**Links:**

**Python Full Course -** [**https://online.codeanddebug.in/courses/core-advance-python-mastery/**](https://online.codeanddebug.in/courses/core-advance-python-mastery/)

* [https://www.interviewbit.com/python-interview-questions/#what-is-python-what-are-the-benefits-of-using-python](https://www.interviewbit.com/python-interview-questions/)
* <https://www.edureka.co/blog/interview-questions/python-interview-questions/>
* <https://www.interviewbit.com/python-cheat-sheet/>
* <https://www.interviewbit.com/python-interview-questions/>
* <https://www.interviewbit.com/blog/python-commands/>
* <https://www.interviewbit.com/python-mcq/>
* <https://www.interviewbit.com/hr-interview-questions/>
* [Code and Debug - YouTube](https://www.youtube.com/@codeanddebug/playlists)
* [https://github.com/DopplerHQ/awesome-interview-questions?tab=readme-ov-file#python](https://github.com/DopplerHQ/awesome-interview-questions?tab=readme-ov-file)
* <https://www.tutorialsteacher.com/python/magic-methods-in-python>
* [**https://codersdaily.in/blog/python-dictionary-interview-questions#google\_vignette**](https://codersdaily.in/blog/python-dictionary-interview-questions#google_vignette)
* [**https://medium.com/@nikitasilaparasetty/python-interview-coding-questions-with-solutions-for-beginners-7f6d782defac**](https://medium.com/@nikitasilaparasetty/python-interview-coding-questions-with-solutions-for-beginners-7f6d782defac)
* [**https://www.prepbytes.com/blog/python/python-interview-questions/**](https://www.prepbytes.com/blog/python/python-interview-questions/)
* [**https://www.prepbytes.com/blog/interview-questions/top-10-basic-python-programming-interview-questions/**](https://www.prepbytes.com/blog/interview-questions/top-10-basic-python-programming-interview-questions/)
* [**https://www.javatpoint.com/python-interview-questions**](https://www.javatpoint.com/python-interview-questions)
* [**https://www.javatpoint.com/python-coding-interview-questions**](https://www.javatpoint.com/python-coding-interview-questions)
* [**https://www.guru99.com/python-dictionary-beginners-tutorial.html**](https://www.guru99.com/python-dictionary-beginners-tutorial.html)
* [**https://www.guru99.com/python-tutorials.html**](https://www.guru99.com/python-tutorials.html)
* [**https://www.interviewbit.com/python-interview-questions/#python-mcq**](https://www.interviewbit.com/python-interview-questions/#python-mcq)
* [**https://luminousmen.com/post/python-interview-questions-junior**](https://luminousmen.com/post/python-interview-questions-junior)
* [**https://github.com/donnemartin/interactive-coding-challenges?tab=readme-ov-file**](https://github.com/donnemartin/interactive-coding-challenges?tab=readme-ov-file)
* [**https://www.google.com/url?sa=t HYPERLINK "https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjQ2aaLkZKIAxWy4DgGHdu1KZ0QFnoECBsQAw&url=https%3A%2F%2Fwww.shiksha.com%2Fonline-courses%2Farticles%2Fconstructors-in-python-definition-types-and-rules%2F%23%3A~%3Atext%3DConstructors%2520in%2520Python%2520is%2520a%2Cvalue%2520to%2520the%2520object%27s%2520members.&usg=AOvVaw1iOMsFgO6rIrWzlWcLk6M9&opi=89978449"& HYPERLINK "https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjQ2aaLkZKIAxWy4DgGHdu1KZ0QFnoECBsQAw&url=https%3A%2F%2Fwww.shiksha.com%2Fonline-courses%2Farticles%2Fconstructors-in-python-definition-types-and-rules%2F%23%3A~%3Atext%3DConstructors%2520in%2520Python%2520is%2520a%2Cvalue%2520to%2520the%2520object%27s%2520members.&usg=AOvVaw1iOMsFgO6rIrWzlWcLk6M9&opi=89978449"rct=j HYPERLINK "https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjQ2aaLkZKIAxWy4DgGHdu1KZ0QFnoECBsQAw&url=https%3A%2F%2Fwww.shiksha.com%2Fonline-courses%2Farticles%2Fconstructors-in-python-definition-types-and-rules%2F%23%3A~%3Atext%3DConstructors%2520in%2520Python%2520is%2520a%2Cvalue%2520to%2520the%2520object%27s%2520members.&usg=AOvVaw1iOMsFgO6rIrWzlWcLk6M9&opi=89978449"& HYPERLINK "https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjQ2aaLkZKIAxWy4DgGHdu1KZ0QFnoECBsQAw&url=https%3A%2F%2Fwww.shiksha.com%2Fonline-courses%2Farticles%2Fconstructors-in-python-definition-types-and-rules%2F%23%3A~%3Atext%3DConstructors%2520in%2520Python%2520is%2520a%2Cvalue%2520to%2520the%2520object%27s%2520members.&usg=AOvVaw1iOMsFgO6rIrWzlWcLk6M9&opi=89978449"q= HYPERLINK "https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjQ2aaLkZKIAxWy4DgGHdu1KZ0QFnoECBsQAw&url=https%3A%2F%2Fwww.shiksha.com%2Fonline-courses%2Farticles%2Fconstructors-in-python-definition-types-and-rules%2F%23%3A~%3Atext%3DConstructors%2520in%2520Python%2520is%2520a%2Cvalue%2520to%2520the%2520object%27s%2520members.&usg=AOvVaw1iOMsFgO6rIrWzlWcLk6M9&opi=89978449"& HYPERLINK "https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjQ2aaLkZKIAxWy4DgGHdu1KZ0QFnoECBsQAw&url=https%3A%2F%2Fwww.shiksha.com%2Fonline-courses%2Farticles%2Fconstructors-in-python-definition-types-and-rules%2F%23%3A~%3Atext%3DConstructors%2520in%2520Python%2520is%2520a%2Cvalue%2520to%2520the%2520object%27s%2520members.&usg=AOvVaw1iOMsFgO6rIrWzlWcLk6M9&opi=89978449"esrc=s HYPERLINK "https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjQ2aaLkZKIAxWy4DgGHdu1KZ0QFnoECBsQAw&url=https%3A%2F%2Fwww.shiksha.com%2Fonline-courses%2Farticles%2Fconstructors-in-python-definition-types-and-rules%2F%23%3A~%3Atext%3DConstructors%2520in%2520Python%2520is%2520a%2Cvalue%2520to%2520the%2520object%27s%2520members.&usg=AOvVaw1iOMsFgO6rIrWzlWcLk6M9&opi=89978449"& HYPERLINK "https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjQ2aaLkZKIAxWy4DgGHdu1KZ0QFnoECBsQAw&url=https%3A%2F%2Fwww.shiksha.com%2Fonline-courses%2Farticles%2Fconstructors-in-python-definition-types-and-rules%2F%23%3A~%3Atext%3DConstructors%2520in%2520Python%2520is%2520a%2Cvalue%2520to%2520the%2520object%27s%2520members.&usg=AOvVaw1iOMsFgO6rIrWzlWcLk6M9&opi=89978449"source=web HYPERLINK "https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjQ2aaLkZKIAxWy4DgGHdu1KZ0QFnoECBsQAw&url=https%3A%2F%2Fwww.shiksha.com%2Fonline-courses%2Farticles%2Fconstructors-in-python-definition-types-and-rules%2F%23%3A~%3Atext%3DConstructors%2520in%2520Python%2520is%2520a%2Cvalue%2520to%2520the%2520object%27s%2520members.&usg=AOvVaw1iOMsFgO6rIrWzlWcLk6M9&opi=89978449"& HYPERLINK "https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjQ2aaLkZKIAxWy4DgGHdu1KZ0QFnoECBsQAw&url=https%3A%2F%2Fwww.shiksha.com%2Fonline-courses%2Farticles%2Fconstructors-in-python-definition-types-and-rules%2F%23%3A~%3Atext%3DConstructors%2520in%2520Python%2520is%2520a%2Cvalue%2520to%2520the%2520object%27s%2520members.&usg=AOvVaw1iOMsFgO6rIrWzlWcLk6M9&opi=89978449"cd= HYPERLINK "https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjQ2aaLkZKIAxWy4DgGHdu1KZ0QFnoECBsQAw&url=https%3A%2F%2Fwww.shiksha.com%2Fonline-courses%2Farticles%2Fconstructors-in-python-definition-types-and-rules%2F%23%3A~%3Atext%3DConstructors%2520in%2520Python%2520is%2520a%2Cvalue%2520to%2520the%2520object%27s%2520members.&usg=AOvVaw1iOMsFgO6rIrWzlWcLk6M9&opi=89978449"& HYPERLINK "https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjQ2aaLkZKIAxWy4DgGHdu1KZ0QFnoECBsQAw&url=https%3A%2F%2Fwww.shiksha.com%2Fonline-courses%2Farticles%2Fconstructors-in-python-definition-types-and-rules%2F%23%3A~%3Atext%3DConstructors%2520in%2520Python%2520is%2520a%2Cvalue%2520to%2520the%2520object%27s%2520members.&usg=AOvVaw1iOMsFgO6rIrWzlWcLk6M9&opi=89978449"ved=2ahUKEwjQ2aaLkZKIAxWy4DgGHdu1KZ0QFnoECBsQAw HYPERLINK "https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjQ2aaLkZKIAxWy4DgGHdu1KZ0QFnoECBsQAw&url=https%3A%2F%2Fwww.shiksha.com%2Fonline-courses%2Farticles%2Fconstructors-in-python-definition-types-and-rules%2F%23%3A~%3Atext%3DConstructors%2520in%2520Python%2520is%2520a%2Cvalue%2520to%2520the%2520object%27s%2520members.&usg=AOvVaw1iOMsFgO6rIrWzlWcLk6M9&opi=89978449"& HYPERLINK "https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjQ2aaLkZKIAxWy4DgGHdu1KZ0QFnoECBsQAw&url=https%3A%2F%2Fwww.shiksha.com%2Fonline-courses%2Farticles%2Fconstructors-in-python-definition-types-and-rules%2F%23%3A~%3Atext%3DConstructors%2520in%2520Python%2520is%2520a%2Cvalue%2520to%2520the%2520object%27s%2520members.&usg=AOvVaw1iOMsFgO6rIrWzlWcLk6M9&opi=89978449"url=https%3A%2F%2Fwww.shiksha.com%2Fonline-courses%2Farticles%2Fconstructors-in-python-definition-types-and-rules%2F%23%3A~%3Atext%3DConstructors%2520in%2520Python%2520is%2520a%2Cvalue%2520to%2520the%2520object%27s%2520members. HYPERLINK "https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjQ2aaLkZKIAxWy4DgGHdu1KZ0QFnoECBsQAw&url=https%3A%2F%2Fwww.shiksha.com%2Fonline-courses%2Farticles%2Fconstructors-in-python-definition-types-and-rules%2F%23%3A~%3Atext%3DConstructors%2520in%2520Python%2520is%2520a%2Cvalue%2520to%2520the%2520object%27s%2520members.&usg=AOvVaw1iOMsFgO6rIrWzlWcLk6M9&opi=89978449"& HYPERLINK "https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjQ2aaLkZKIAxWy4DgGHdu1KZ0QFnoECBsQAw&url=https%3A%2F%2Fwww.shiksha.com%2Fonline-courses%2Farticles%2Fconstructors-in-python-definition-types-and-rules%2F%23%3A~%3Atext%3DConstructors%2520in%2520Python%2520is%2520a%2Cvalue%2520to%2520the%2520object%27s%2520members.&usg=AOvVaw1iOMsFgO6rIrWzlWcLk6M9&opi=89978449"usg=AOvVaw1iOMsFgO6rIrWzlWcLk6M9 HYPERLINK "https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjQ2aaLkZKIAxWy4DgGHdu1KZ0QFnoECBsQAw&url=https%3A%2F%2Fwww.shiksha.com%2Fonline-courses%2Farticles%2Fconstructors-in-python-definition-types-and-rules%2F%23%3A~%3Atext%3DConstructors%2520in%2520Python%2520is%2520a%2Cvalue%2520to%2520the%2520object%27s%2520members.&usg=AOvVaw1iOMsFgO6rIrWzlWcLk6M9&opi=89978449"& HYPERLINK "https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjQ2aaLkZKIAxWy4DgGHdu1KZ0QFnoECBsQAw&url=https%3A%2F%2Fwww.shiksha.com%2Fonline-courses%2Farticles%2Fconstructors-in-python-definition-types-and-rules%2F%23%3A~%3Atext%3DConstructors%2520in%2520Python%2520is%2520a%2Cvalue%2520to%2520the%2520object%27s%2520members.&usg=AOvVaw1iOMsFgO6rIrWzlWcLk6M9&opi=89978449"opi=89978449**](https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjQ2aaLkZKIAxWy4DgGHdu1KZ0QFnoECBsQAw&url=https%3A%2F%2Fwww.shiksha.com%2Fonline-courses%2Farticles%2Fconstructors-in-python-definition-types-and-rules%2F%23%3A~%3Atext%3DConstructors%2520in%2520Python%2520is%2520a%2Cvalue%2520to%2520the%2520object%27s%2520members.&usg=AOvVaw1iOMsFgO6rIrWzlWcLk6M9&opi=89978449)
* **MAP()**
* **LAMBDA()**
* **Memory managed in python**
* [**https://drive.google.com/drive/folders/1LahwPSc6f9nkxBiRrz6LFUzkrg-Kzvov**](https://drive.google.com/drive/folders/1LahwPSc6f9nkxBiRrz6LFUzkrg-Kzvov)



**What is Python**

* Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures combined with dynamic typing and dynamic binding make it very attractive for Rapid Application Development

key features of Python

* Python is an **interpreted** language, Python does not need to be compiled before it is run,
* Python is **dynamically typed,** this means that you don’t need to state the types of variables when you declare them or anything like that. You can do things like x=111 and then x="I'm a string" without error
* Python is well suited to [**object orientated programming**](https://www.edureka.co/blog/python-class/) in that it allows the definition of classes along with composition and inheritance.

**Why We Use Python**

* We Can Python on different platforms Like Mac, Linux,Windows,  Raspberry.
* python Syntax is very easy  that why its allow to write code in few line.
* Python can be treated in a procedural way, an object-orientated way or a functional way.

When to use list vs. tuple vs. dictionary vs. set?

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  
30000.0

Tuples are similar to lists, but there data can be changed once created throught the execution of program. Individual element of Tuples can be accessed using indexing.  
  
**Tuples Code Snippet:** The Days  
  
days = ("Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday")  
print days  
------------------------------  
('Sunday', 'Mondays', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday')  
  
Sets stores unordered values & have no index. And unlike Tuples and Lists, Sets can have no duplicate data, It is similar to Mathematical sets.- add() function can be used to add element to a set.  
- update() function can be used to add a group of elements to a set.  
- Copy() function can be used to create clone of set.  
  
**Set Code Snippet:**

disneyLand = set (['Minnie Mouse', 'Donald Duck', 'Daisy Duck', 'Goofy'])  
disneyLand.add('Pluto')  
print disneyLand  
-----------------------------------------------  
**Output**  
set(['Goofy', 'Daisy Duck', 'Donald Duck', 'Minnie Mouse', ’Pluto’])

Dictionary are similar to what their name is. In a dictionary, In python, the word is called a 'key', and the definition a 'value'. Dictionaries consist of pairs of keys and their corresponding values.  
  
**Dictionary Code Snippet:**

>>> dict = {'India': 'Bharat', 'Angel': ‘Mother Teresa’, 'Cartoon': 'Mickey'}  
>>>print dict[India]  
Bharat  
>>>print dict[Angel]  
Mother Teresa

**What is enumarate function inside a dictionary?**  
You can using enumarate function with dictionary to get position index and corresponding index at the same time.

For

i in enumerate(dict1):

dict1 = {0:”name”, 1: “age”, 2: “city”}

for I in enumerate(dict1):

print(i)

#output

(0, 'name')

(1, 'age')

(2, 'city')

How to do unit testing?

* A typical unit test contains 3 phases: First, it initializes a small piece of an application it wants to test (also known as the system under test, or SUT), then it applies some stimulus to the system under test (usually by calling a method on it), and finally, it observes the resulting behavior.

\*Explain the use of decorators.

Decorators in Python are used to modify or inject code in functions or classes. Using decorators, you can wrap a class or function method call so that a piece of code can be executed before or after the execution of the original code. Decorators can be used to check for permissions, modify or track the arguments passed to a method, logging the calls to a specific method, etc.

**Q.**How does break, continue and pass work?

|  |  |
| --- | --- |
| Break | Allows loop termination when some condition is met and the control is transferred to the next statement. |
| Continue | Allows skipping some part of a loop when some specific condition is met and the control is transferred to the beginning of the loop |
| Pass | Used when you need some block of code syntactically, but you want to skip its execution. This is basically a null operation. Nothing happens when this is executed. |

What is a Python package vs module?



A Python Module can be a simple python File (. py extension file), i.e., a combination of numerous Functions and Global variables. A Python Package is a collection of different Python modules with an \_\_init\_\_.py File.

* [Python modules](https://www.geeksforgeeks.org/python-modules/) may contain several classes, functions, variables, etc. whereas Python packages contain several modules. In simpler terms, Package in Python is a folder that contains various modules as files.

**Public Access Modifier:**

The members of a class that are declared public are easily accessible from any part of the program. All data members and member functions of a class are public by default.

**Protected Access Modifier:**

The members of a class that are declared protected are only accessible to a class derived from it. Data members of a class are declared protected by adding a single underscore ‘\_’ symbol before the data member of that class.

**Private Access Modifier:**

The members of a class that are declared private are accessible within the class only, private access modifier is the most secure access modifier. Data members of a class are declared private by adding a double underscore ‘\_\_’ symbol before the data member of that class.

What is the use of self in Python?

**Self**is used to represent the instance of the class. With this keyword, you can access the attributes and methods of the class in python. It binds the attributes with the given arguments. self is used in different places and often thought to be a keyword. But unlike in C++, self is not a keyword in Python. Instead of self we can use our own name

Ex:

def start(yash):

return yash.\_manageBDE.enable(yash.\_driveLetter)

12. 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. It helps in distinguishing methods and attributes of a class from local variables.

# class definition

**class** **Student**:

**def** **\_\_init\_\_**(self, fname, lname, age, section):

self.firstname = fname

self.lastname = lname

self.age = age

self.section = section

# creating a new object

# stu1 is the instance of class **Student**.

#\_\_init\_\_ allocates memory for stu1.

stu1 = Student("Sara", "Ansh", 22, "A2")

In Python, a constructor is a special method used to initialize objects of a class. It's called automatically when a new object is created.

Here's how it works:

* Defining the Constructor:
* Use the special method \_\_init\_\_() to define the constructor.
* It's typically the first method defined within a class.
* The self parameter is always the first argument, referring to the newly created object itself.
* Calling the Constructor:
* When you create an object of the class using the class name followed by parentheses (e.g., Person("Alice", 30)) the constructor is automatically invoked.
* You pass any necessary arguments to the constructor within those parentheses.

Example:

Python

class Person:

def \_\_init\_\_(self, name, age): # Constructor

self.name = name

self.age = age

# Create a Person object

person1 = Person("Bob", 25) # \_\_init\_\_() is called automatically

print(person1.name, person1.age) # Output: Bob 25

Key Points:

* Purpose: Constructors are used to set initial values for an object's attributes, ensuring objects are created in a valid and usable state.
* self Parameter: It's essential for accessing and assigning values to the object's attributes within the constructor.
* Default Constructor: If you don't define a constructor, Python provides a default one that doesn't take any arguments.
* Multiple Constructors: Python doesn't have multiple constructors in the traditional sense, but you can use default arguments or method overloading techniques to achieve similar results.

Example with Default Arguments:

Python

class Person:

def \_\_init\_\_(self, name="Unknown", age=0): # Default constructor

self.name = name

self.age = age

* Syntax for slicing is **[start : stop : step]**
* Slicing can be done on **strings, arrays, lists**, and **tuples**.

18. What is the difference between Python Arrays and lists?

* Arrays in python can only contain elements of same data types i.e., data type of array should be homogeneous. It is a thin wrapper around C language arrays and consumes far less memory than lists.
* Lists in python can contain elements of different data types i.e., data type of lists can be heterogeneous. It has the disadvantage of consuming large memory.

**import** array

a = array.array('i', [1, 2, 3])

**for** i **in** a:

print(i, end=' ') #OUTPUT: 1 2 3

a = array.array('i', [1, 2, 'string']) #OUTPUT: TypeError: an integer is required (got type str)

a = [1, 2, 'string']

**for** i **in** a:

print(i, end=' ') #OUTPUT: 1 2 string

‘’’

Here's an explanation of polymorphism in Python, with examples:

What is Polymorphism?

* It literally means "many forms."
* In programming, it refers to the ability of a function or object to behave differently depending on the context in which it's used.
* It allows you to create more adaptable and reusable code.

Types of Polymorphism in Python:

* Duck Typing: or Class Polymorphism in Python
* Focuses on behavior rather than explicit types.
* If an object has the necessary methods or attributes, it can be used, regardless of its actual type.

Example:

Python

def make\_sound(obj):

obj.quack() # If obj has a quack() method, it will be called

class Duck:

def quack(self):

print("Quack!")

class Person:

def quack(self):

print("I'm pretending to be a duck!")

duck = Duck()

person = Person()

make\_sound(duck) # Output: Quack!

make\_sound(person) # Output: I'm pretending to be a duck!

* Operator Overloading:
* Python doesn't support operator overloading directly.
* But you can create custom classes that mimic it using special methods (e.g., \_\_add\_\_, \_\_sub\_\_).

Example:

Python

class Point:

def \_\_init\_\_(self, x, y):

self.x = x

self.y = y

def \_\_add\_\_(self, other):

return Point(self.x + other.x, self.y + other.y)

p1 = Point(1, 2)

p2 = Point(3, 4)

p3 = p1 + p2 # Calls \_\_add\_\_ method

print(p3.x, p3.y) # Output: 4 6

* Method Overriding:
* Involves redefining a method in a child class that's inherited from its parent class.
* Allows for specialized behavior in child classes while maintaining a common interface.

Example:

Python

class Animal:

def make\_sound(self):

print("Generic animal sound")

class Dog(Animal):

def make\_sound(self): # Overridden method

print("Woof!")

class Cat(Animal):

def make\_sound(self): # Overridden method

print("Meow!")

animals = [Dog(), Cat(), Animal()]

for animal in animals:

animal.make\_sound() # Calls the appropriate version

Benefits of Polymorphism:

* Code Reusability: Write code that works with multiple types, reducing redundancy.
* Flexibility: Adapt code to different situations without extensive modifications.
* Maintainability: Easier to manage and extend code with polymorphic structures.
* Extensibility: Add new features without breaking existing code.

‘’’

Project:

Good Morning sir,

My self I am Sanjeev Yaswanth, I am from Andhra Pradesh, I completed my graduation in 2020 in ECE stream.

Currently I am working for Intel on behalf of Tech Mahindra Cerium systems from DEC 2020.

#In my current company,

Project is based on a RAILS (Reliable Automation Infrastructure Libraries and Services) framework which expects test cases to be automated in python using test steps.

Rails is about steps, cases and suite execution through a container like framework.

Project’s main primary purpose is to execute the test cases on the non-released Intel processors regressively to check the efficiency.

In this project, intel will be providing the non-released processors with respective platforms,

This project consists of multiple domains like audio, security, USB, power, storage, gaming, Bluetooth, display, graphics, sensor and many more domains.

And they will provide the testcases with different scenarios in these domains.

This project has 3 teams,

|  |
| --- |
| Development Team testing team deployment team |

So, as a developers we will pick those testcases and will build the testcase using test steps.

Will test end to end multiple time in different platforms and the code will be merged to dev environment

Now this TC will be released to testing team, they will test regressively on all the platforms and release to deployment team they will be testing multiple iterations and then the code will be deployed to production.

In middle, if in case TC fails at any step in testing or deployment, Bug Jira will be created on developers and we have to resolve it as soon as possible based on priority.

So, in this way we will be continuously build, test and merge the code, this process is known as continuous integration.

Testing or deployment team, regressively test and deliver or deploy the TC, this process is known as continuous deployment.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Coming to my role in this project is,

I am an internal lead as well as a developer to one of the domain in this project, my team strength is 3.

So, before every quarter starts, I will be scrubbing the testcases and shortlist few testcases for quarter,

And when new quarter begins, I need to plan the TC for sprint wise.

\* will be assigning 1 or 2 TCs to the developer,

And I will be creating the task Jira’s and every day I will be taking standup meetings with my team,

noting the status of every task and helping the blockers in meeting if any,

So, these updates I will be giving to our team lead. And I will be continuing my work.

\*In this domain, I have implemented one scenario from scratch i.e. Flashing IFWI(Integrated FirmWare Image) from OS and from UEFI shell through automation using FFT (firmware flashing Tool) and FPT(Intel Flash Programming Tool) intel tools \*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Here in this project, I involved in implementation, end to end testing and leading a small team.

At the beginning of sprint, I will be

Implementation &amp; testing of individual test steps along with E2E run and delivery of the complete testcase.

* Involved in implementation of Flashing IFWI from OS and from UEFI shell through automation using FFT and FPT tool as a part of Security Domain.

#

# project is about regressive automation testing on non released Intel processors using Python.

# Automation is based on RAILS (Reliable Automation Infrastructure Libraries and Services) framework,

This project based on a RAILS framework which expects test cases to be automated in python using

test steps.

Rails is about steps, cases and suite execution through a container like framework

The primary purpose of the project is to execute the test cases on the non-released mother

boards to regress the efficiency of the mother board.

will test each and every step manually first and then we will proceed to implementation using the RAILS framework and then end to end testing multiple times in different platforms.

And then we will merge the code to dev environment, this is known as continuous integration.

This code will be tested by deployment team regressively, and deploy the code after continuous iteration.

And I will be working on my own tasks, in middle of the testcase, if there are any bugs reported from the deployment team, I will be working on those bugs based on the priorities.

Here I will be involved in both implementation and testing the testcases and as a lead I will be tracking my team from blockers or any help

Pycharm starts

class accessVariables:

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

self.var1 = "aaaaaa" # public

self.\_var2 = "bbbb" # protected

self.\_\_var3 = "cccc" # private

def example(self):

print(f"public: {self.var1}, protected:"

f" {self.\_var2}, private: {self.\_\_var3}")

def get\_privateValue(self): #getter

return self.\_\_var3

def set\_privateValue(self, newData): #setter

self.\_\_var3 = newData

# a=accessVariables()

# a.example()

# print(a.get\_privateValue())

# a.set\_privateValue("ffffff")

# a.example()

# print(a.get\_privateValue())

# print(a.var1)

# a.var1 = "dddddd"

# a.example()

# print(a.\_var2)

# a.\_var2 = "eeeeeee"

# a.example()

# # a.\_\_var3 = "ffff"

# a.example()

# print(a.\_accessVariables\_\_var3) # able to access private variables

# # print(a.\_\_var3)

# a.\_accessVariables\_\_var3 = "ffffffffff"

# print(a.\_accessVariables\_\_var3)

# a.example()

# if \_\_name\_\_ == '\_\_main\_\_':

# a=accessVariables()

# a.example()

# print(a.var1, "1111",) #a.\_var2, "222", a.\_\_var3)

# print(a.\_var2)

# print(a.\_\_var3)

# def gen(num1, num2):

# string1 = "this is interview"

# sample = []

# # string2 = ""

# str3 = string1.split(" ")

# print(str3)

# for i in str3:

# result = i[::-1]

# sample.append(result)

# print(sample)

# print(" ".join(sample))

# [12:00 PM] Mahendra (Guest)

# s = "aabbbcceddbb"

# #"2a3b2c1e2d2b"

#

# aCount = s.count("a")

# bCount = s.count("b")

# cCount = s.count("c")

# dCount = s.count("d")

# eCount = s.count("e")

# print(s.split())

# data = [10, 20, 30, 40, 30, 22, 30,40,30,50,30]

# resultData = []

# count = 0

# for i in data:

# if i == 30:

# # print(i)

# count +=1

#

# if count == 2 or count==3:

# continue

# resultData.append(i)

# # print(data.index(30))

# # data.remove(30)

# print(resultData)

#

# polymorpism examples

# abstact

# encapsualtion

# generator

# decorator

# oops concepts

# public private protected

# polymorpism examples

# poly - many morp - forms

# 1.Class & object Polymorphism in Python or Duck-Typing

# def emptyLand(obj):

# obj.land()

#

# class office:

# def land(self):

# print("This land is for Office")

#

# class hotel:

# def land(self):

# print("this land is for hotel")

#

# off = office()

# hot = hotel()

#

# emptyLand(off)

# emptyLand(hot)

# 2. inheritance and method overriding

# class Animal:

# def make\_sound(self):

# print("Generic animal sound")

#

#

# class Dog(Animal):

# def make\_sound(self): # Overridden method

# print("Boww!")

#

#

# class Cat(Animal):

# def make\_sound(self): # Overridden method

# print("Meow!")

#

#

# animals = [Dog(), Cat(), Animal()]

# for animal in animals:

# animal.make\_sound() # Calls the appropriate version

# c = Animal()

# c.make\_sound()

# b=Dog()

# b.make\_sound()

# 3. inbuilt polymorphic functions

# len() being used for a string

# print(len("geeks"))

#

# # len() being used for a list

# print(len([10, 20, 30]))

#

#

# # 4. operator overriding

# class marks:

# def \_\_init\_\_(self, a, b):

# self.a = a

# self.b = b

#

# def \_\_add\_\_(self, other):

# a1 = self.a + other.a

# a2 = self.b + other.b

# return a1, a2

#

# def \_\_gt\_\_(self, other):

# b1 = self.a + self.b

# b2 = other.a + other.b

# if b1 > b2:

# return True

# else:

# return False

#

# def \_\_str\_\_(self):

# return f'{self.a}, {self.b}'

#

# s1 = marks(10, 20)

# s2 = marks(12, 15)

#

# total = s1 + s2

# print(total)

#

# greater = s1 > s2

# if greater:

# print("s1 wins")

# else:

# print("s2 wins")

#

# print(s1, s2) # \_\_str\_\_

#

# Python doesn't support method overloading,

# it will take the latest function,

# if we have same function name

class company:

def interview(self):

print('This is first Round')

def interview(self, round):

print(f"This is {round} round")

def interview(self, round, qualifyStatus):

print(f"This is {round} and you are {qualifyStatus}")

a = company()

# a.interview()

# a.interview(round="Technical")

# a.interview(round="HR", qualifyStatus="Qualified")

#abstract class & method

from abc import ABC, abstractmethod

class vehicle(ABC):

# def \_\_init\_\_(self, n):

# self.num\_of\_Tyres = n

@abstractmethod

def start(self, n):

pass

class bike(vehicle):

# def \_\_init\_\_(self, n):

# self.num\_of\_tyres = n

def start(self, n):

print("starts with kick", f"and number of tyres: {n}")

class scooty(vehicle):

# def \_\_init\_\_(self, n):

# self.num\_of\_tyres = n

def start(self, n):

print("starts with self button", f"and number of tyres: {n}")

a=bike()

a.start(2)

#Decorators

def div(a,b):

print(a/b)

def smart\_div(func):

def inner\_div(a,b):

if a<b:

a, b = b, a

return func(a,b)

return inner\_div

div1 = smart\_div(div)

div1(2,4)

#Generators

def gen():

yield 1

yield 2

yield 3

yield 4

values = gen()

# print(values.\_\_next\_\_())

# print(values.\_\_next\_\_())

# print(values.\_\_next\_\_())

# print(values.\_\_next\_\_())

# for i in values:

# print(i)

#Generators

def square(n):

for i in range(n):

sq = i\*i

yield sq

s = square(10)

# print(s.\_\_next\_\_())

# print(s.\_\_next\_\_())

print("-----------------------------")

for i in s:

print(i)

def fib(n):

a, b = 0, 1

while a < n:

yield a

a, b = b, a+b

a= fib(5)

# print(a.\_\_next\_\_())

# print(a.\_\_next\_\_())

print("-----------------------------")

for i in a:

print(i)

#iterator

l1 = [2,45,1,7,3,0]

a= iter(l1)

# print(a.\_\_next\_\_())

# print(a.\_\_next\_\_())

print("-----------------------------")

# class iterator:

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

# self.num = 1

#

# def \_\_iter\_\_(self):

# return self

#

# def \_\_next\_\_(self):

# if self.num <=10:

# val = self.num

# self.num+=1

# return val

# else:

# raise StopIteration

# z= iterator()

# print(z.\_\_next\_\_())

# print(z.\_\_next\_\_())

# print(z.\_\_next\_\_())

# for i in z:

# print(i)

#mutli level inheritance

class a:

def x(self):

print("x")

class b(a):

def y(self):

print("y")

class c(b):

def z(self):

print("z")

f=c()

f.z()

f.y()

f.x()

#decorator

def my\_logging\_decorator(func):

def wrapper(\*args, \*\*kwargs):

print(f"Calling function {func.\_\_name\_\_} with arguments: {args}, {kwargs}")

result = func(\*args, \*\*kwargs)

print(f"Function {func.\_\_name\_\_} returned: {result}")

return result

return wrapper

@my\_logging\_decorator

def add(x, y, z):

return x + y + z

add(5, 3, 2) # Output:

# Calling function add with arguments: (5, 3), {}

# Function add returned: 8

def hello\_decorator(func):

def inner1(\*args, \*\*kwargs):

print("before Execution")

# getting the returned value

returned\_value = func(\*args, \*\*kwargs)

print("after Execution of", func.\_\_name\_\_)

# returning the value to the original frame

return returned\_value

return inner1

# adding decorator to the function

@hello\_decorator

def sum\_two\_numbers(a, b):

print("Inside the function")

return a + b

@hello\_decorator

def div(a,b):

if a<b:

a,b = b,a

return a/b

a, b = 1, 2

# getting the value through return of the function

print("Sum =", sum\_two\_numbers(a, b))

print(f"division = {div(2,4)}")

'''The key difference between `return` and `yield` in Python lies in how they control the execution flow and data generation within a function:

\*\*Return:\*\*

- Exits the function immediately.

- Returns a single value to the caller.

- Can only be used once per function invocation.

- Used typically when a function completes a calculation and delivers a final result.

\*\*Yield:\*\*

- Temporarily pauses the function's execution.

- Returns a value to the caller but retains the function's state.

- Can be used multiple times within a function, generating a sequence of values.

- Used when you want to produce a stream of values on demand, often for large datasets or iterators.

Here's a table summarizing the key differences:

| Feature | `return` | `yield` |

|--------------------|-------------------------------------------------------|----------------------------------------------------|

| Execution | Ends function immediately | Pauses execution temporarily |

| Returned value | Single value to the caller | Single value to the caller, retains function state |

| Usages per call | One usage per function call | Multiple usages per function call |

| When to use | Returning final result, exiting function | Generating sequences, memory efficiency |

\*\*Example:\*\*

```python

def calculate\_squares\_return(n):

squares = []

for i in range(n):

squares.append(i \* i)

return squares

def calculate\_squares\_yield(n):

for i in range(n):

yield i \* i

# Using return

squares\_list = calculate\_squares\_return(5)

print(squares\_list) # Output: [0, 1, 4, 9, 16]

# Using yield

for square in calculate\_squares\_yield(5):

print(square) # Output: 0, 1, 4, 9, 16

```

\*\*In essence:\*\*

- Use `return` when you're done with the function and want to send a single value back.

- Use `yield` when you want to create a sequence of values on demand, making your function an iterator.'''