Ques. How to get Id()?

id() function takes a single parameter object.

```
a = 5
print(id(a))
```

Ques. a=1, b=1 does both have same Id or not?

Two variables in Python have same id, but not in lists.

```
a = 10
b = 10
print(id(a))
print(id(b))

Output:-
9788992
9788992
---------------------------------------------
# In List
a = [1, 2, 3]
b = [1, 2, 3]
print(a is b)

Output:- False
---------------------------------------------------
# In tuples
a = (1, 2, 3)
b = (1, 2, 3)
print(a is b)

Output:- True
```

Ques. What is Generator Functions?

- Generator are functions that returns a sequens of value. we use **yield** statement to return the from function.
- Yield statement returns the element from a generator function into a genrater object.(EX:- yield a)
- This function is used to retrieve element by element from a generator object.(Ex:- next(gen_obj))
- A generator is a special type of function which does not return a single value, instead, it returns an iterator object with a sequence of values.
- In a generator function, a **yield** statement is used rather than a return statement.
- The generator function cannot include the return keyword. If you include it, then it will terminate the function.

- The **difference** between **yield** and **return** is that yield **returns a value and pauses the execution** while maintaining the internal states, whereas the **return statement returns a value and terminates the execution** of the function.

```
def mygenerator():
    print('First item')
    yield 10

    print('Second item')
    yield 20

    print('Last item')
    yield 30

gen = mygenerator()
print(next(gen))
print(gen.__next__())  # 2 option to write the next function
print(next(gen))
print(next(gen))

Output:-
First item
10
Second item
20
Last item
30
Traceback (most recent call last):
File "<string>", line 22, in <module>
StopIteration

---------------------------------------------------------
# 2nd Option
gen = mygenerator()
while True:
    try:
        print ("Received on next(): ", next(gen))
    except StopIteration:
        break
Output:-
First item
Received on next():  10
Second item
Received on next():  20
Last item
Received on next():  30

# Example 2:-
def bhai(a,b):
    yield a+b
    yield a-b
result = bhai(3,2)
```

```
print(next(result))
print(next(result))

Output:-
5
1

# Example 3:-
def numberPrint():
    n = 1
    while n <= 10:
        sq = n*n
        yield sq
        n += 1
values = numberPrint()
for i in values:
    print(i)

Output:-
1
4
9
16
25
36
49
64
81
100
```

Ques. What are global, protected and private attributes in Python?

- **Global variables** are public variables that are defined in the global scope. To use the variable in the global scope inside a function, we use the **global** keyword.
- **Protected attributes** are attributes defined with an **single underscore** prefixed to their identifier eg. _mohit. They can still be accessed and modified from outside the class they are defined in but a responsible developer should refrain from doing so.
- **Private attributes** are attributes with **double underscore** prefixed to their identifier eg. __mohit. They cannot be accessed or modified from the outside directly and will result in an AttributeError if such an attempt is made.

Python Collections (Arrays)?

There are four collection data types in the Python programming language:

- **List** is a collection which is ordered and changeable. Allows duplicate members.
- **Tuple** is a collection which is ordered and unchangeable. Allows duplicate members.
- **Set** is a collection which is unordered and unindexed. No duplicate members.
- **Dictionary** is a collection which is unordered, changeable and indexed. No duplicate members.

Ques. How would you convert a list to an array?

- Python list is a linear data structure that can hold heterogeneous elements. Python does not have a built-in array data type. If you want to create an array in Python, then use the numpy library.
- To *install numpy* in your system, type the following command.
- python3 -m pip install numpy
- **1. Using numpy.array()**

```
import numpy as np

elon_list = [11, 21, 19, 18, 29]
elon_array = np.array(elon_list)

print(elon_array)
print(type(elon_array))

Output:-
[11 21 19 18 29]
<class 'numpy.ndarray'>

# 2. Using numpy.asarray()
import numpy as np

elon_list = [11, 21, 19, 18, 29]
elon_array = np.asarray(elon_list)

print(elon_array)
print(type(elon_array))

Output:- [11 21 19 18 29]
<class 'numpy.ndarray'>
```

Ques. What do * (single asterisk) and ** (double asterisk) do for parameters in Python?

Ques. What is * args and ** kwargs in Python?

1. *args (Non Keyword Arguments)
2. **kwargs (Keyword Arguments)

- **Single Asterisk:-** Single Asterisk allows the developer to pass a variable number of Positional parameters and automatically converts the input values in the form of tuples. At the same time.

```
def print_colors(*args):
    print(args)
print_colors('red','blue','green','yellow')

Output:- ('red', 'blue', 'green', 'yellow')
```

- **Double Asterisks** Double Asterisk allows the users to pass a variable number of Keyword parameters in the form of a Dictionary.

```
def print_numbers(**kwargs):
  for key, value in kwargs.items():
      print (f"{key} is a {value}")
print_numbers(mohit="TL", two="two",three=3,four="four")

Output:-
mohit is a TL
two is a two
three is a 3
four is a four
```

Ques. What is built-in module in Python?

https://docs.python.org/3/py-modindex.html
Example

```
>>> import html
>>> import html.parser
import mysql.connector
```

ye buil in packege hote hai
agar or karne hai to pip ki help sa karenge

Ques. How is memory managed in Python?

- Memory management in Python is handled by the **Python Memory Manager**.
- 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.
- 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.
- 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.

Ques. What is Map, Filter, Reduce?

sdafasf fasf das f

Ques. What is an Iterable?

- An Iterable is an object that implements the **iter**() method and returns an iterator object or an object that implements **getitem**() method (and should raise an IndexError when indices are exhausted).

What is Iterators?

- iterator obj print the value one by one.
- An iterator is an object that contains a countable number of values.

- An iterator is an object that can be iterated upon, meaning that you can traverse through all the values.
- Technically, in Python, an iterator is an object which implements the iterator protocol, which consist of the methods **iter**() and **next**().

```
# Example1
mytuple = ("apple", "banana", "cherry")
myit = iter(mytuple)

print(next(myit))
print(next(myit))
print(next(myit))

Output:-
apple
banana
cherry

# Example 2
mystr = "banana"
myit = iter(mystr)

print(next(myit))
print(next(myit))
print(next(myit))
print(next(myit))
print(next(myit))
print(next(myit))

Output:-
b
a
n
a
n
a

# Looping Through an Iterator
# The for loop actually creates an iterator object and executes the next() method
for each loop
mytuple = ("apple", "banana", "cherry")

for x in mytuple:
  print(x)

Output:-
apple
banana
cherry
```

Ques. Difference between Iterators and iterable?

- Every iterator is also an iterable, but not every iterable is an iterator.

Iterable	**Iterator**
An Iterable is basically an object that any user can iterate over.	An Iterator is also an object that helps a user in iterating over another object (that is iterable).
We can generate an iterator when we pass the object to the iter() method.	We use the **next**() method for iterating. This method helps iterators return the next item available from the object.
Every iterator is basically iterable.	Not every iterable is an iterator.

Ques. Difference between generator and iterators in python?

Iterator	**Generator**
Class is used to implement an iterator	Function is used to implement a generator.
Local Variables aren't used here.	All the local variables before the yield function are stored.
Iterators are used mostly to iterate or convert other objects to an iterator using iter() function.	Generators are mostly used in loops to generate an iterator by returning all the values in the loop without affecting the iteration of the loop
Iterator uses iter() and next() functions	Generator uses yield keyword
Every iterator is not a generator	Every generator is an iterator

String to array:-

```
thislist = ["apple", "banana", "cherry"]
str1 = ' '.join(thislist)
counter = dict.fromkeys(str1, 0)
print(counter)
for item in str1:
    counter[item] += 1
print(counter)

import collections
print(collections.Counter("hello"))
```

Use of “get()” function

Ques. What is NumPy?

- NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

Ques. Why is NumPy Faster Than Lists?

Ques. What is difference betweenr repr() and str()?

the built-in str() and repr() functions both produce a textual representation of an object.

- **str():-** The str() function returns a user-friendly description of an object.
- **repr():-** The repr() method returns a developer-friendly string representation of an object.

```
import datetime
today = datetime.datetime.now()
print(str(today))
print(repr(today))

Output:-
2021-08-14 08:18:25.138663
datetime.datetime(2021, 8, 14, 8, 18, 25, 138663)
```

```
class Fruit:
    def __init__(self, name):
        self.name = name

banana = Fruit("Banana")
print(banana)

Output:- <__main__.Fruit object at 0x7f97c77704c0>
-----------------------------------------------------
class Fruit:
    def __init__(self, name):
        self.name = name

    def __str__(self):
        return f'I am a {self.name}'

banana = Fruit("Banana")
print(banana)

Output:- I am a Banana
```

Ques. What is Scope Resolution in Python?

- scope resolution in python follows the LEGB rules.

1. Local(L): Defined inside function/class
2. Enclosed(E): Defined inside enclosing functions(Nested function concept)
3. Global(G): Defined at the uppermost level
4. Built-in(B): Reserved names in Python builtin modules

- Local, Enclosed, and Global Scopes in Python

```
pi = 'global pi variable'

def outer():
    pi = 'outer pi variable'
    def inner():
        # pi = 'inner pi variable'
        nonlocal pi
        print(pi)
    inner()

outer()
print(pi)

Output:-
outer pi variable
global pi variable
```

What is docstring in Python?

What are Python namespaces?

A namespace in python refers to the name which is assigned to each object in python. The objects are variables and functions. As each object is created, its name along with space(the address of the outer function in which the object is), gets created. The namespaces are maintained in python like a dictionary where the key is the namespace and value is the address of the object. There 4 types of namespace in python-

Built-in namespace– These namespaces contain all the built-in objects in python and are available whenever python is running. Global namespace– These are namespaces for all the objects created at the level of the main program. Enclosing namespaces– These namespaces are at the higher level or outer function. Local namespaces– These namespaces are at the local or inner function.

```
a = "mohit kumar"
new_list = [1,2,3,4,3,2,1,5,6,7]
list4 = []
for item in a:
    list4.append(item)
print(list4)
list5 = []
[list5.append(item*5) for item in new_list if item not in list5]
print(list5)
dict1 = {}
# for item in List:
#   dict1[len(item)] = item
# print(dict1)
dict1 = {len(item): item  for item in List}
print(dict1)
abc = sorted(dict1)
list6 = [dict1[item] for item in abc]
print(list6)
```

Unique dictionary from list?

```
a = [
{'id':1,'name':'john', 'age':34},
{'id':1,'name':'john', 'age':34},
{'id':2,'name':'hanna', 'age':30},
]

b = {x['id']:x for x in a}.values()

print(b)

Output:-
dict_values([{'id': 1, 'name': 'john', 'age': 34}, {'id': 2, 'name': 'hanna',
'age': 30}])
```

Ques. What is the purpose of the range() function?

- The range() function in Python is used to generate a sequence of numbers, which is often used for iterating over a loop a specific number of times.
- The range() function can take up to three arguments:
  - start (optional): The starting value of the sequence (default is 0).
  - stop: The stopping value of the sequence. The sequence will go up to, but not include, this value.
  - step (optional): The difference between each number in the sequence (default is 1).

```
# Using range with only one argument (stop)
for i in range(5):
    print(i)  # Prints 0, 1, 2, 3, 4

# Using range with two arguments (start and stop)
for i in range(2, 8):
    print(i)  # Prints 2, 3, 4, 5, 6, 7

# Using range with all three arguments (start, stop, step)
for i in range(1, 10, 2):
    print(i)  # Prints 1, 3, 5, 7, 9
```

Ques. How do you open and close a file in Python?

```
file = open("example.txt", "r")  # Opens the file in read mode
content = file.read()  # Read the content of the file
print(content)
file.close()  # Close the file when done
```

g. WAP to count from a string? l. Count occurrence of each number in list. s. Write a custom insert() function for list eg. def custom_insert (list, index, item). bb. Can we pass list or tuple as a key in dictionary? ee. What is property decorator? ii. What is lazy evaluation in python? b. Static file contains which type of file normally? c. How can be file read in specific location? d. How do you remove a file from a folder? e. If user upload excel file check file format if it is valid or invalid. f. What is context manager? g. What is Garbage Collector? g. What is the difference between Static method and Class method? h. Private variable and how we can access that? i. Inheritance and types of inheritance. j. Difference multilevel and multiple inheritance and drawback?

1. Exception Handling a. How can we handle errors in python? b. How many ways exception in python?
2. Multithreading a. What is GIL?
3. Numpy a. Tell me some python libraries .. Numpy b. Convert a list into numpy

https://pynative.com/python-constructors/

######## chatgpt Certainly, here are some Python interview questions across different levels of complexity:

Beginner:

How can you iterate over a dictionary?

Intermediate:

What are decorators in Python, and how do they work? Explain the Global Interpreter Lock (GIL) in Python. How does it affect multi-threading? What are list comprehensions? Provide an example. What is the purpose of the **init** method in a Python class? How does exception handling work in Python? Provide examples of using try, except, finally. Describe the difference between shallow copy and deep copy in Python.

Advanced:

What are generators and iterators in Python? How are they different from each other? Explain the concept of "duck typing" in Python. How can you profile and optimize Python code for performance? Describe the use of context managers using the with statement. Explain the concept of metaclasses in Python. What are async functions in Python? How do they relate to asynchronous programming?

Problem-Solving:

Write a function to check if a given string is a palindrome. Implement a binary search algorithm. Given a list of numbers, write a function to find the two numbers that add up to a specific target sum. Implement a basic LRU (Least Recently Used) cache. Remember, interview questions can vary widely based on the specific role and company. It's important to not only memorize answers but also to understand the underlying concepts. Practice coding these questions and explaining your thought process out loud, as interviews often focus on problem-solving skills and how well you can communicate your solutions. ############

AbstractUser vs AbstractBaseUser The default User model in Django uses a username to uniquely identify a user during authentication. If you'd rather use an email address, you'll need to create a custom User model by either subclassing AbstractUser or AbstractBaseUser.

Options:

AbstractUser: Use this option if you are happy with the existing fields on the User model and just want to remove the username field. AbstractBaseUser: Use this option if you want to start from scratch by creating your own, completely new User model.

https://books.agiliq.com/projects/django-orm-cookbook/en/latest/null_vs_blank.html