**Assessment Question**

**Task 1: Python Programming Basics**

1. **Describe the difference between mutable and immutable types in Python. Provide examples.**

**Answer:** There are two types of datatypes in python.

1. Mutable
2. Immutable
3. **Mutable Datatypes**

* In these datatypes values can be changed after the object is created.
* When we modify a mutable object, the object itself is altered, not new object is created.
* Examples:
  + Lists: list 🡪 my\_list = [1, 2, 3]
  + Dictionaries: dict 🡪 my\_dict = {'a': 1, 'b': 2}
  + Sets: set 🡪 my\_set = {1, 2, 3}

**Code:**

my\_list = [1, 2, 3]

my\_list[0] = 10

print(my\_list)

**Output:**



1. **Immutable Datatypes**

* In these datatypes values cannot be changed once the object is created.
* When we try to modify an immutable object, a new object is created instead.
* Examples:
  + Strings: str 🡪 my\_string = "hello"
  + Tuples: tuple 🡪 my\_tuple = (1, 2, 3)
  + Integers: int 🡪 x = 5
  + Floats: float 🡪 y = 3.14

**Code:**

my\_string = "hello"

my\_string = my\_string.replace('h', 'j')

print(my\_string)

**Output:**

****

1. **Write a Python program to check if a number is a palindrome.**

**Answer:** The below program shows the number is a palindrome or not. Also, I have share the file for the same.

def is\_palindrome(num):

    num\_str = str(num)

    if num\_str == num\_str[::-1]:

        return True

    else:

        return False

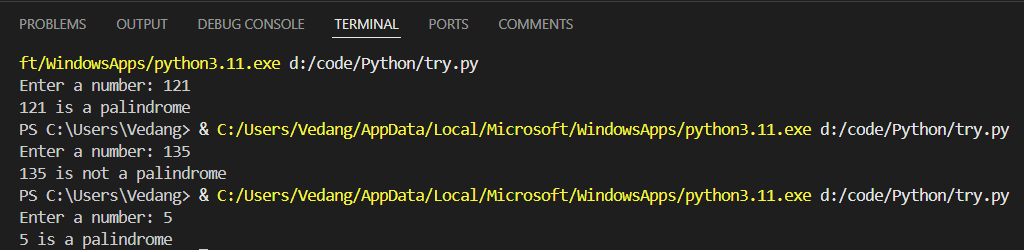
Number = int(input("Enter a number: "))

if is\_palindrome(Number):

    print(f"{Number} is a palindrome")

else:

    print(f"{Number} is not a palindrome")



1. **What are Python decorators? Provide an example of how to use one.**

**Answer:** A decorator is a function that takes another function as input and extends or modifies its behaviour without changing the original function.

OR

It is a function that allows you to modify or extend the behaviour of other functions or methods without changing their actual code. Decorators are commonly used for logging, access control, caching, and more.

In Python, decorators are implemented using higher-order functions. A decorator takes a function as an argument, adds some functionality, and then returns a new function (which usually calls the original function and adds some extra behaviour).

Also, I have shared the file of code for the same.

**Code:**

def my\_decorator(func):

    def wrapper():

        print("Before the function is called.")

        func()

        print("After the function is called.")

    return wrapper

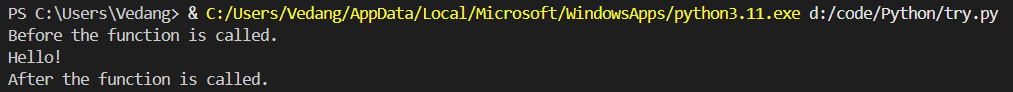
@my\_decorator

def say\_hello():

    print("Hello!")

say\_hello()

**Output:**



1. **Explain the purpose of \_\_init\_\_ in Python classes.**

**Answer:** It is special method that acts as constructor for class. The \_\_init\_\_ method in Python classes is used to initialize an object's attributes when an instance of the class is created. It is often referred to as the constructor, although the actual object creation is handled by the \_\_new\_\_ method. The \_\_init\_\_ method sets up the initial state of the object by defining its attributes and their values.

**Code:**

class Person:

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

        self.name = name

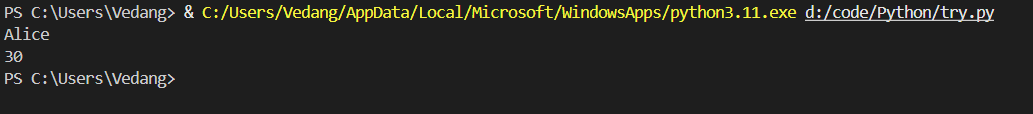
        self.age = age

person1 = Person("Alice", 30)

print(person1.name)

print(person1.age)

**Output:**



1. **What is the difference between a list and a tuple? When would you use each?**

**Answer:**

* **List:**
* Lists are mutable, meaning you can modify them after creation
* Lists are slower than tuples because of the overhead associated with their mutability.
* Defined using square brackets []
* Lists come with a variety of methods, such as append(), remove(), and sort(), allowing for manipulation of their contents.

**Code:**

fruits = ["apple", "banana", "cherry"]

fruits.append("orange")

print(fruits)

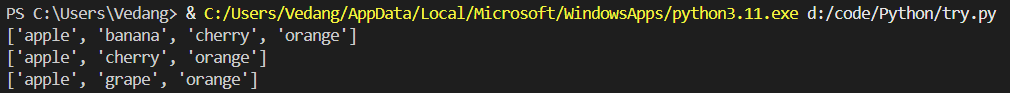
fruits.remove("banana")

print(fruits)

fruits[1] = "grape"

print(fruits)

**Output:**



* **Tuples:**
* Tuples are immutable, meaning once a tuple is created, its contents cannot be changed.
* Tuples are faster and more memory-efficient due to their immutability.
* Defined using parentheses () or by simply separating items with commas.
* Since tuples are immutable, they have fewer methods available, primarily count() and index().

**Code:**

coordinates = (10, 20)

print(coordinates[0])

print(coordinates[1])

**Output:**

