Introduction to Python

Python is a general purpose programming language, which supports a variety of paradigms, and it has a well-known data structures.

A data structure is a way to store and organize data, to be accessed efficiently. Common examples of data structures are the following

- Heaps
- Stacks
- Queues
- Lists
- Binary Search Trees (BST)
- Hashes (Dictionaries)
- Arrays

Typical pitfalls when programming with python, it to do everything like a "c style". The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than **nested.**
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!

Python is an interpreted language, this means that it is not “compiled” such c++, c , obj-c, go, or any other.

It has an interpreter, and the sentences are evaluated at “execution”.
Python was developed to be easy to read, this means, that if a developer starts to check out your code, he will be able to read at most part of it, and this helps to understand the logic inside your program.

Python uses indentation to specify blocks of code, Let's see the following c++ function.

```
std::list<float> sort(float *data, int length){
    // Creating the list
    std::list<float> listA;

    for (auto i =0; i<length;i++){
        listA.push_back(*(data+i));

        }
    return listA;
    }
}
```

Python doesn´t need “{}” to specify where a block starts it needs indentation using tabs or spaces and “:”, also the return type it is not required, the function can return everything you want or “none” type. Example:

```
def square(data):
    """ Creating a new copy and returns its square """
    ListA = []
    for element in data:
        ListA.append(element**2)
    return ListA
```

Everything in python is an object, so each one has “attributes” and “methods”, basic functions in python.

- dir(object): shows attributes and methods of an object
- id(object): shows the reference number of an object
- type(object): shows the “type” of object

Valid control structures

- if
- while
- for
- break
- continue