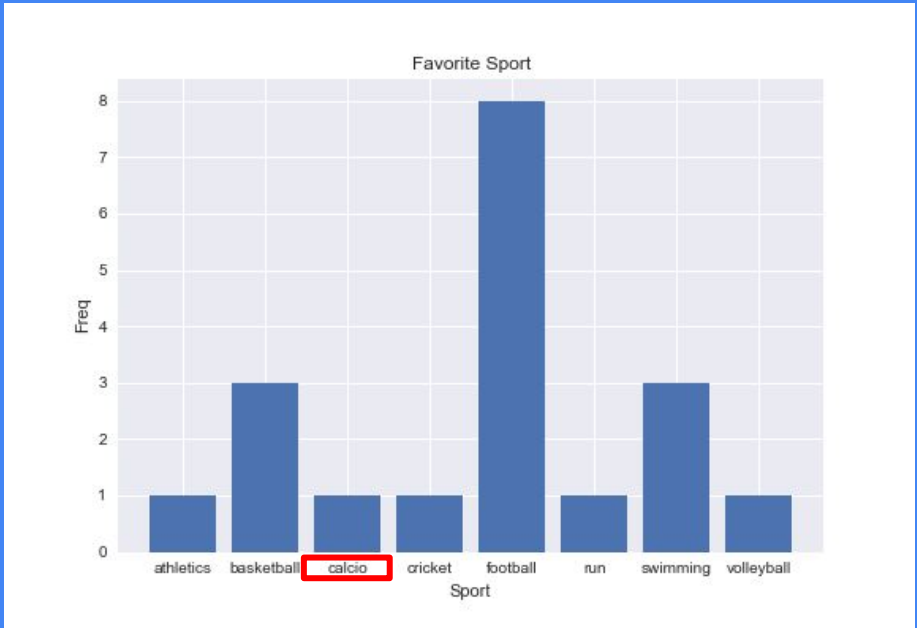
And the winner is...



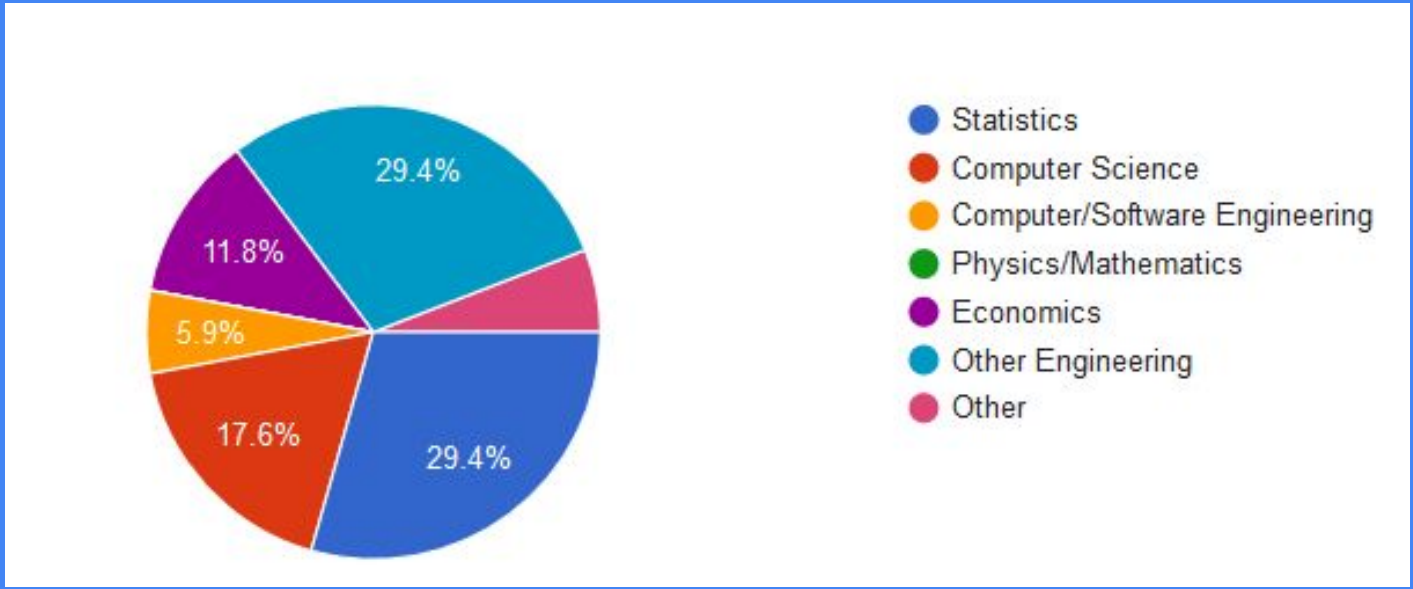
The best student awards go to...



The best student awards go to...



...another winner



Good news!

Who am I?

Who am I (!?)

- Bsc in Statistics
- MSc in Data Science
- Eurecat intern Fall 2016, Barcelona

SAPIENZA
UNIVERSITÀ DI ROMA

eurecat
Centre Tecnològic de Catalunya

If you have any questions
feel free to contact me:

francesco_fabbri@outlook.com

I’m going to...

I’m going to...

Terrify you

I’m going to...

Terrify you

List all the ones who annoys me, and then send THE LIST to the prof. Leonardi

I’m going to...

Terrify you

List all the ones who annoys me, and then send THE LIST to the prof. Leonardi

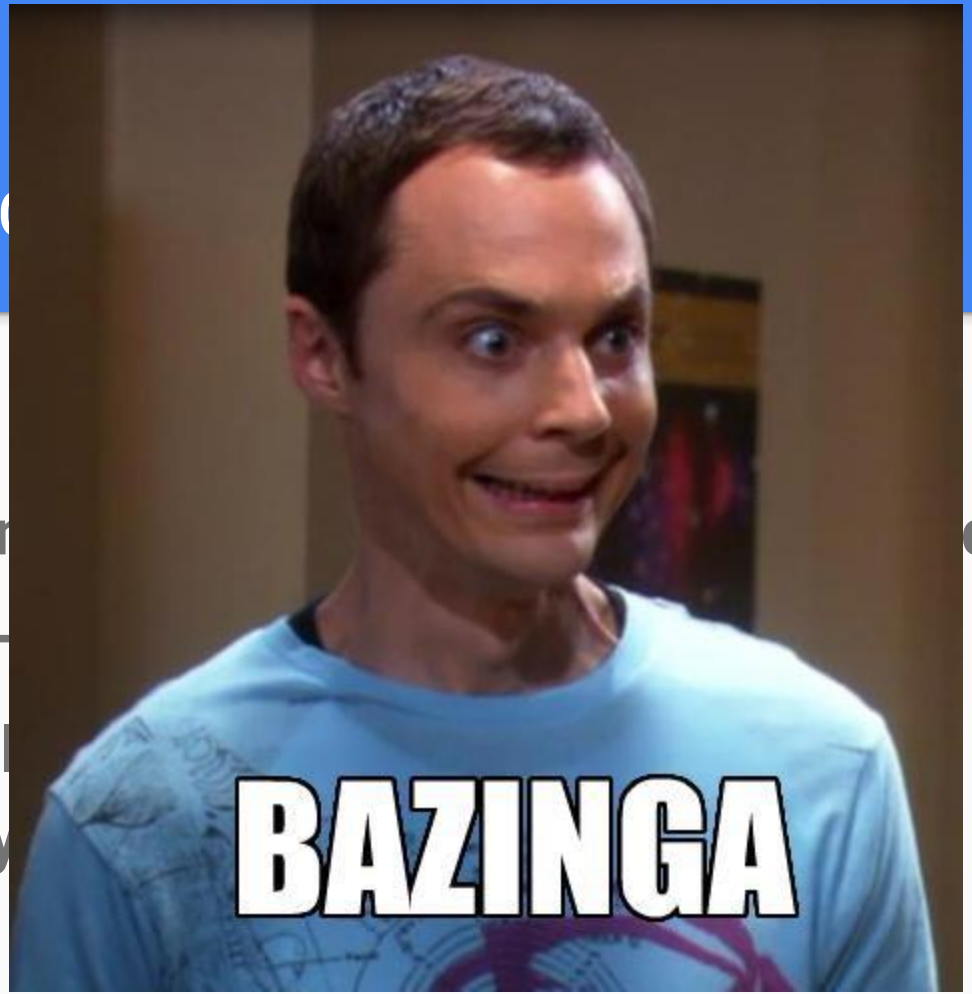
And obviously…
be unfriendly and rude all the time!

I’m going t

Terrify you

List all the or d THE LIST to the prof. L

And obvious be unfriendly



I’m going to… (seriously)

Introduce Python and its basic concepts

Follow you step by step in order to allow ALL of you understanding each thing I’m going to talk about

And obviously…

I’m available for all the questions you have about Python, the course, the Master, the meaning of life and so on...

1. Introduction
2. Data Structures
3. Control Flow Statements
4. Functions and Advanced Concepts

Introduction

1. Introduction - Basic Concepts

- Object-oriented programming language
- Open Source
- Great Online Community
- Easy to learn

1. Introduction - Python vs Rest of the World

1. Python code is **3 times** shorter than Java code and **5 times** than C++ code.

2. In Python you don’t have to declare the type of the variables.

3. There is a library for everything.

1. Introduction - Python vs Rest of the World

1. Introduction - Python vs Rest of the World

```
1 class Hello{
2   public static void main(String[] args){
3     System.out.print("Hello World\n");
4   }
5 }
```

1. Introduction - Python vs Rest of the World

```java
class Hello{
  public static void main(String[] args){
    System.out.print("Hello World\n");
  }
}
```

Java

1. Introduction - Python vs Rest of the World

```java
class Hello{
  public static void main(String[] args){
    System.out.print("Hello World\n");
  }
}
```

Java

```cpp
#include <iostream>
int main()
{
   std::cout << "Hello World";
   return 0;
}
```

1. Introduction - Python vs Rest of the World

```java
class Hello{
  public static void main(String[] args){
    System.out.print("Hello World\n");
  }
}
```

Java

```cpp
 #include <iostream>
int main()
{
   std::cout << "Hello World";
   return 0;
}
```

C++

1. Introduction - Python vs Rest of the World

```java
class Hello{
  public static void main(String[] args){
    System.out.print("Hello World\n");
  }
}
```

Java

```cpp
#include <iostream>
int main()
{
   std::cout << "Hello World";
   return 0;
}
```

C++

Python

TIPS - 1

TIPS - 1

How to learn fastly a new language as Python?

TIPS - 1

How to learn fastly a new language as Python?

1. Follow this course!

TIPS - 1

How to learn fastly a new language as Python?

1. Follow this course!
2. Think in a Pythonic-Way

TIPS - 1

How to learn fastly a new language as Python?

1. Follow this course!
2. Think in a Pythonic-Way
3. Use Stackoverflow!

Data Structures

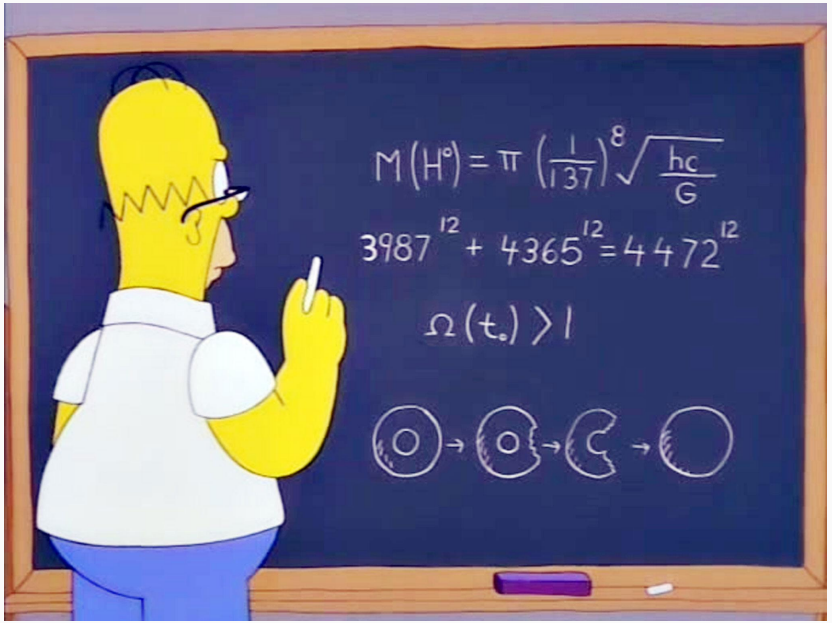
2. Data Structures - Strings and Numbers

```
In [1]: #assign variables
        welcome = 'Hey guys, welcome to the Python Crash Course!'
        print (welcome)

        Hey guys, welcome to the Python Crash Course!

In [2]: print ("You are reading a string!")

        You are reading a string!

In [3]: first_number = 5

In [4]: print ('This is an integer: %s...'%first_number)

        This is an integer: 5...

In [5]: '... even this one: ' + str(first_number)

Out[5]: '... even this one: 5'

In [6]: second_number = 5.12345 # As you can see, we use the DOT and not the COMMA

In [7]: 'And now, this a simple float: %s'%second_number

Out[7]: 'And now, this a simple float: 5.12345'
```

2. Data Structures - List

- One of most used data structure in Python.

- You can:
  - store everything you want.
  - ***append***
  - ***sort***
  - ***count*** elements
  - compute the ***len***gth



2. Data Structures - List

In [81]:
```
# 1 - one example
one_lst = [0,1.5678,'hello guys',4]
```

In [82]:
```
first_list = [1,2,3,3,4,5]
```

In [83]:
```
# 2 - Extract a sub-list
first_list[3:6]
```

Out[83]:

In [84]:
```
# 3 - Append an element in 2 ways
first_list.append(-5)
first_list += [-10]
len(first_list)
```

Out[84]:

In [85]:
```
# 5 - Sort the list
first_list = sorted(first_list)
first_list
```

Out[85]:

In [86]:
```
# 6 - Reverse order and get the last element
first_list = sorted(first_list, reverse = True)
first_list[-1]
```

Out[86]:

In [87]:
```
# 7 - Count the element
first_list.count(3)
```

Out[87]:

In [ ]:

2. Data Structures - List

```
In [81]: # 1 - one example
         one_lst = [0,1.5678,'hello guys',4]

In [82]: first_list = [1,2,3,3,4,5]

In [83]: # 2 - Extract a sub-list
         first_list[3:6]

Out[83]: [3, 4, 5]

In [84]: # 3 - Append an element in 2 ways
         first_list.append(-5)
         first_list += [-10]
         len(first_list)

Out[84]: [redacted]

In [85]: # 5 - Sort the list
         first_list = sorted(first_list)
         first_list

Out[85]: [redacted]

In [86]: # 6 - Reverse order and get the last element
         first_list = sorted(first_list, reverse = True)
         first_list[-1]

Out[86]: [redacted]

In [87]: # 7 - Count the element
         first_list.count(3)

Out[87]: [redacted]

In [ ]:
```

2. Data Structures - List

```
In [81]: # 1 - one example
         one_lst = [0,1.5678,'hello guys',4]

In [82]: first_list = [1,2,3,3,4,5]

In [83]: # 2 - Extract a sub-list
         first_list[3:6]

Out[83]: [3, 4, 5]

In [84]: # 3 - Append an element in 2 ways
         first_list.append(-5)
         first_list += [-10]
         len(first_list)

Out[84]: 8

In [85]: # 5 - Sort the list
         first_list = sorted(first_list)
         first_list

Out[85]: [redacted]

In [86]: # 6 - Reverse order and get the last element
         first_list = sorted(first_list, reverse = True)
         first_list[-1]

Out[86]: [redacted]

In [87]: # 7 - Count the element
         first_list.count(3)

Out[87]: [redacted]

In [ ]:
```

2. Data Structures - List

```
In [81]: # 1 - one example
         one_lst = [0,1.5678,'hello guys',4]

In [82]: first_list = [1,2,3,3,4,5]

In [83]: # 2 - Extract a sub-list
         first_list[3:6]

Out[83]: [3, 4, 5]

In [84]: # 3 - Append an element in 2 ways
         first_list.append(-5)
         first_list += [-10]
         len(first_list)

Out[84]: 8

In [85]: # 5 - Sort the list
         first_list = sorted(first_list)
         first_list

Out[85]: [-10, -5, 1, 2, 3, 3, 4, 5]

In [86]: # 6 - Reverse order and get the last element
         first_list = sorted(first_list, reverse = True)
         first_list[-1]

Out[86]: [redacted]

In [87]: # 7 - Count the element
         first_list.count(3)

Out[87]: [redacted]

In [ ]:
```

2. Data Structures - List

```
In [81]: # 1 - one example
         one_lst = [0,1.5678,'hello guys',4]

In [82]: first_list = [1,2,3,3,4,5]

In [83]: # 2 - Extract a sub-list
         first_list[3:6]

Out[83]: [3, 4, 5]

In [84]: # 3 - Append an element in 2 ways
         first_list.append(-5)
         first_list += [-10]
         len(first_list)

Out[84]: 8

In [85]: # 5 - Sort the list
         first_list = sorted(first_list)
         first_list

Out[85]: [-10, -5, 1, 2, 3, 3, 4, 5]

In [86]: # 6 - Reverse order and get the last element
         first_list = sorted(first_list, reverse = True)
         first_list[-1]

Out[86]: -10

In [87]: # 7 - Count the element
         first_list.count(3)

Out[87]:

In [ ]:
```

2. Data Structures - List

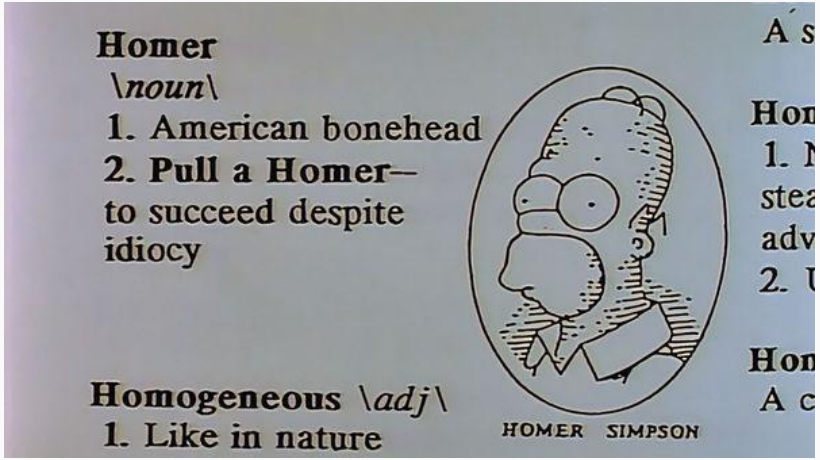
```
In [81]: # 1 - one example
         one_lst = [0,1.5678,'hello guys',4]

In [82]: first_list = [1,2,3,3,4,5]

In [83]: # 2 - Extract a sub-list
         first_list[3:6]

Out[83]: [3, 4, 5]

In [84]: # 3 - Append an element in 2 ways
         first_list.append(-5)
         first_list += [-10]
         len(first_list)

Out[84]: 8

In [85]: # 5 - Sort the list
         first_list = sorted(first_list)
         first_list

Out[85]: [-10, -5, 1, 2, 3, 3, 4, 5]

In [86]: # 6 - Reverse order and get the last element
         first_list = sorted(first_list, reverse = True)
         first_list[-1]

Out[86]: -10

In [87]: # 7 - Count the element
         first_list.count(3)

Out[87]: 2

In [ ]:
```

2. Data Structures - Vocabulary

- Crucial in Python: fast

- Looking inside...
  - **Keys**:
    - Int and strings
    - Tuples (only with int and strings)
  - **Values**: everything you want (more or less…)

- Used to store data in format like *json* or txt (later more details…).



2. Data Structures - Vocabulary

```
In [30]: # 1 - Build dictionary
         first_dict = {}
         first_dict[1] = 'Data'
         first_dict['two'] = 'Science'
         first_dict[(3,'four')] = 'Master'

In [31]: # 2 - Alternative way
         first_dict = {1: 'Data', 'two': 'Science', (3, 'four'): 'Master'}

In [32]: # 3 - Show the keys!
         first_dict.keys()

Out[32]: dict_keys([1, 'two', (3, 'four')])

In [33]: # 4 - Use them
         list(first_dict.keys())

Out[33]: [1, 'two', (3, 'four')]

In [34]: # 5 - Values
         list(first_dict.values())

Out[34]: ['Data', 'Science', 'Master']

In [35]: # 6 - And if I would them together...?
         print ('We will see later...')

         We will see later...
```

2. Data Structures - Sets and Tuples

- **Sets**: useful for detecting unique elements

- **Tuples**: useful for building dictionary or keeping data during a process.

2. Data Structures - Sets and Tuples

- **Sets**: useful for detecting unique elements

- **Tuples**: useful fo dictionary or kee a process.

2. Data Structures - Try again...

We have a list of strings.

2. Data Structures - Try again...

```
In [190]: lst1 = ['Anna']*3 + ['Alessandro']*2 + ['Silvia'] + [('Carlo','Michele')]
          lst1

Out[190]: ['Anna',
           'Anna',
           'Anna',
           'Alessandro',
           'Alessandro',
           'Silvia',
           ('Carlo', 'Michele')]
```

2. Data Structures - Try again...

```
In [190]: lst1 = ['Anna']*3 + ['Alessandro']*2 + ['Silvia'] + [('Carlo','Michele')]
          lst1

Out[190]: ['Anna',
           'Anna',
           'Anna',
           'Alessandro',
           'Alessandro',
           'Silvia',
           ('Carlo', 'Michele')]
```

Starting from this list we want to build a dictionary which contains only the distinct names, stored as values.

2. Data Structures - Try again...

First, we need a list of only string, then we need to change the structure of the last element, which is a tuple .

2. Data Structures - Try again...

First, we need a list of only string, then we need to change the structure of the last element, which is a tuple .

```
In [191]: sub_lst = list(lst1[-1])

In [192]: lst1 = lst1[:-1] + sub_lst
          lst1

Out[192]: ['Anna',
           'Anna',
           'Anna',
           'Alessandro',
           'Alessandro',
           'Silvia',
           'Carlo',
           'Michele']
```

2. Data Structures - Try again...

Now, we can obtain the a list of distinct names

How?!

Applying the built-in functions ***set*** and ***list***

2. Data Structures - Try again...

```
In [193]: lst1 = set(lst1)
          lst1

Out[193]: {'Alessandro', 'Anna', 'Carlo', 'Michele', 'Silvia'}

In [194]: lst1 = list(lst1)
          lst1

Out[194]: ['Alessandro', 'Michele', 'Anna', 'Silvia', 'Carlo']
```

Applying the built-in functions ***set*** and ***list***

2. Data Structures - Try again...

How to store in the dictionary our names?!

Unique IDs

2. Data Structures - Try again...

How to store in the dictionary our names?!

Unique IDs

We use the function ***range*** which, IN THIS CASE, gets as argument an integer N and returns a list of integers starting from zero to the number N-1.

2. Data Structures - Try again...

How to store in the dictionary our names?!

Unique IDs

We use the function ***range*** which, IN THIS CASE, gets as argument an integer N and returns a list of integers starting from zero to the number N-1.

```
In [195]: lst2 = range(len(lst1))
          print (list(lst2))

          [0, 1, 2, 3, 4]
```

2. Data Structures - Try again...

We are almost done! We just need to put all together!

How?!

Using even now, built-in functions.

2. Data Structures - Try again...

We are almost done! We just need to put all together!

How?!

Using even now, built-in functions.

1. ***zip***: aggregates elements from each of the 2 list involved.
2. ***dict***: from a list of tuples it returns a dictionary where the first elements are the keys and the second the values

2. Data Structures - Try again...

```
In [196]: list(zip(lst2, lst1))
Out[196]: [(0, 'Alessandro'), (1, 'Michele'), (2, 'Anna'), (3, 'Silvia'), (4, 'Carlo')]

In [197]: dict1 = dict(zip(lst2, lst1))
          dict1
Out[197]: {0: 'Alessandro', 1: 'Michele', 2: 'Anna', 3: 'Silvia', 4: 'Carlo'}
```

1. ***zip***: aggregates elements from each of the 2 list involved.
2. ***dict***: from a list of tuples it returns a dictionary where the first elements are the keys and the second the values

2. Data Structures - Try again...

```
In [196]: list(zip(lst2, lst1))
Out[196]: [(0, 'Alessandro'), (1, 'Michele'),

In [197]: dict1 = dict(zip(lst2, lst1))
          dict1
Out[197]: {0: 'Alessandro', 1: 'Michele', 2:
```

1. ***zip***: aggregates of the 2 list involved.
2. ***dict***: from a list lictionary where the first elemen e second the values

Control Flow Statements

3. CFS - Looping with **for** and...

In [260]:
```
# 1 - Using the loop for appending elements
new_lst = []
for x in range(5):
    new_lst.append(x)
new_lst
```

Out[260]:
```
[0, 1, 2, 3, 4]
```

In [261]:
```
# 2 - A bit more tricky...
new_lst = []
lst3 = zip(range(1,7),range(5,11))
for x,y in lst3:
    new_lst.append([y,x])
new_lst
```

Out[261]:
```
[[5, 1], [6, 2], [7, 3], [8, 4], [9, 5], [10, 6]]
```

3. CFS - … a little taste of List Comprehension

```
In [264]: # 1 - Optimization in many ways...
          lst4 = [x + y for [x,y] in new_lst]
          lst4

Out[264]: [6, 8, 10, 12, 14, 16]

In [265]: 'Soon more details...'

Out[265]: 'Soon more details...'
```

3. CFS - **If** and **While**

In [285]:
```
# 1 - Simple IF condition
num1 = 5
num2 = 2
if num1 > num2:
    print(list(range(5)))
```

In [286]:
```
# 2 - FOR and IF together
only_even = []
for x in range(10):
    if x%2 == 0:
        only_even.append(x)
only_even
```

Out[286]:

In [287]:
```
# 3 - List Comprehension
only_even = [x for x in range(10) if x % 2 == 0]
only_even
```

Out[287]:

In [288]:
```
# 4 - A bit more tricky...
counter = 10
res = []
stop = 50
while counter < stop:
    lst = [x for x in range(counter -10, counter+1) if x % 5 == 0]
    res +=lst
    counter +=10
res
```

Out[288]:

3. CFS - **If** and **While**

In [285]:
```
# 1 - Simple IF condition
num1 = 5
num2 = 2
if num1 > num2:
    print(list(range(5)))
```

```
[0, 1, 2, 3, 4]
```

In [286]:
```
# 2 - FOR and IF together
only_even = []
for x in range(10):
    if x%2 == 0:
        only_even.append(x)
only_even
```

Out[286]:

In [287]:
```
# 3 - List Comprehension
only_even = [x for x in range(10) if x % 2 == 0]
only_even
```

Out[287]:

In [288]:
```
# 4 - A bit more tricky...
counter = 10
res = []
stop = 50
while counter < stop:
    lst = [x for x in range(counter -10, counter+1) if x % 5 == 0]
    res +=lst
    counter +=10
res
```

Out[288]:

3. CFS - **If** and **While**

In [285]:
```
# 1 - Simple IF condition
num1 = 5
num2 = 2
if num1 > num2:
    print(list(range(5)))
```

```
[0, 1, 2, 3, 4]
```

In [286]:
```
# 2 - FOR and IF together
only_even = []
for x in range(10):
    if x%2 == 0:
        only_even.append(x)
only_even
```

Out[286]:
```
[0, 2, 4, 6, 8]
```

In [287]:
```
# 3 - List Comprehension
only_even = [x for x in range(10) if x % 2 == 0]
only_even
```

Out[287]:
```
[0, 2, 4, 6, 8]
```

In [288]:
```
# 4 - A bit more tricky...
counter = 10
res = []
stop = 50
while counter < stop:
    lst = [x for x in range(counter -10, counter+1) if x % 5 == 0]
    res +=lst
    counter +=10
res
```

Out[288]:

3. CFS - **If** and **While**

In [300]:
```
# 1 - Simple IF condition
num1 = 5
num2 = 2
if num1 > num2:
    print(list(range(5)))
```

```
[0, 1, 2, 3, 4]
```

In [301]:
```
# 2 - FOR and IF together
only_even = []
for x in range(10):
    if x%2 == 0:
        only_even.append(x)
only_even
```

Out[301]:
```
[0, 2, 4, 6, 8]
```

In [302]:
```
# 3 - List Comprehension
only_even = [x for x in range(10) if x % 2 == 0]
only_even
```

Out[302]:
```
[0, 2, 4, 6, 8]
```

In [303]:
```
# 4 - A bit more tricky...
counter = 10
res = []
stop = 50
while counter < stop:
    lst = [x for x in range(counter, counter+10) if x % 5 == 0]
    res +=lst
    counter +=10
res
```

Out[303]:
```
[10, 15, 20, 25, 30, 35, 40, 45]
```

TIPS - 2

Optimize as much as possible!

TIPS - 2

Optimize as much as possible!

1. Exploit the data structures you can find in Python.

TIPS - 2

Optimize as much as possible!

1. Exploit the data structures you can find in Python.
2. Produce readable code (http://sphinxcontrib-napoleon.readthedocs.io/en/latest/example_google.html)

4. Built-in Functions - **range, any, all, enumerate...**

In [100]:
```
for idx, value in enumerate(range(10,20,2)):
    print(idx,value)
```

```
0 10
1 12
2 14
3 16
4 18
```

In [101]:
```
# 1 - Enumerate & Range
[(idx, value) for idx, value in enumerate(range(10,20,2)) if idx %2 == 0]
```

Out[101]:

In [102]:
```
# 2 - Any
any(name in 'Homer is a state of mind' for name in simpson)
```

Out[102]:

In [103]:
```
any(name in 'Homer is a state of mind' for name in simpson[1:])
```

Out[103]:

In [104]:
```
# 3 - All
simpson = ['Homer', 'Bart', 'Lisa']
all(name in 'Lisa and Bart are the daughters' for name in simpson)
```

Out[104]:

In [105]:
```
all(name in 'Lisa and Bart are the daughters' for name in simpson[1:])
```

Out[105]:

4. Built-in Functions - **range, any, all, enumerate...**

In [100]:
```
for idx, value in enumerate(range(10,20,2)):
    print(idx,value)
```

```
0 10
1 12
2 14
3 16
4 18
```

In [101]:
```
# 1 - Enumerate & Range
[(idx, value) for idx, value in enumerate(range(10,20,2)) if idx %2 == 0]
```

Out[101]:
```
[(0, 10), (2, 14), (4, 18)]
```

In [102]:
```
# 2 - Any
any(name in 'Homer is a state of mind' for name in simpson)
```

Out[102]:

In [103]:
```
any(name in 'Homer is a state of mind' for name in simpson[1:])
```

Out[103]:

In [104]:
```
# 3 - All
simpson = ['Homer', 'Bart', 'Lisa']
all(name in 'Lisa and Bart are the daughters' for name in simpson)
```

Out[104]:

In [105]:
```
all(name in 'Lisa and Bart are the daughters' for name in simpson[1:])
```

Out[105]:

4. Built-in Functions - **range, any, all, enumerate...**

In [100]:
```
for idx, value in enumerate(range(10,20,2)):
    print(idx,value)
```

```
0 10
1 12
2 14
3 16
4 18
```

In [101]:
```
# 1 - Enumerate & Range
[(idx, value) for idx, value in enumerate(range(10,20,2)) if idx %2 == 0]
```

Out[101]:
```
[(0, 10), (2, 14), (4, 18)]
```

In [102]:
```
# 2 - Any
any(name in 'Homer is a state of mind' for name in simpson)
```

Out[102]:
```
True
```

In [103]:
```
any(name in 'Homer is a state of mind' for name in simpson[1:])
```

Out[103]:

In [104]:
```
# 3 - All
simpson = ['Homer', 'Bart', 'Lisa']
all(name in 'Lisa and Bart are the daughters' for name in simpson)
```

Out[104]:

In [105]:
```
all(name in 'Lisa and Bart are the daughters' for name in simpson[1:])
```

Out[105]:

4. Built-in Functions - **range, any, all, enumerate...**

In [100]:
```
for idx, value in enumerate(range(10,20,2)):
    print(idx,value)
```

```
0 10
1 12
2 14
3 16
4 18
```

In [101]:
```
# 1 - Enumerate & Range
[(idx, value) for idx, value in enumerate(range(10,20,2)) if idx %2 == 0]
```

Out[101]:
```
[(0, 10), (2, 14), (4, 18)]
```

In [102]:
```
# 2 - Any
any(name in 'Homer is a state of mind' for name in simpson)
```

Out[102]:
```
True
```

In [103]:
```
any(name in 'Homer is a state of mind' for name in simpson[1:])
```

Out[103]:
```
False
```

In [104]:
```
# 3 - All
simpson = ['Homer', 'Bart', 'Lisa']
all(name in 'Lisa and Bart are the daughters' for name in simpson)
```

Out[104]:

In [105]:
```
all(name in 'Lisa and Bart are the daughters' for name in simpson[1:])
```

Out[105]:

4. Built-in Functions - **range, any, all, enumerate...**

```
In [100]: for idx, value in enumerate(range(10,20,2)):
              print(idx,value)

          0 10
          1 12
          2 14
          3 16
          4 18

In [101]: # 1 - Enumerate & Range
          [(idx, value) for idx, value in enumerate(range(10,20,2)) if idx %2 == 0]

Out[101]: [(0, 10), (2, 14), (4, 18)]

In [102]: # 2 - Any
          any(name in 'Homer is a state of mind' for name in simpson)

Out[102]: True

In [103]: any(name in 'Homer is a state of mind' for name in simpson[1:])

Out[103]: False

In [104]: # 3 - All
          simpson = ['Homer', 'Bart', 'Lisa']
          all(name in 'Lisa and Bart are the daughters' for name in simpson)

Out[104]: False

In [105]: all(name in 'Lisa and Bart are the daughters' for name in simpson[1:])

Out[105]:
```

4. Built-in Functions - **range, any, all, enumerate...**

```
In [100]: for idx, value in enumerate(range(10,20,2)):
              print(idx,value)

          0 10
          1 12
          2 14
          3 16
          4 18

In [101]: # 1 - Enumerate & Range
          [(idx, value) for idx, value in enumerate(range(10,20,2)) if idx %2 == 0]

Out[101]: [(0, 10), (2, 14), (4, 18)]

In [102]: # 2 - Any
          any(name in 'Homer is a state of mind' for name in simpson)

Out[102]: True

In [103]: any(name in 'Homer is a state of mind' for name in simpson[1:])

Out[103]: False

In [104]: # 3 - All
          simpson = ['Homer', 'Bart', 'Lisa']
          all(name in 'Lisa and Bart are the daughters' for name in simpson)

Out[104]: False

In [105]: all(name in 'Lisa and Bart are the daughters' for name in simpson[1:])

Out[105]: True
```

4. Built-in Functions - ...**zip, open** and the first **lib**

In [115]:
```
# 1 - New Lists
lst1 = range(10)
lst2 = ["The code doesn't work!" if x %2 == 0 else "I don't know why!" for x in range(10)]
```

In [116]:
```
# 2 - Open the file and save strings
outfile = open('saved_strings.txt', 'w')
outfile.write("\n".join(lst2))
outfile.close()
```

In [117]:
```
# 3 - Read the strings in the file
with open('saved_strings.txt', 'r') as f:
    lines = f.readlines()
print(lines)
```

```
["The code doesn't work!\n", "I don't know why!\n", "The code doesn't work!\n", "I don't know why!\n", "The code doesn't work!
\n", "I don't know why!\n", "The code doesn't work!\n", "I don't know why!\n", "The code doesn't work!\n", "I don't know why!"]
```

In [118]:
```
print([x.strip() for x in lines])
```

```
["The code doesn't work!", "I don't know why!", "The code doesn't work!", "I don't know why!", "The code doesn't work!", "I do
n't know why!", "The code doesn't work!", "I don't know why!", "The code doesn't work!", "I don't know why!"]
```

In [119]:
```
# 4 - Zip the dict
zipped = list(zip(lst1, lst2))
one_dict = dict(zipped)
print(one_dict)
```

```
{0: "The code doesn't work!", 1: "I don't know why!", 2: "The code doesn't work!", 3: "I don't know why!", 4: "The code doesn't
 work!", 5: "I don't know why!", 6: "The code doesn't work!", 7: "I don't know why!", 8: "The code doesn't work!", 9: "I don't k
now why!"}
```

4. Built-in Functions - ...**zip, open** and the first **lib**

In [84]:
```
# 5 - Save the dictionary in a file using the json lib
import json
with open('new_dict.json', 'w') as f:
    json.dump(zipped, f)
    f.close()
```

In [85]:
```
# 6 - Save the dictionary in a file using the function from the json lib
from json import dump
with open('new_dict.json', 'w') as f:
    dump(one_dict, f)
    f.close()
```

In [120]:
```
# 7 - Load the dictionary stored
from json import load
with open('new_dict.json', 'r') as f:
    one_dict = load(f)
    f.close()
print(one_dict)
```

```
{'0': "The code doesn't work!", '1': "I don't know why!", '2': "The code doesn't work!", '3': "I don't know why!", '4': "The cod
e doesn't work!", '5': "I don't know why!", '6': "The code doesn't work!", '7': "I don't know why!", '8': "The code doesn't wor
k!", '9': "I don't know why!"}
```

4. Advanced concepts - Deal with errors

4. Advanced concepts - Deal with errors

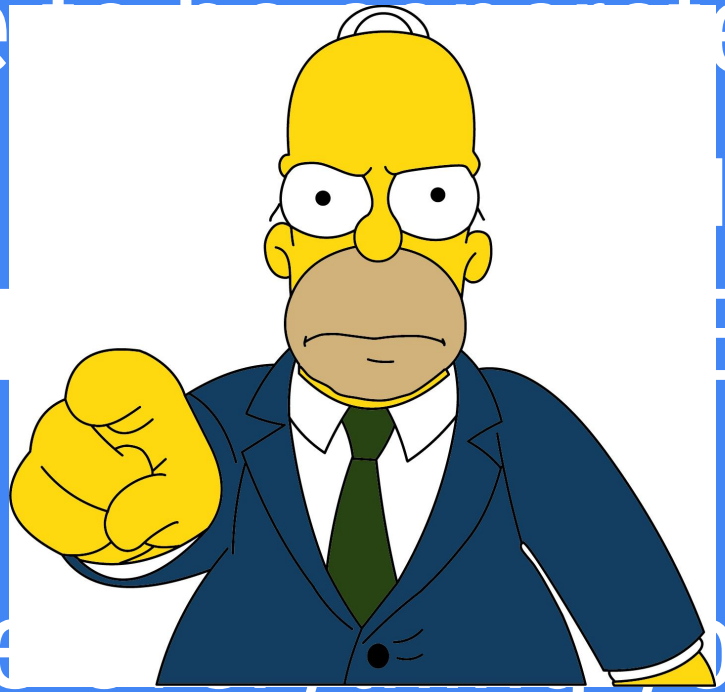
The end… (?!)

...Not Yet!

It’s time to be concrete and tell you the truth about the DATA SCIENCE!

Ask me everything you want!

It’s time e and tell you t the E!



Ask me u want!