ADDITIONAL 15 Python Developer Interview Questions-Answers

Watch Full Video On Youtube:
https://youtu.be/KRUBHw92aLI

Connect with me:
Youtube: https://www.youtube.com/c/nitmantalks
Instagram: https://www.instagram.com/nitinmangotra/
LinkedIn: https://www.linkedin.com/in/nitin-mangotra-9a075a149/
Facebook: https://www.facebook.com/NitManTalks/
Twitter: https://twitter.com/nitinmangotra07/
Telegram: https://t.me/nitmantalks/

1. Difference Between List and Dictionary

LIST

1. Lists are the collection of various elements (Heterogeneous).
2. List is mutable in nature.
3. **Syntax:** Placing all the elements inside square brackets [], separated by commas(,)

   list = ['a', 'b', 'c', 1,2,3]

4. Indices are integer values starts from value 0.
5. We can access the elements using the index value
6. The default order of elements is always maintained
7. List object is created using list() function

Dictionary

1. Dictionary are collection of elements in the hashed structure as key-value pairs.
2. It is also mutable, but keys do not allow duplicates.
3. **Syntax:** Placing all key-value pairs inside curly brackets({}), separated by a comma. Also, each key-pair is separated by a semi-colon (:)

   dict = {1: 'Apple', 2: 'Orange', 3: 'Mango'}

4. The keys in the dictionary are of any data type
5. We can access the elements using the keys
6. No guarantee of maintaining the order
7. Dictionary object is created using dict() function

2. Explain Append() And Extend() Property Of List

Append():.

- append() adds an element to a list
- append() adds its argument as a single element to the end of a list.
- The length of the list itself will increase by one.

Extend():

- extend() concatenates the first list with another list/iterable.
- extend() iterates over its argument adding each element to the list, extending the list.
- The length of the list will increase by however many elements were in the iterable argument.

2. Explain Append() And Extend() Property Of List

In Case Of String Value

Consider 2 Lists - List1 & List2

list1 = [1,2,3]
list2 = [5,6,7]

Append():

```
list1.append('AB')
print(list1)
```

Output:
[1, 2, 3, 'AB']

Extend():

```
list1.extend('AB')
print(list1)
```

Output:
[1, 2, 3, 'A', 'B']

2. Explain Append() And Extend() Property Of List

In Case Of List Value

Consider 2 Lists - List1 & List2

list1 = [1,2,3]
list2 = [5,6,7]

Append():

```
list1.append(list2)
print(list1)
```

Output:
[1, 2, 3, [5, 6, 7]]

Extend():

```
list1.extend(list2)
print(list1)
```

Output:
[1, 2, 3, 5, 6, 7]

2. Explain Append() And Extend() Property Of List

In Case Of Integer Value

Consider 2 Lists - List1 & List2

list1 = [1,2,3]
list2 = [5,6,7]

Append():

```
list1.append(4)
print(list1)
```

Output:
[1, 2, 3, 4]

Extend():

```
list1.extend(4)
print(list1)
```

Output:

```
Traceback (most recent call last):
  File "main.py", line 5, in <module>
    list1.extend(4)
TypeError: 'int' object is not iterable
```

3. Can We Use List Or Tuple As A Key In Dictionary?

- ➔ NO, List Can’t Be Used As Key In Dictionary
- ➔ YES, Tuple Can Be Used As A Key In Dictionary.
- ➔ NO, Tuple Containing List Can’t Be Used As A Key In Dictionary.

3. Can We Use List Or Tuple As A Key In Dictionary?

- Dictionaries are indexed by keys.
- Those Keys can be any immutable type i.e strings and numbers can always be keys.
- Tuples can be used as keys if they contain only strings, numbers, or tuples.
- If a tuple contains any mutable object either directly or indirectly, it cannot be used as a key.
- You can’t use lists as keys, since lists can be modified in place using index assignments, slice assignments, or methods like append() and extend().

3. Can We Use List Or Tuple As A Key In Dictionary?

Tuple As Key

```
tuple1 = (1,2,3)
tuple2 = (2,3,4)

d1 = {tuple1: 'First', tuple2: 'Second'}
print(d1)

OUTPUT:
{(1, 2, 3): 'First', (2, 3, 4): 'Second'}
```

List As Key

```
list1 = [1,2,3]
list2 = [2,3,4]

d1 = {list1: 'First', list2: 'Second'}
print(d1)


OUTPUT:

Traceback (most recent call last):
  File "main.py", line 10, in <module>
    d1 = {list1: 'First', list2: 'Second'}
TypeError: unhashable type: 'list'
```

Tuple Having List As Key

```
tuple1 = (1,2,3,[6,5,4])
tuple2 = (2,3,4)

d1 = {tuple1: 'First', tuple2: 'Second'}
print(d1)


OUTPUT:

Traceback (most recent call last):
  File "main.py", line 4, in <module>
    d1 = {tuple1: 'First', tuple2: 'Second'}
TypeError: unhashable type: 'list'
```

4. When To Use List And When To Tuple In Python?

- ➔ If you have data which is not meant to be changed in the first place, you should choose tuple data type over lists.
- ➔ And if you have data which is meant to be changed in the first place, you should choose list data type over tuple.

LIST

1. Lists are **mutable**
2. List is a container to contain different types of objects and is used to iterate objects.
3. Syntax Of List

   list = ['a', 'b', 'c', 1,2,3]

4. List iteration is slower
5. Lists consume more memory
6. Operations like insertion and deletion are better performed.

Tuple

1. Tuples are **immutable**
2. Tuple is also similar to list but contains immutable objects.
3. Syntax Of Tuple

   tuples = ('a', 'b', 'c', 1, 2)

4. Tuple processing is faster than List.
5. Tuple consume less memory
6. Elements can be accessed better.

5. Why Python Is Called As Dynamic Typed Programming Language OR What Is Duck Typing?

NOTE: The **"Duck typing"** name comes from the phrase, “If it walks like a duck and it quacks like a duck, then it must be a duck.”

- Python don't have any problem even if we don't declare the type of variable.
- It states the kind of variable in the runtime of the program.
- Python also take cares of the memory management which is crucial in programming. So, Python is a dynamically typed language.

```
# variable a is assigned to a string
a ="NitMan Talks"
print(type(a))

Output: <class 'str'>
```

```
# variable a is assigned to an integer
a = 7
print(type(a))

Output: <class 'int'>
```

6. What If We Don't Use “With” Statement

If We Don't use "WITH" Statement, We need to close the opened file manually but using close().

In Case Of open():

```
file = open("hello.txt", "w")
file.write("Hello, World!")
file.close()
```

```
# Safely open the file
file = open("hello.txt", "w")
try:
    file.write("Hello, World!")
finally:
    file.close()
```

```
with open("hello.txt", "w") as file:
    file.write("Hello, World!")
```

2 Internal functions called in "WITH" Statement

- **.___enter___()** is called by the with statement to enter the runtime context.
- **.___exit___()** is called when the execution leaves the with code block.

7. Why Python Is Called As An Interpreted Language?

- Interpreted simply means your code run line by line. While in compiled language whole program compiled at once.
- In compilation, source code is first converted to object code and then to the machine code.
- You can see that in a compiled language your whole program compiled at once and give the output.
- But in interpreted language, every single line is converted to machine code directly. That's why Python is very slow, because it interpret one line at a time.
- You have seen that if, you run code in Python and do any mistake at the bottom of your python code. And you execute the code then it will give you output till the correct code part and the remaining part gives the error in a console window where you do the error. Because the interpreted language executes line by line instead of executing the whole program at once.
- But, the same thing in a compiled language you always get the error. Your whole code must be correct for the execution of the program.

8. What is Lambda Function. Explain with an example.

Lambda Function:

- A Lambda Function in Python programming is an anonymous function or a function having no name.
- It is a small and restricted function having no more than one line.
- Just like a normal function, a Lambda function can have multiple arguments with one expression.

Syntax:

```
lambda arguments : expression
```

Example:

Add 5 to argument a, and return the result:

```
x = lambda a : a + 5
print(x(5))
```

Output:
10

Example 2:

Find Cubes Of All Elements In A List.

```
list1 = [1, 2, 3, 4, 5, 6, 7]
res = list(map(lambda x: x ** 3, list1))
print(res)
```

Output:
[1, 8, 27, 64, 125, 216, 343]

9. What Does Python Support? - Call By Reference OR Call By Value.

- Python utilizes a system, which is known as "**Call by Object Reference**" or "**Call by assignment**".
- In the event that you pass arguments like whole numbers, strings or tuples to a function, the passing is like **call-by-value** because you can not change the value of the immutable objects being passed to the function.
- Whereas passing mutable objects can be considered as **call by reference** because when their values are changed inside the function, then it will also be reflected outside the function.

Source: https://www.geeksforgeeks.org/is-python-call-by-reference-or-call-by-value/

9. What Does Python Support? - Call By Reference OR Call By Value.

Example 1:

```
s1 = "Geeks"
def test(s1):
        s1 = "GeeksforGeeks"
        print("Inside Function:", s1)

test(s1)
print("Outside Function:", s1)
```

Output:
Inside Function: GeeksforGeeks
Outside Function: Geeks

Example 2:

```
def add_more(list):
        list.append(50)
        print("Inside Function", list)

mylist = [10,20,30,40]
add_more(mylist)
print("Outside Function:", mylist)
```

Output:
Inside Function [10, 20, 30, 40, 50]
Outside Function: [10, 20, 30, 40, 50]

Binding Names to Objects: In python, each variable to which we assign a value/container is treated as an object. When we are assigning a value to a variable, we are actually binding a name to an object.

Source: https://www.geeksforgeeks.org/is-python-call-by-reference-or-call-by-value/

9. What Does Python Support? - Call By Reference OR Call By Value.

Example 3:

```
a = "first"
b = "first"

print(id(a))
print(id(b))
print(a is b)
```

Output:

```
110001234557894
110001234557894
True
```

Example 2:

```
a = [10, 20, 30]
b = [10, 20, 30]

print(id(a))
print(id(b))
print(a is b)
```

Output:

```
541190289536222
541190288737777
False
```

The output of the above two examples are different because the list is **mutable** and the string is **immutable**.
An **immutable** variable cannot be changed once created. If we wish to change an **immutable** variable, such as a string, we must create a new instance and bind the variable to the new instance.
Whereas, mutable variable can be changed in place.

Source: https://www.geeksforgeeks.org/is-python-call-by-reference-or-call-by-value/

10. What Is MRO In Python

- MRO stands for **Method Resolution Order**
- MRO is a concept used in inheritance.
- It is the order in which a method is searched for in a classes hierarchy and is especially useful in Python because Python supports multiple inheritance.
- In Python, the MRO is from **bottom to top and left to right**.
- This means that, first, the method is searched in the class of the object. If it's not found, it is searched in the immediate super class.
- In the case of multiple super classes, it is searched left to right, in the order by which was declared by the developer.
- For example:

```
def class C(B,A):
```

In this case, the **MRO would be C -> B -> A.**
Since B was mentioned first in class declaration, it will be searched first while resolving a method.

Source: https://www.educative.io/edpresso/what-is-mro-in-python

10. What Is MRO In Python

Example 1:

```
class A:
  def method(self):
    print("A.method() called")

class B(A):
  def method(self):
    print("B.method() called")

b = B()
b.method()
```

This is a simple case with single inheritance.

In this case, when `b.method()` is called, it first searches for the method in class B.

In this case, class B had defined the method; hence, it is the one that was executed.

In the case where it is not present in B, then the method from its immediate super class (A) would be called.

So, the MRO for this case is: B -> A

Source: https://www.educative.io/edpresso/what-is-mro-in-python

10. What Is MRO In Python

Example 2:

```
class A:
  def method(self):
    print("A.method() called")

class B:
  pass

class C(B, A):
  pass

c = C()
c.method()
```

The MRO for this case is:

C -> B -> A

The method only existed in A, where it was searched for last.

Source: https://www.educative.io/edpresso/what-is-mro-in-python

10. What Is MRO In Python

Example 3:

```
class A:
 def method(self):
  print("A.method() called")

class B:
 def method(self):
  print("B.method() called")

class C(A, B):
 pass

class D(C, B):
 pass

d = D()
d.method()
```

The MRO for this can be a bit tricky.
The immediate superclass for D is C, so if the method is not found in D, it is searched for in C. However, if it is not found in C, then you have to decide if you should check A (declared first in the list of C's super classes) or check B (declared in D's list of super classes after C).
In Python 3 onwards, this is resolved as first checking A.

So, the MRO becomes:

D -> C -> A -> B

Source: https://www.educative.io/edpresso/what-is-mro-in-python

11. Is Python A Fully Object Oriental Language?

- Python supports all the concept of "object oriented programming" but it is NOT fully object oriented because - The code in Python can also be written without creating classes.
- The answer is simply philosophy. Guido doesn't like hiding things, and many in the Python community agree with him.
- While it borrows heavily from the OOP language, it is also at the same time functional, procedural, imperative, and reflective.
- Python doesn't support strong encapsulation, which is only one of many features associated with the term "object-oriented".
- Python doesn’t support Interfaces

12. Explain decorator / Create A Customized Decorator / Add two numbers using Decorator / Parameterized Decorator.

A Decorator is just a function that takes another function as an argument, add some kind of functionality and then returns another function.
All of this without altering the source code of the original function that you passed in.

Decorator Coding Example

```
def decorator_func(func):
        def wrapper_func():
                print("wrapper_func Worked")
                return func()
        print("decorator_func worked")
        return wrapper_func

def show():
        print("Show Worked")
decorator_show = decorator_func(show)
decorator_show()


#Alternative
@decorator_func
def display():
        print('display
                worked')
display()
```

Output:

```
decorator_func worked
wrapper_func Worked
Show Worked
decorator_func worked
wrapper_func Worked
display worked
```

12. Explain decorator / Create A Customized Decorator / Add two numbers using Decorator / Parameterized Decorator.

A Decorator is just a function that takes another function

```
def addTwoNumbers(a, b):
   c=a+b
   return c

c=addTwoNumber(4, 5)

print("Addition of two numbers=", c)

#Addition of two numbers=9
```

Now our aim is to modify the behavior of addTwoNumbers() without changing function definition and function call.

Source: https://www.csestack.org/python-decorators/

12. Explain decorator / Create A Customized Decorator / Add two numbers using Decorator / Parameterized Decorator.

What function behavior do we want to change?
We want addTwoNumbers function should calculate the sum of the square of two numbers instead of the sum of two numbers. Here is a simple decorator to change the behavior of the existing function.a

```
def decorateFun(func):
  def sumOfSquare(x, y):
    return func(x**2, y**2)
  return sumOfSquare

@decorateFun
def addTwoNumbers(a, b):
  c = a+b
  return c

c = addTwoNumbers(4,5)
print("Addition of two numbers=", c)

#Addition of two numbers=41
```

The below simple program is equivalent to the above decorator example. Here we are changing the function call.

```
def decorateFun(func):
  def sumOfSquare(x, y):
    return func(x**2, y**2)
  return sumOfSquare

def addTwoNumbers(a, b):
  c = a+b
  return c

obj=decorateFun(addTwoNumbers)
c=obj(4,5)
print("Addition of square of two numbers=", c)
#Addition of square of two numbers=41
```

Source: https://www.csestack.org/python-decorators/

13. Difference Between Static & Class Method.

Class Method	Static Method
The class method takes cls (class) as first argument.	The static method does not take any specific parameter.
Class method can access and modify the class state.	Static Method cannot access or modify the class state.
The class method takes the class as parameter to know about the state of that class.	Static methods do not know about class state. These methods are used to do some utility tasks by taking some parameters.
@classmethod decorator is used here.	@staticmethod decorator is used here.

14. Explain OOPS Concept In Python.

Like other general-purpose programming languages, Python is also an object-oriented language since its beginning. It allows us to develop applications using an Object-Oriented approach. In Python, we can easily create and use classes and objects.

An object-oriented paradigm is to design the program using classes and objects. The object is related to real-word entities such as book, house, pencil, etc. The oops concept focuses on writing the reusable code. It is a widespread technique to solve the problem by creating objects.

Major principles of object-oriented programming system are given below.

- **Class**
- **Object**
- **Method**
- **Inheritance**
- **Polymorphism**
- **Data Abstraction**
- **Encapsulation**

Source: https://www.javatpoint.com/python-oops-concepts

15. What is Abstraction And How To Define Abstract Classes and Functions Using Abstraction

Abstraction in Python:

Abstraction is used to hide the internal functionality of the function from the users. The users only interact with the basic implementation of the function, but inner working is hidden. User is familiar with that "what function does" but they don't know "how it does."

In simple words, we all use the smartphone and very much familiar with its functions such as camera, voice-recorder, call-dialing, etc., but we don't know how these operations are happening in the background. Let's take another example - When we use the TV remote to increase the volume. We don't know how pressing a key increases the volume of the TV. We only know to press the "+" button to increase the volume.

That is exactly the abstraction that works in the object-oriented concept.

Why Abstraction is Important?

In Python, an abstraction is used to hide the irrelevant data/class in order to reduce the complexity. It also enhances the application efficiency. Next, we will learn how we can achieve abstraction using the Python program.

Source: https://www.javatpoint.com/abstraction-in-python

15. What is Abstraction And How To Define Abstract Classes and Functions Using Abstraction

Abstraction classes in Python

In Python, abstraction can be achieved by using abstract classes and interfaces.

A class that consists of one or more abstract method is called the abstract class. Abstract methods do not contain their implementation. Abstract class can be inherited by the subclass and abstract method gets its definition in the subclass. Abstraction classes are meant to be the blueprint of the other class. An abstract class can be useful when we are designing large functions. An abstract class is also helpful to provide the standard interface for different implementations of components. Python provides the abc module to use the abstraction in the Python program. Let's see the following syntax.

Syntax

```
from abc import ABC
class ClassName(ABC):
```

We import the ABC class from the abc module.

Source: https://www.javatpoint.com/abstraction-in-python

15. What is Abstraction And How To Define Abstract Classes and Functions Using Abstraction

Abstract Base Classes

An abstract base class is the common application program of the interface for a set of subclasses. It can be used by the third-party, which will provide the implementations such as with plugins. It is also beneficial when we work with the large code-base hard to remember all the classes.

Working of the Abstract Classes:

Unlike the other high-level language, Python doesn't provide the abstract class itself. We need to import the abc module, which provides the base for defining Abstract Base classes (ABC). The ABC works by decorating methods of the base class as abstract. It registers concrete classes as the implementation of the abstract base. We use the @abstractmethod decorator to define an abstract method or if we don't provide the definition to the method, it automatically becomes the abstract method. Let's understand the following example.

Source: https://www.javatpoint.com/abstraction-in-python

15. What is Abstraction And How To Define Abstract Classes and Functions Using Abstraction

Example -

```
from abc import ABC, abstractmethod
class Car(ABC):
   def mileage(self):
      pass

class Tesla(Car):
   def mileage(self):
      print("The mileage is 30kmph")
class Suzuki(Car):
   def mileage(self):
      print("The mileage is 25kmph ")
class Duster(Car):
    def mileage(self):
       print("The mileage is 24kmph ")

class Renault(Car):
   def mileage(self):
        print("The mileage is 27kmph ")
```

```
# Driver code
t= Tesla ()
t.mileage()

r = Renault()
r.mileage()

s = Suzuki()
s.mileage()
d = Duster()
d.mileage()
```

Output:

```
The mileage is 30kmph
The mileage is 27kmph
The mileage is 25kmph
The mileage is 24kmph
```

Explanation -

In the above code, we have imported the abc module to create the abstract base class.
We created the Car class that inherited the ABC class and defined an abstract method named mileage().
We have then inherited the base class from the three different subclasses and implemented the abstract method differently.
We created the objects to call the abstract method.

Source: https://www.javatpoint.com/abstraction-in-python

Thanks! Hope It Helps You!

Watch The Answers For The Remaining Questions On My Youtube Channel.
Link For The Remaining Questions : https://youtu.be/KRUBHw92aLI

Connect with me:
Youtube: https://www.youtube.com/c/nitmantalks
Instagram: https://www.instagram.com/nitinmangotra/
LinkedIn: https://www.linkedin.com/in/nitin-mangotra-9a075a149/
Facebook: https://www.facebook.com/NitManTalks/
Twitter: https://twitter.com/nitinmangotra07/
Telegram: https://t.me/nitmantalks/

Do Connect With Me!!!!

Please Do Comment Your Feedback In Comment Section Of My Video On Youtube.
Here Is The Link: https://youtu.be/KRUBHw92aLI

When You Get Placed In Any Company Because Of My Video, DO Let Me Know.
It will Give Me More Satisfaction and Will Motivate me to make more such video Content!!

Thanks
PS: Don't Forget To Connect WIth ME.
Regards,
Nitin Mangotra (NitMan)

Connect with me:
Youtube: https://www.youtube.com/c/nitmantalks
Instagram: https://www.instagram.com/nitinmangotra/
LinkedIn: https://www.linkedin.com/in/nitin-mangotra-9a075a149/
Facebook: https://www.facebook.com/NitManTalks/
Twitter: https://twitter.com/nitinmangotra07/
Telegram: https://t.me/nitmantalks/