**Python Interview Question**

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

|  |  |
| --- | --- |
| **List** | **Tuple** |
| Lists are mutable | Tuples are immutable |
| Lists are slower than tuples. | Tuples are faster than list. |
|  |  |

* **Tuples** are stored in a single block of memory. Tuples are immutable so, It doesn't 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.
* It is the reason creating a ***tuple is faster than List.***
* It also explains the slight difference in indexing speed is faster than lists, because in tuples for indexing it follows fewer pointers.

**Uses of property decorator.**

 The decorator "property" makes the method that acts like an attribute to the python class. "

### ****What is pep 8****

PEP stands for **Python Enhancement Proposal.**It is a set of rules that specify how to format Python code for maximum readability

### ****How is memory managed in Python?****

1. Memory management in python is managed by ***Python private heap space***. All Python objects and data structures are located in a private heap. The programmer does not have access to this private heap. The python interpreter takes care of this instead.
2. The allocation of heap space for Python objects is done by Python’s memory manager. The core API gives access to some tools for the programmer to code.
3. Python also has an inbuilt garbage collector, which recycles all the unused memory and so that it can be made available to the heap space.

### ****What is namespace in Python****

A namespace is a naming system used to make sure that names are unique to avoid naming conflicts.

### ****What is 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.

### ****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.

### ****What is a lambda function?****

An anonymous function is known as a lambda function. This function can have any number of parameters but, can have just one statement.

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

**print(a(5, 6))**

**Using Lambda :** Lambda definition does not include a “return” statement, it always contains an expression which is returned. We can also put a lambda definition anywhere a function is expected, and we don’t have to assign it to a variable at all. This is the simplicity of lambda functions.

**Use of lambda() with filter()**

The filter() function in Python takes in a function and a list as arguments. This offers an elegant way to filter out all the elements of a sequence “sequence”, for which the function returns True. Here is a small program that returns the odd numbers from an input list:

# Python code to illustrate

# filter() with lambda()

li = [5, 7, 22, 97, 54, 62, 77, 23, 73, 61]

final\_list = list(filter(lambda x: (x%2 != 0) , li))

print(final\_list)

**Use of lambda() with map()**

The map() function in Python takes in a function and a list as argument. The function is called with a lambda function and a list and a new list is returned which contains all the lambda modified items returned by that function for each item. Example:

# Python code to illustrate

# map() with lambda()

# to get double of a list.

li = [5, 7, 22, 97, 54, 62, 77, 23, 73, 61]

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

print(final\_list)

**filter vs map**

filter return only true/ false so we can only filter the list,if we do

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

list(filter(lambda x : x+1,l1)) # It will not add because all are true. How ever map will add it.

Filter take only single list as an argument, whereas map takes one or more list of argument.

list(filter(lambda x,y : x+y,l1,l2)) # Error

list(Map(lambda x,y : x+y,l1,l2)) # okay

### ****What are python iterators?****

Iterators are objects which can be traversed though or iterated upon.

### ****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 are the generators in python?****

Functions that return an iterable set of items are called generators.

**Generator-Function :**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.

def simpleGeneratorFun():

    yield 1

    yield 2

    yield 3

# Driver code to check above generator function

for value in simpleGeneratorFun():

    print(value)

### ****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 ‘a’)

not: returns the inverse of the boolean value

in: checks if some element is present in some sequence

### ****What is the usage of help() and dir() function in Python?****

Help() and dir() both functions are accessible from the Python interpreter and used for viewing a consolidated dump of built-in functions.

1. Help() function: The help() function is used to display the documentation string and also facilitates you to see the help related to modules, keywords, attributes, etc.
2. Dir() function: The dir() function is used to display the defined symbols.

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

We use \*args when we aren’t sure how many arguments are going to be passed to a function, or if we want to pass a stored 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, you could also use \*bob and \*\*billy but that would not be wise.

**Positioning of parameter \*args**

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

### ****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]]

### What is the function to randomize the items of a list in-place?

### Python has a built-in module called as <random>. It exports a public method <shuffle(<list>)> which can randomize any input sequence.

### What is the best way to split a string in Python?

### We can use Python <split()> function to break a string into substrings based on the defined separator. It returns the list of all words present in the input string.

### What is the right way to transform a Python string into a list?

### In Python, strings are just like lists. And it is easy to convert a string into the list. Simply by passing the string as an argument to the list would result in a string-to-list conversion.

list("I am learning Python.")

### Exception handling clausing:

### try-except-finally ****2.**** try-except-else

### What do you know about the <list> and <dict> comprehensions? Explain with an example.

### The <List/Dict> comprehensions provide an easier way to create the corresponding object using the existing iterable. As per official Python documents, the list comprehensions are usually faster than the standard loops. But it’s something that may change between releases.

### Can you write code to determine the name of an object in Python?

### No objects in Python have any associated names. So there is no way of getting the one for an object. The assignment is only the means of binding a name to the value.

### Can you write code to check whether the given object belongs to a class or its subclass.

### Python has a built-in method to list the instances of an object that may consist of many classes. It returns in the form of a table containing tuples instead of the individual classes. Its syntax is as follows:

<isinstance(obj, (class1, class2, ...))>

### How would you produce a list with unique elements from a list with duplicate elements?

duplicates = ['a','b','c','d','d','d','e','a','b','f','g','g','h']

uniqueItems = list(set(duplicates))

print sorted(uniqueItems)

### ****What is the purpose of PYTHONPATH environment variable?****

PYTHONPATH has a role similar to PATH. This variable tells Python Interpreter where to locate the module files imported into a program. It should include Python source library directory and the directories containing Python source code. PYTHONPATH is sometimes preset by Python Installer.

### ****What is the purpose of PYTHONSTARTUP, PYTHONCASEOK, and PYTHONHOME environment variables?****

* **PYTHONSTARTUP**: It contains the path of an initialization file having Python source code. It is executed every time we start the interpreter. It is named as .pythonrc.py in Unix, and it contains commands that load utilities or modify PYTHONPATH.
* **PYTHONCASEOK**: It is used in Windows to instruct Python to find the first case-insensitive match in an import statement. We can set this variable with any value to activate it.
* **PYTHONHOME**: It is an alternative module search path. It is usually embedded in PYTHONSTARTUP or PYTHONPATH directories to make switching of module libraries easy.

### ****What is a map function in Python?****

The **map()** function in Python has two parameters, function and iterable. The map() function takes a function as an argument and then applies that function to all the elements of an iterable, passed to it as another argument. It returns an object list of results.

For example:

def calculateSq(n):

return n\*n

numbers = (2, 3, 4, 5)

result = map(calculateSq, numbers)

print(result)

### ****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 and its syntax.****

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.

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

### ****What is \_\_init\_\_ in Python?****

Equivalent to constructors in OOP terminology, \_\_init\_\_ is a reserved method in Python classes. The \_\_init\_\_ method is called automatically whenever a new object is initiated. This method allocates memory to the new object as soon as it is created. This method can also be used to initialize variables

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

Both append() and extend() methods are methods used to add elements at the end of a list.

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

# **Decorators in Python**

In Python, functions are the first class objects, which means that –

* Functions are objects; they can be referenced to, passed to a variable and returned from other functions as well.
* Functions can be defined inside another function and can also be passed as argument to another function.
* [Decorators](https://www.geeksforgeeks.org/function-decorators-in-python-set-1-introduction/) are very powerful and useful tool in Python since it allows programmers to modify the behavior of function or class. Decorators allow us to wrap another function in order to extend the behavior of wrapped function, without permanently modifying it.
* In Decorators, functions are taken as the argument into another function and then called inside the wrapper function.

**def check(func):**

**def divCheck(a, b):**

**if b == 0:**

**print("Can not be divided")**

**return**

**return func(a, b)**

**return divCheck**

**@check**

**def divide1(a,b):**

**return a/b**

**divide1(5,0)**