Interview questions

How is python interpreted ?

Python language is an interpreted language. Python program runs directly from the source code. It converts the source code that is written by the programmer into an intermediate language, which is again translated into machine language that has to be executed.

The source code is interpreted each time a python program is run.

What is the difference between list and tuples?

- Lists are mutable and can be edited, whereas tuples are immutable
- Lists are slower than tuples (tuples being immutables, they are stored in a single block memory)
- Syntax of lists is: x=[1,2] whereas syntax of tuples is x=(1,2)

What is dictionary in Python ?

- Built-in python object which defines a one to one relationship between a list of keys and a list of values

- Example:
  `dict={“SX5E”:3500,”FTSE100”:7000,”SP500”:2800}`

What is map function in Python?

- map() applies a given function to each item of an iterable and returns a list of the results

- Example:

```
numbers = (1, 2, 3, 5)
g= lambda x:x**2
result = map(g, numbers)
print(result)

[1, 4, 9, 25]
```

What are f-strings in Python?

- String formatting released since Python 3.6 which lets the user embed Python expressions inside string constants

- Example:

```
a = 5
b = 10

>>> f'Five plus ten is {a + b} and not {2 * (a + b)}.'
'Five plus ten is 15 and not 30.'
```

How are arguments passed ? by value or by reference?

- Arguments are passed neither by value and nor by reference in Python - instead they are passed by assignment (i.e. reference to the object). Behaviour similar to a by-reference for a mutable object, and by-value for an immutable one.

- "variables" in Python would be more accurately called names. "assignment" refers to the binding of a name to an object.

Example:

```
a=[5] -> list named “a” corresponding to an object O
b=a ->new name “b” referring to the same object O
b[0]=1 -> changing content of object O
print(a) -> will return [1]
```

What is a list comprehension in Python?

- List comprehensions provide a concise way to create lists: consists of brackets containing an expression followed by a for clause

- Example:

```
new_list = []
for i in [1,2,3]:
    if filter(i):
        new_list.append(expressions(i))
```

Would become as a list comprehension:

```
new_list = [expression(i) for i in [1,2,3] if filter(i)]
```

How can you convert a number to a string?

- Use the inbuilt function str()

Example:

```
a=1
a_string=str(a)
```

What are the main python packages for data science ?

- **Pandas**: used for data manipulation, aggregation, and visualization. Main structures are either “Series” (1 dimensional) or “Dataframes (2 dimensional)
- **Numpy**: operations on n-arrays and matrices
- **Scipy**: contains modules for linear algebra, optimization, integration, and statistics.
- **Matplotlib:** visualization of data (line plot, scatter plots…)
- **Bokeh:** interactive visualization (based on Javascript modules)

What Are The Principal Differences Between The Lambda And Def?

- Def can hold multiple expressions while lambda is a uni-expression function.
- Def can have a return statement, whereas Lambda can’t
- Lambda supports to get used inside a list and dictionary.

Example:

```
g= lambda x:x**2

def g(x):
    return x**2
```

What is the difference between shallow copy and deep copy?

Difference between shallow and deep copying is only relevant for compound objects (objects that contain other objects, like lists)

- **Shallow Copy:copy.copy(x)**
  Constructs a new compound object with references to the objects found in the original. Modifying the original object could then modify as well the “copy” object

- **Deep Copy: copy.deepcopy(x)**
  Constructs a new compound object and then, recursively, inserts copies into it of the objects found in the original. Copying an object this way walks the whole object tree to create a fully independent clone

What is the difference between shallow copy and deep copy?

```
import copy

a = [1, 2, 3]
b = [4, 5, 6]
c = [a, b]

d = c

print(id(c) == id(d)  )      #True, same objects
print(id(c[0]) == id(d[0])) # True
```

```
True
True
```

What is the difference between shallow copy and deep copy?

Entrée [4]:

```
#Shallow Copy

d = copy.copy(c)

print(id(c) == id(d) )         # False - d is now a new object
print(id(c[0]) == id(d[0]))  # True - d[0] is the same object as c[0]
```

```
False
True
```

Entrée [5]:

```
#Deep Copy

d = copy.deepcopy(c)

print(id(c) == id(d) )         # False - d is now a new object
print(id(c[0]) == id(d[0]))  # False - d[0] is now also a new object
```

```
False
False
```

What are the pros and cons of Python ?

- **Pros**
  - Easy to learn (light syntax, no need to declare variables...)
  - Scripting language
  - Supports object oriented programming
  - Many frameworks for web programming (Flask, Django)
  - Large quantity of resources

- **Cons**
  - Slower than other languages like C++
  - Not efficient for memory consuming tasks