**1. What is the difference between list and tuples in Python**

Lists are mutable, therefore they can be assigned a new value. whereas tuples are immutable, they cannot be assign new value:

Example :

l1 = [1,2,3,4,5]

l1[1] = 1000 # Assigning a new value at position 1

print(l1)

tuple1 = (1,2,3)

tuple1[2] = 100 # Error, tuple object does not support item assiggment

Lists are slower than tuples. whereas tuples are faster then lists.

**2. How tuples are faster than lists ?**

**Tuples** are stored in a single block of memory and it is immutable, therefore It does not require extra space to store new objects.

**Lists** are allocated in two blocks: the fixed one with all the Python object information and a variable sized block for the data.Hence it is the reason why tuples are faster than lists.

**3. What is the use of property decorator ?**

The property decorator makes the methods that act like an attribute to the python class.

**4. What is Pep8 in Python?**

PEP Stands for **Python Enhancement Proposal.** It is a set of rules that specify how to format python code in order to maximize readability of python code. Click on link to read more about it. <https://www.python.org/dev/peps/pep-0008/>

**5. What  do you understand by  PYTHONPATH ?**

It is an environment variable which is used when a module is imported. Whenever a module is imported, PYTHONPATH is also looked up to check for the presence of the imported modules in various directories. The interpreter uses it to determine which module to load.

**6. What is type conversion in Python?**

**int()** – converts any data type into integer type

**float()** – converts any data type into float type

**ord()** – converts characters into integer

**hex(**) – converts integers to hexadecimal

**oct()** – converts integer to octal

**tuple() –**This function is used to convert to a tuple.

**set() –**This function returns the type after converting to set.

**list() –**This function is used to convert any data type to a list type.

**dict() –**This function is used to convert a tuple of order (key,value) into a dictionary.

**str() –**Used to convert integer into a string.

**complex(real,imag) –** This functionconverts real numbers to complex(real,imag) number.

**7. What is a lambda function ?**

In python, an anonymous function is known as a lambda function. This function can have any number of parameters but, can have just one statement. Moreover, Lambda  function does not include return statement ,it always contain expression which is returned.

Syntax:

**a = lambda x,y : x+y**

**print(a(100,200))**

**8. What is Filter and Map function ? how they are differ from each other ?**

* The **filter()** function in Python takes  a function and a list as argument. This offers an outstanding way to filter out all the elements of a sequence, for which the function returns **True**. Below is the example of filtering even number.

**Example:**

**l1 = [1,2,10,9,20,32,16,63]**

**filteredList = list(filter(lambda x : (x%2 == 0),l1))**

**print(filteredList) # [2, 10, 20, 32, 16]**

* The **map()** function in Python takes a function and a list as an argument. A new list is returned which contains all the modified items returned by that function for each item.

**Example:**

**li = [1,2,3,4]**

**final\_list = list(map(lambda x: x\*2 , li))**

**print(final\_list) # [2,4,6,8]**

* Filter can only filter list of item for which function returns true, however map is used to modify the given list of Items.
* Filter takes only single list as an argument whereas map takes one or more argument as a list

### ****9. What is pickling and unpickling?****

Pickle module accepts any Python object and converts it into a string representation and dumps it into a file by using dump function, this process is called pickling. While the process of retrieving original Python objects from the stored string representation is called unpickling

### ****10. What are the generators in python?****

A generator-function is defined like a normal function, but whenever it needs to generate a value, it does so with the [**yield Keyword**](https://www.geeksforgeeks.org/use-yield-keyword-instead-return-keyword-python/)rather than return. If the body of a def contains yield, the function automatically becomes a generator function. In simple words, Functions that return an iterable set of items are called generators.

**def simpleGeneratorFun():**

**yield 1**

**yield 2**

**yield 3**

**# Driver code to check above generator function**

**for value in simpleGeneratorFun():**

**print(value)**

### ****11. What is the purpose of is, not and in operators?****

Operators are special functions. They take one or more values and produce a corresponding result.

**is**: returns true when 2 operands are true  (Example: “a” is ‘b’)

**not**: returns the inverse of the boolean value

**in**: this operator is used to check whether an item is present in sequence or not

### ****12. What does this mean: \*args, \*\*kwargs? And why would we use it?****

 \***args** is used when we aren’t sure how many arguments are going to be passed to a function, or if we want to pass a pass list or tuple of arguments to a function. \*\***kwargs** is used when we don’t know how many keyword arguments will be passed to a function, or it can be used to pass the values of a dictionary as keyword arguments

The identifiers **args** and **kwargs** are a convention, we could also use \***s** and \*\***srg** but that is not advisable

**Note**: While defining a function we can add other parameters too along with \***args**. But we need to make sure that parameter \***args** should always be after formal arguments

### ****13. What is the difference between deep and shallow copy?****

### ****Shallow copy doesn't create a copy of nested objects, instead it just copies the reference of nested objects or we cans say that shallow copy just share the reference of original object. While A deep copy creates a new object and recursively adds the copies of nested objects present in the original elements.****

**# Example:**

**import copy**

**l1 = [[1,2],[3,4],[5,6]]**

**l2 = copy.copy(l1) # Shallow copy**

**print(l1)**

**print(l2)**

**# Output**

**[[1, 2], [3, 4], [5, 6]]**

**[[1, 2], [3, 4], [5, 6]]**

**l1[0][0] = 10000**

**print(l1)**

**print(l2)**

**# Output:**

[**[10000, 2], [3, 4], [5, 6]]**

**[[10000, 2], [3, 4], [5, 6]]**

**# Example of deepcopy:**

**l3 = [[1,2],[3,4],[5,6]]**

**l4 = copy.deepcopy(l3)**

**print(l3)**

**print(l4)**

**[[1, 2], [3, 4], [5, 6]]**

**[[1, 2], [3, 4], [5, 6]]**

**l3[0][0] = 10000**

**print(l3)**

**print(l4)**

**# output**

**[[10000, 2], [3, 4], [5, 6]]**

**[[1, 2], [3, 4], [5, 6]]**

**14. What do you know about the list/dictionary comprehensions?**

List/Dictionary comprehensions is an elegant way to create or define new list based od existing list. Moreover, List/Dictionary comprehensions are faster than standard loop.

**Syntax:**

**list\_val = [ x for x in iretable]**

**Example:**

**List1 = [x for x in range(1,10)]**

**15. What is a Python module?**

Modules are independent Python scripts with the .py extension that can be reused in other Python codes or scripts using the import statement. A module can consist of functions, classes, and variables, or some runnable code. Modules not only help in keeping Python codes organized but also in making codes less complex and more efficient

**Explain the use of the 'with' statement.**

In Python, using the ‘with’ statement, we can open a file and close it as soon as the block of code, where ‘with’ is used, exits. In this way, we can opt for not using the close() method.

**Syntax:**

**with open("filename", "mode") as file\_var:**

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

Both methods are used to add elements at the end of the list.

* **append(item)**: Adds the given element at the end of the list.
* **extend(another-list)**: Adds the elements of another list at the end of the list that called this extend() method.

**Example:**

**# Appending an element in the list**

**list1 = [1,2,3,4,5]**

**print(list1)**

**list1.append(100)**

**print(list1)**

**Output:**

**[1, 2, 3, 4, 5]**

**[1, 2, 3, 4, 5, 100]**

**# Extending list**

**list1 = [1,2,3,4,5]**

**list2 = [10,20,30]**

**print('list1: ',list1)**

**print('list2: ',list2)**

**list1.extend(list2)**

**print("After exteding:",list1)**

**Output:**

**list1: [1, 2, 3, 4, 5]**

**list2: [10, 20, 30]**

**After exteding: [1, 2, 3, 4, 5, 10, 20, 30]**

### ****17. What is self in Python?****

Self is a keyword in Python used to define an instance or an object of a class. In Python, it is explicity used as the first parameter. It helps in distinguishing between the methods and attributes of a class from its local variables.

### ****18. What is \_\_init\_\_?****

\_\_init\_\_ is a contructor method in Python and is automatically called to allocate memory when a new object/instance is created. All classes have a \_\_init\_\_ method associated with them.

### **What is pickling and unpickling?**

Pickle module accepts any Python object and converts it into a string representation and dumps it into a file by using dump function, this process is called pickling. While the process of retrieving original Python objects from the stored string representation is called unpickling

### ****What is slicing in Python?****

Slicing in python is an elegant way to retrieve part of the sequence/list. In simple word, getting element from lists/array without using any loop.

Syntax:

**[ start : stop : step ]**

**Start**: is the starting index

**Stop**: index where to stop

**Step** : number of step to jump

Exmaple:

l1 = [1,2,3,4,5,6,7,8,9,10]

print(l1[2:9:2]) # Start from index 2,end at index 9 and jump two steps

Output:

[3, 5, 7, 9 ]