Python - review for interviews

What is python?

Python is an easy **script language** to learn, powerful programming language. It has efficient high-level data structures (**non strongly typed**) and effective approach to object-oriented programming. Python has **interpreted nature**, make it an ideal language for scripting and rapid application development in many areas on most platforms.
The Python interpreter is easily extended with new functions and data types implemented in C or C++ (or other languages callable from C).

Write a sorting algorithm

```
x=[7,1,5,2,8,10]

for i in range (1,len(x)):
  for j in range (i,len(x)):
     if x[j-1]>x[j]:
        temp=x[j-1]
        x[j-1]=x[j]
        x[j]=temp

print(x)
```

In the given sorting algorithm above - how many times will the condition run?

If the array is length “n”, the condition will run: n*(n!)

What are the common types in python?

```
#---------------------VALUE TYPES-------------------------
x=4
print(type(x))  # <class 'int'>

x=4.5
print(type(x))  # <class 'float'>

x=True
print(type(x))  # <class 'bool'>

x=None
print(type(x))  # <class 'NoneType'>

x=bytes(2)
print(type(x))  # <class 'bytes'>


#---------------------REF TYPES-------------------------
x='a'
print(type(x))  # <class 'str'>


x="a"
print(type(x))  # <class 'str'>


x={"name":"bob"}
print(type(x))  # <class 'dict'>


x=("name","bob")
print(type(x))  # <class 'tuple'>


x=["name","bob"]
print(type(x))  # <class 'list'>

x=lambda y:y*2
print(type(x))  # <class 'function'>

x=(2+3j)
print(type(x))  # <class 'complex'>


x=set()
x.add("Ben")
print(type(x))  # <class 'set'>
```

How to write ternary operator in python?

```
num1=6
num2=7


# get the min between num1 and num2
x=num1 if num1<num2 else num2
print(x)  # --> 6

# get the max between num1 and num2
x=num1 if num1>num2 else num2
print(x)  # --> 7
```

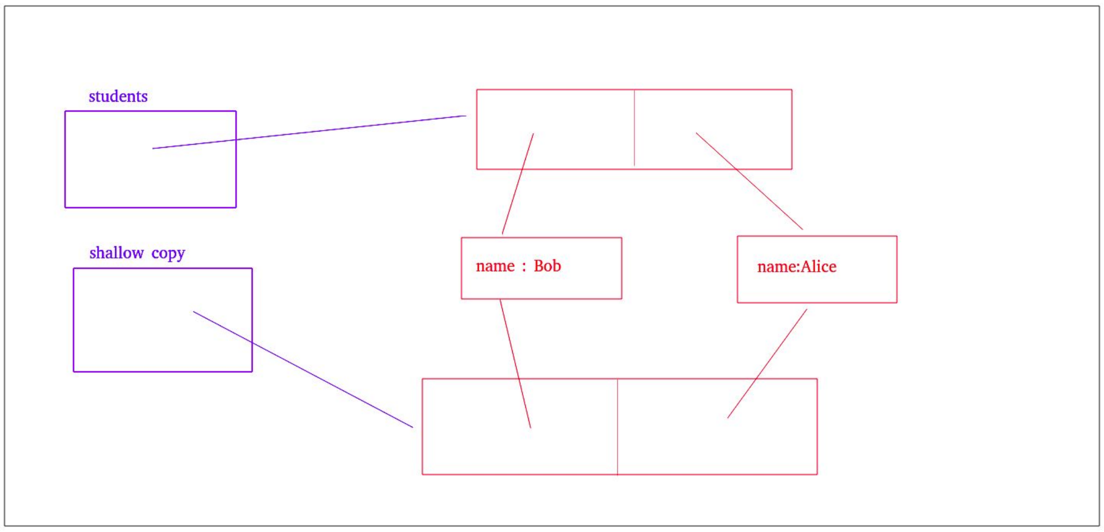
What is the difference between shallow copy and deep copy?

Shallow copy is used when a new instance type gets created and it keeps the values that are copied in the new instance. Shallow copy is used to copy the reference pointers just like it copies the values. These references point to the original objects and the changes made in any member of the class will also affect the original copy of it. Shallow copy allows faster execution of the program and it depends on the size of the data that is used.

Deep copy is used to store the values that are already copied. Deep copy doesn’t copy the reference pointers to the objects. It makes the reference to an object and the new object that is pointed by some other object gets stored. The changes made in the original copy won’t affect any other copy that uses the object. Deep copy makes execution of the program slower due to making certain copies for each object that is been called.

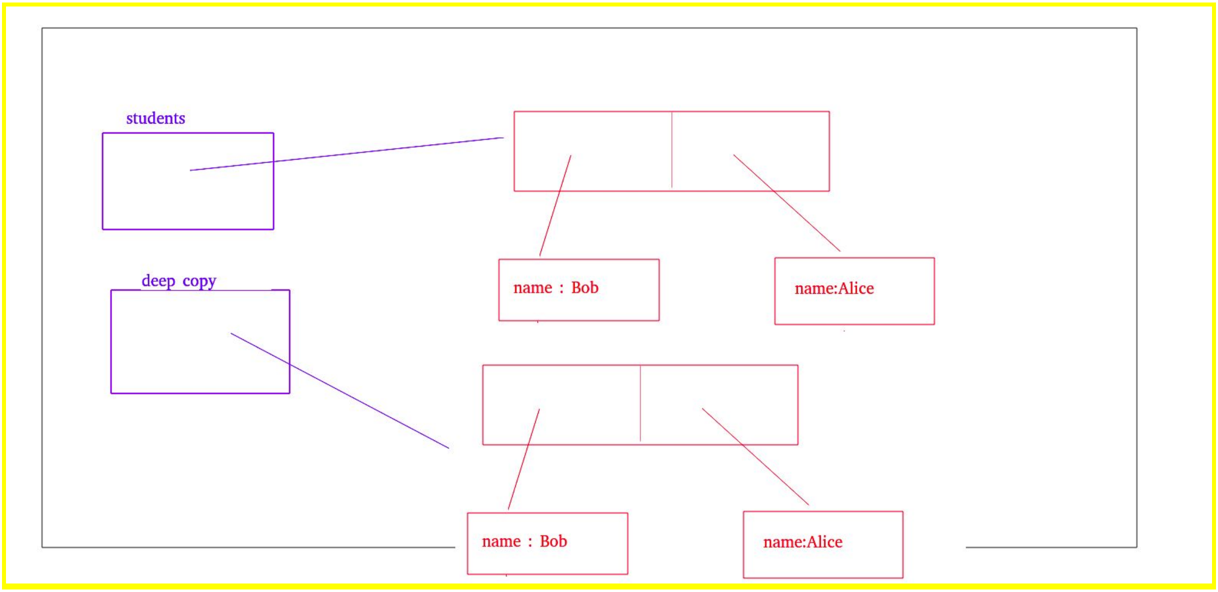
Shallow copy example:

```
students=[{"name":"Bob"}, {"name":"Alice"}]

shallow_copy=[]


for i in range(0, len(students)):

   shallow_copy.append(students[i])


print("students",students)           #--> students [{'name': 'Bob'}, {'name': 'Alice'}]

print("shallow_copy",shallow_copy)  #--> shallow_copy [{'name': 'Bob'}, {'name': 'Alice'}]


shallow_copy.append({"name":"Tom"})

print("students",students)           #--> students [{'name': 'Bob'}, {'name': 'Alice'}]

print("shallow_copy",shallow_copy)  #--> shallow_copy [{'name': 'Bob'}, {'name': 'Alice'}, {'name': 'Tom'}]


shallow_copy[0]['name']="Tom"

print("students",students)           #--> students [{'name': 'Tom'}, {'name': 'Alice'}]

print("shallow_copy",shallow_copy)  #--> shallow_copy [{'name': 'Tom'}, {'name': 'Alice'}, {'name': 'Tom'}]
```



Deep copy example:

```
students=[{"name":"Bob"}, {"name":"Alice"}]

deep_copy=[]

for i in range(0, len(students)):

   deep_copy.append({"name":students[i]["name"]})

print("students",students)          #--> students [{'name': 'Bob'}, {'name': 'Alice'}]

print("deep_copy",deep_copy)        #--> deep_copy [{'name': 'Bob'}, {'name': 'Alice'}]



deep_copy.append({"name":"Tom"})

print("students",students)          #--> students [{'name': 'Bob'}, {'name': 'Alice'}]

print("deep_copy",deep_copy)        #--> deep_copy [{'name': 'Bob'}, {'name': 'Alice'}, {'name': 'Tom'}]



deep_copy[0]['name']="Tom"

print("students",students)          #--> students [{'name': 'Bob'}, {'name': 'Alice'}]

print("deep_copy",deep_copy)        #--> deep_copy [{'name': 'Tom'}, {'name': 'Alice'}, {'name': 'Tom'}]
```



how can you copy an object in Python?

To copy an object in Python, you can use copy () or deepcopy() for the general case.

```
import copy


x={"hobbies":["Sport","Math"]}

x1=copy.copy(x)          # x1 shallow copy of x

x2=copy.deepcopy(x)      # x2 Deep copy of x


x["hobbies"][0]="Music"

print(x1)    # {'hobbies': ['Music', 'Math']}

print(x2)    # {'hobbies': ['Sport', 'Math']}


x["hobbies"]=["Computer","Swim"]

print(x1)    # {'hobbies': ['Music', 'Math']}

print(x2)    # {'hobbies': ['Sport', 'Math']}
```

What is pip?

A tool for installing and managing Python packages

What is the difference between list and tuples?

LIST	TUPLES
Lists are mutable i.e they can be edited.	Tuples are immutable (tuples are lists which can’t be edited).
Lists are slower than tuples.	Tuples are faster than list.
Syntax: list_1 = [10, ‘Chelsea’, 20]	Syntax: tup_1 = (10, ‘Chelsea’ , 20)

What is a Python module?

A module is a Python script that generally contains import statements, functions, classes and variable definitions, and Python runnable code and it “lives” file with a ‘.py’ extension. zip files and DLL files can also be modules.Inside the module, you can refer to the module name as a string that is stored in the global variable name .

Is Python object oriented? what is object oriented programming?

Yes. Python is Object Oriented Programming language. OOP is the programming paradigm based on classes and instances of those classes called objects. The features of OOP are:

Encapsulation, Data Abstraction, Inheritance, Polymorphism

When Would You Use a List vs. a Tuple vs. a Set in Python?

A list is a common data type that is highly flexible. It can store a sequence of objects that are mutable, so it’s ideal for projects that demand the storage of objects that can be changed later.

A tuple is similar to a list in Python, but the key difference between them is that tuples are immutable. They also use less space than lists and can only be used as a key in a dictionary. Tuples are a perfect choice when you want a list of constants.

Sets are a collection of unique elements that are used in Python. Sets are a good option when you want to avoid duplicate elements in your list. This means that whenever you have two lists with common elements between them, you can leverage sets to eliminate them.

What is lambda in Python?

It is a single expression anonymous function often used as inline function.

Why lambda forms in python does not have statements?

A lambda form in python does not have statements as it is used to make new function object and then return them at runtime.

What is pass in Python?

Pass means, no-operation Python statement, or in other words it is a place holder in compound statement, where there should be a blank left and nothing has to be written there.

What is negative index in Python?

Python sequences can be index in positive and negative numbers. For positive index, 0 is the first index, 1 is the second index and so forth. For negative index, (-1) is the last index and (-2) is the second last index and so forth.

Suppose list1 is [2, 33, 222, 14, 25], What is list1[-1] ?

25

Mention the use of // operator in Python?
It is a Floor Division operator , which is used for dividing two operands with the result as quotient showing only digits before the decimal point. For instance, 10//5 = 2 and 10.0//5.0 = 2.0.

Links:
http://www.pythonchallenge.com/

Threads in python
Memory in python