1. **What are the two major sequence types in Python?**

* **LIST**
* **TUPLES**
* **DICTIONARIES**

1. **How list indices work in python explain with example**

The list is a most versatile datatype available in Python which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list is that items in a list need not be of the same type.

Creating a list is as simple as putting different comma-separated values between square brackets. For example –

1. list1 = ['physics', 'chemistry', 1997, 2000];
2. list2 = [1, 2, 3, 4, 5 ];
3. list3 = ["a", "b", "c", "d"];

**3. What is the basic difference between list and tuple**

**Lists**  
  
Lists are what they seem - a list of values. Each one of them is numbered, starting from zero - the first one is numbered zero, the second 1, the third 2, etc. You can remove values from the list, and add new values to the end.  
  
**Example:** Your many cats' names.  
  
[code]  
"A list"  
cats = ['Tom', 'Snappy', 'Kitty', 'Jessie', 'Chester']  
"If you want to print for example the first name on the list you would do the following"  
print cats[0]  
"Also if you want to add a name to the list you would do this"  
cats.append('Catherine')  
"To remove a name you would do this"  
del cats[0]  
[/code]

**Tuples**  
  
Tuples are just like lists, but you can't change their values. The values that you give it first up, are the values that you are stuck with for the rest of the program. Again, each value is numbered starting from zero, for easy reference.  
  
**Example:** the names of the months of the year.  
  
[code]  
"Here is an example of a tuple"  
months = ('January','February','March','April','May','June',\  
'July','August','September','October','November',' December')  
[/code]  
  
"So that’s the difference between lists and tuples. A list can be modified but a tuple cannot be modified in anyway(unless changed in the source code of the program)"

**4. How do you convert a tuple to a list?**

tuple1 = (1,2,3,'HIREN')

print(tuple1)

list1 = list(tuple1)

print(list1)

Output 🡪 (1, 2, 3, 'HIREN')

[1, 2, 3, 'HIREN']

**5. Give any 3 comparison between dictionary and list**

List is like array, it can be used to store homogeneous as well as heterogeneous data type (It can store same data type as well as different data type). List are faster compared to array. Individual element of List data can be accessed using indexing & can be manipulated.

**List Code Snippet:**  
list = ["Sarah",29,30000.00]  
for i in range (3):  
     print list[i]  
------------------  
**Output**Sarah , 29 , 30000.00

1. **Which statement is used to remove (key,value) in a dictionary**

You can either remove individual dictionary elements or clear the entire contents of a dictionary. You can also delete entire dictionary in a single operation.

To explicitly remove an entire dictionary, just use the **del** statement. Following is a simple example −

#!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'};

del dict['Name']; # remove entry with key 'Name'

dict.clear(); # remove all entries in dict

del dict ; # delete entire dictionary

print "dict['Age']: ", dict['Age']

print "dict['School']: ", dict['School']

1. **Give any 3 operations of list**

*l.*append (*object*)

Update list *l*by appending *object*to end of the list.

*l.*extend (*list*)

Extend list *l*by appending *list*elements. Note the difference from append(*object*), which treats the argument as a single list object.

*l.*insert (*index*, *object*)

1. **Give any 3 operations of list**

|  |  |
| --- | --- |
|  |  |
| 1 | [**dict.clear()**](http://www.tutorialspoint.com/python/dictionary_clear.htm)  Removes all elements of dictionary *dict* |
| 2 | [**dict.copy()**](http://www.tutorialspoint.com/python/dictionary_copy.htm)  Returns a shallow copy of dictionary *dict* |
| 3 | [**dict.fromkeys()**](http://www.tutorialspoint.com/python/dictionary_fromkeys.htm)  Create a new dictionary with keys from seq and values *set* to *value*. |

**9. Give any 3 operations of tuple**

Methods that add items or remove items are not available with tuple. Only the following two methods are available.

|  |  |
| --- | --- |
| Python Tuple Method | |
| Method | Description |
| count(*x*) | Return the number of items that is equal to *x* |
| index(*x*) | Return index of first item that is equal to*x* |

>>> my\_tuple = ('a','p','p','l','e',)

>>> my\_tuple.count('p')

2

>>> my\_tuple.index('l')

3

**10. What is the use of map function explain it with an example**

def fahrenheit(T):

return ((float(9)/5)\*T + 32)

def celsius(T):

return (float(5)/9)\*(T-32)

temp = (36.5, 37, 37.5,39)

temp = (36.5, 37, 37.5, 38, 39)

F = map(fahrenheit, temp)

C = map(celsius, F)

a=list(map(fahrenheit, temp))

print(a)

b=list(map(celsius, F))

print(b)

**12. Which python function is used to traverse the directory structure recursively?**

* dirName: The next directory it found.
* subdirList: A list of sub-directories in the current directory.
* fileList: A list of files in the current directory.

**13. What is pickle? How it is used in python**

 It is used for serializing and de-serializing a Python object structure. Any object in python can be pickled so that it can be saved on disk. What pickle does is that it “serialises” the object first before writing it to file. Pickling is a way to convert a python object (list, dict, etc.) into a character stream. The idea is that this character stream contains all the information necessary to reconstruct the object in another python script.

[1](https://wiki.python.org/moin/UsingPickle#CA-4432c9c973c553ebd06665a3edc404c23544eb46_1) # Save a dictionary into a pickle file.

[2](https://wiki.python.org/moin/UsingPickle#CA-4432c9c973c553ebd06665a3edc404c23544eb46_2) import pickle

[3](https://wiki.python.org/moin/UsingPickle#CA-4432c9c973c553ebd06665a3edc404c23544eb46_3)

[4](https://wiki.python.org/moin/UsingPickle#CA-4432c9c973c553ebd06665a3edc404c23544eb46_4) favorite\_color = { "lion": "yellow", "kitty": "red" }

[5](https://wiki.python.org/moin/UsingPickle#CA-4432c9c973c553ebd06665a3edc404c23544eb46_5)

[6](https://wiki.python.org/moin/UsingPickle#CA-4432c9c973c553ebd06665a3edc404c23544eb46_6) pickle.dump( favorite\_color, open( "save.p", "wb" ) )

[Toggle line numbers](https://wiki.python.org/moin/UsingPickle)

[1](https://wiki.python.org/moin/UsingPickle#CA-10a60777cc6a4f82f809e00de9efde0d29c23f9f_1) # Load the dictionary back from the pickle file.

[2](https://wiki.python.org/moin/UsingPickle#CA-10a60777cc6a4f82f809e00de9efde0d29c23f9f_2) import pickle

[3](https://wiki.python.org/moin/UsingPickle#CA-10a60777cc6a4f82f809e00de9efde0d29c23f9f_3)

[4](https://wiki.python.org/moin/UsingPickle#CA-10a60777cc6a4f82f809e00de9efde0d29c23f9f_4) favorite\_color = pickle.load( open( "save.p", "rb" ) )

1. [5](https://wiki.python.org/moin/UsingPickle#CA-10a60777cc6a4f82f809e00de9efde0d29c23f9f_5) # favorite\_color is now { "lion": "yellow", "kitty": "red" }

**14. What is the use of \_\_init\_\_method in python?**

There are many method names which have special significance in Python classes. We will see the significance of the \_\_init\_\_ method now.

The \_\_init\_\_ method is run as soon as an object of a class is instantiated. The method is useful to do any *initialization* you want to do with your object. Notice the double underscore both in the beginning and at the end in the name.

**Using the \_\_init\_\_ method**

**ExampleÂ 11.3.Â Using the \_\_init\_\_ method**

#!/usr/bin/python

# Filename: class\_init.py

class Person:

def \_\_init\_\_(self, name):

self.name = name

def sayHi(self):

print 'Hello, my name is', self.name

p = Person('Swaroop')

p.sayHi()

# This short example can also be written as Person('Swaroop').sayHi()

**Output**

$ python class\_init.py

Hello, my name is Swaroop

**16. What is the difference between static method and class method in python?**

Static Method

- Simple functions with no self arguments

-nested inside class

-work on class attributes; not on instance attributes

-can be called through both class and instance.

-built-in function staticmethod()is used to create them

-classes can inherit the static method without redefining it

Example for,

class spam(object):

numi=0

def \_\_init\_\_(self):

spam.numi = spam.numi + 1

def pn():

print("created",spam.numi)

pn = staticmethod(pn)

x=spam()

y=spam()

spam.pn()

x.pn()

Class method:

functions that have first argument as class name

can be called through both class and instance

These are created with classmethod inbuilt function

These always receive the lowest class in an instance's tree

class Employee:

'Common base class for all employees'

empCount = 0

def \_\_init\_\_(self, name, salary):

self.name = name

self.salary = salary

Employee.empCount += 1

def displayCount(self):

print "Total Employee %d" % Employee.empCount

def displayEmployee(self):

print "Name : ", self.name, ", Salary: ", self.salary

"This would create first object of Employee class"

emp1 = Employee("Zara", 2000)

"This would create second object of Employee class"

emp2 = Employee("Manni", 5000)

emp1.displayEmployee()

emp2.displayEmployee()

print "Total Employee %d" % Employee.empCount