1. Which data types are supported in Python?

- Primitive Data Structures
  - Integers
  - Float
  - Strings
  - Boolean
- Non-Primitive Data Structures
  - Arrays
  - Lists
  - Tuples
  - Dictionary
  - Sets
  - Files

And classes

2. What is the difference between lists and tuples?

Tuples are recommended to use because they're immutable, which is a better way to handle the state of a variable. Lists are mutable. Tuples have (,) and lists [,]. There is also a difference of how they can be used in sets. For example,

Suppose A and B are lists. The easiest way (but not the fastest) to find an intersection of A and B is 'return set(A) & set(B)'. Another example, set(A) ^ set(B) will give unique elements, which are either only in A or only in B (exclusive or).

Tuples are, in fact, immutable objects, and incidentally it's possible to make a set of tuples: set([(1, 2), (4, 5), (8, 9)])
{(8, 9), (4, 5), (1, 2)}
>>> type(set([(1, 2), (4, 5), (8, 9)]))
as such it will retain the property of having unique but unordered tuples: set([(8, 9), (1, 2), (4, 5), (8, 9)])
{(4, 5), (1, 2), (8, 9)}

You can't have a set of lists: >> set([[1, 2, 3], [5, 6, 7], [9, 10, 11]])
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
TypeError: unhashable type: 'list'

Set element is immutable, while set is not (you can add/remove element)

Set has no duplicates.

3. How is memory managed in Python?

Memory management in Python involves a private heap containing all Python objects and data structures. The management of this private heap is ensured internally by the Python memory manager.

4. Explain Inheritance in Python with an example.

```
# Define a Rectangle class
class Rectangle:
    def __init__(self, h, w):
        self.h=h
        self.w=w


# Define a Square class
class Square(Rectangle):
    def __init__(self, w):
        self.h=w
        self.w=w
```

Class Square inherits from Rectangle.

5. What is a dictionary in Python?

Dictionary example:

```
s="This is a string with words."
# showing how to use a dictionary
import timeit
# check using a dictionary  is time-efficient
start_time = timeit.default_timer()
# code you want to evaluate
```

```
import math
import os
import random
import re
import sys


if __name__ == '__main__':
    words=s.split()
    dict1={} # dict is a reserved word, so we name the first dictionary as dict1
    prev1=prev2=''
    i=0
    for w in words:
        if i>1:
            string=(prev2+' '+prev1+' '+w).lower()
            if string not in dict1:
                dict1[string] = 1 # is key does not exist in a dictionary , it is initialized here, we can add as many keys as we want in a loop
            else:
                dict1[string]+=1 # is the key is present, we can do operation on the value contained is this key


        prev2=prev1
        prev1=w
        i+=1


    keymax = max(dict1, key=dict1.get)  # this is an example of dictionary usage, we can find the maximum value in all keys.
    for key, val in dict1.items(): # this is a way to iterate over all elements in a dictionary and display both the key and the value.
```

```
        if key == keymax:
            print(keymax)
            break

    print(dict1[keymax]) # the key corresponding to the max value has been found


elapsed = timeit.default_timer() - start_time
print(elapsed)
```

6. How will you remove the last object from a list in Python?

lst[:-1]

```
>>> lst=[1, 2, 3, 4, 5]
>>> lst[-1]
5
>>> lst[:-1]
[1, 2, 3, 4]
```

7. What are split(), sub(), and subn() methods in Python?

split() demonstration:

```
 s='This is a sentence.'
>>> s.split()
['This', 'is', 'a', ‘sentence.']
```

For both sub() and subn() the RegEx library needs t be imported :

```
>>> import re
>>> print(re.subn('ov', '~*' , 'movie tickets booking in online'))
('m~*ie tickets booking in online', 1)
>>> t = re.subn('ov', '~*' , 'movie tickets booking in online', flags = re.IGNORECASE)
>>> print(t)
('m~*ie tickets booking in online', 1)
>>> print(len(t))
2
>>> print(t[0])
m~*ie tickets booking in online
>>> t = re.sub('ov', '~*' , 'movie tickets booking in online', flags = re.IGNORECASE)
```

```
>>> t = re.sub('ov', '~*' , 'movie tickets booking in online', flags = re.IGNORECASE)
>>> print(t)
m~*ie tickets booking in online
>>> print(len(t))
31
>>> print(t[0])
m
>>>
```

8. What is a map function in Python?

```
>>> # Python program to demonstrate working
>>> # of map.
>>>
>>> # Return double of n
>>> def addition(n):
...     return n + n
...
>>> # We double all numbers using map()
>>> numbers = (1, 2, 3, 4)
>>> result = map(addition, numbers)
>>> print(list(result))
[2, 4, 6, 8]
```

9. Is indentation optional in Python?

No, in general. On a rare occasion I saw interfaces like CodePad that allow indentations, which are not 4 spaces, as long as indentation is consistent; but it is not a good practice.

10. How are Python arrays and Python lists different from each other?

This is an array:

```
>>> import numpy
>>> x = numpy.array([3, 6, 9, 12])
>>> x/3.0
array([1., 2., 3., 4.])
```

This is a list:

```
>>> y = [3, 6, 9, 12]
>>> y/3.0
Traceback (most recent call last):
```

```
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for /: 'list' and 'float'
>>> print(y)
[3, 6, 9, 12]
```

With a list you can't do list/3.0, but with array array/3.0 is valid.

11. What is __init__ in Python?

It is a class constructor like so:

```
class car():

    # init method or constructor
    def __init__(self, model, color):
        self.model = model
        self.color = color

    def show(self):
        print("Model is", self.model )
        print("color is", self.color )
```

12. What is lambda function in Python?

A lambda expression defines a function:

```
>>> lambda x: x + 1
<function <lambda> at 0x7f7efed879d0>
>>> (lambda x: x + 1)(2)
3
>>> f=lambda x: x + 1
>>> f(2)
3
```

13. What is self-keyword in Python?

In the car class defined above, self refers to a class instance. For example,

```
class car():

    # init method or constructor
    def __init__(self, model, color):
        self.model = model
        self.color = color
```

```
    def show(self):
        print("Model is", self.model )
        print("color is", self.color )

# both objects have different self which
# contain their attributes
audi = car("audi a4", "blue")
ferrari = car("ferrari 488", "green")
```

14. What is the difference between append() and extend() methods?

```
>>> l=[]
>>> l
[]
>>> l.append(10)
>>> l
[10]
>>> k=[]
>>> k
[]
>>> k.extend(10)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'int' object is not iterable
>>>
```

When append() method adds its argument as a single element to the end of a list, the length of the list itself will increase by one. Whereas extend() method iterates over its argument adding each element to the list, extending the list.

This will work for ‘extend’:

```
>>> x = [1, 2, 3]
>>> x.extend([4, 5])
>>> print (x)
[1, 2, 3, 4, 5]
```