Name:- Limbasiya Vishvash

Module:- 3 Advance Python Programming

Q.1: Introduction to the print() function in Python.

Ans:-

* It can print **strings, numbers, variables, and**

print("Hello, World!")

print(5 + 3)

**Example with end:**

print("Hello", end=" ")

print("World!")

Q.2:- Formatting outputs using f-strings and format().

Ans:-

1. Using f-strings (Python 3.6+):

name = ""Vishvash

age = 25

print(f"My name is {name} and I am {age} years old.")

1. Using .format() method:

name = "Vishvash"

age = 25

print("My name is {} and I am {} years old.".format(name, age))

Q.3:- Using the input() function to read user input from the keyboard.

Ans:-

variable = input("Enter something: ")

Example:

name = input("Enter your name: ")

print(f"Hello, {name}!")

Converting to numbers:

age = int(input("Enter your age: "))

Q.4:- Converting user input into different data types (e.g., int, float, etc.).

Ans:-

**To Integer (int):**

num = int(input("Enter an integer: "))

print(num + 5)

**To Float (float):**

decimal = float(input("Enter a decimal number: "))

print(decimal \* 2)

Q.5:- Opening files in different modes ('r', 'w', 'a', 'r+', 'w+').

Ans:-

| **Mode** | **Description** |
| --- | --- |
| ‘r’ | Open text file for reading. Raises an I/O error if the file does not exist. |
| ‘r+’ | Open the file for reading and writing. Raises an I/O error if the file does not exist. |
| ‘w’ | Open the file for writing. Truncates the file if it already exists. Creates a new file if it does not exist. |
| ‘w+’ | Open the file for reading and writing. Truncates the file if it already exists. Creates a new file if it does not exist. |
| ‘a’ | Open the file for writing. The data being written will be inserted at the end of the file. Creates a new file if it does not exist. |
| ‘a+’ | Open the file for reading and writing. The data being written will be inserted at the end of the file. Creates a new file if it does not exist. |

Q.6:- Using the open() function to create and access files.

Ans:-

file = open("data.txt", "w")

file.write("Hello World")

file.close()

Use **with open(...) as f:** for auto close:

with open("data.txt", "r") as file:

print(file.read())

Q.7:- Closing files using close()

Ans:-

Closing Files Using close():

f = open("data.txt", "r")

print(f.read())

f.close()

Q.8:- Reading from a file using read(), readline(), readlines().

Ans:-

f = open("file.txt", "r")

print(f.read()) # Full content

print(f.readline()) # First line

print(f.readlines()) # List of lines

f.close()

Q.9:- Writing to a file using write() and writelines().

Ans:-

Writing to a File:

f = open("file.txt", "w")

f.write("Hello World\n")

f.writelines(["Line1\n", "Line2\n"])

f.close()

Q.10:- Introduction to exceptions and how to handle them using try, except, and finally.

Ans:-

try:

x = 10 / 0 # Causes ZeroDivisionError

except ZeroDivisionError:

print("Cannot divide by zero!")

**try-except with Exception Object**

try:

num = int("abc") # Causes ValueError

except ValueError as e:

print(f"Error occurred: {e}")

**finally (Always Executes, Used for Cleanup)**

try:

file = open("example.txt", "r")

except FileNotFoundError:

print("File not found!")

finally:

print("This will always execute.")

Q.11:- Understanding multiple exceptions and custom exceptions.

Ans:-

class MyCustomError(Exception):

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

super().\_\_init\_\_(message)

try:

raise MyCustomError("This is a custom error!")

except MyCustomError as e:

print(f"Caught custom exception: {e}")

Q.12:- Understanding the concepts of classes, objects, attributes, and methods in Python.

Ans:-

class Car:

def \_\_init\_\_(self, brand, model):

self.brand = brand # Attribute

self.model = model # Attribute

def display\_info(self): # Method

print(f"Car: {self.brand} {self.model}")

my\_car = Car("Toyota", "Corolla")

my\_car.display\_info()

class Dog:

species = "Canine"

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

self.name = name

dog1 = Dog("Buddy")

print(dog1.name, dog1.species)

class MathOperations:

@staticmethod

def add(a, b):

return a + b

print(MathOperations.add(5, 3))

Q.13:- Difference between local and global variables.

Ans:-

1. Local Variables:-

def my\_function():

x = 10

print(x)

my\_function()

1. Global Variables:-

y = 20 # Global variable

def my\_function():

global y

y += 5

print(y)

my\_function()

print(y)

Q.14:- Single, Multilevel, Multiple, Hierarchical, and Hybrid inheritance in Python.

Ans:-

1.**Single Inheritance**

* A child class inherits from a single parent class.
* **Example:**

class Parent:

def func1(self):

print("This is Parent class")

class Child(Parent):

def func2(self):

print("This is Child class")

obj = Child()

obj.func1() # Inherited method

obj.func2()

2.Multilevel Inheritance:-

class Grandparent:

def func1(self):

print("This is Grandparent")

class Parent(Grandparent):

def func2(self):

print("This is Parent")

class Child(Parent):

def func3(self):

print("This is Child")

obj = Child()

obj.func1() # Inherited from Grandparent

obj.func2() # Inherited from Parent

obj.func3()

1. Multiple Inheritance:-

class Parent1:

def func1(self):

print("This is Parent1")

class Parent2:

def func2(self):

print("This is Parent2")

class Child(Parent1, Parent2):

def func3(self):

print("This is Child")

obj = Child()

obj.func1() # From Parent1

obj.func2() # From Parent2

obj.func3()

1. Hierarchical Inheritance:-

class Parent:

def func1(self):

print("This is Parent")

class Child1(Parent):

def func2(self):

print("This is Child1")

class Child2(Parent):

def func3(self):

print("This is Child2")

obj1 = Child1()

obj1.func1() # Inherited from Parent

obj1.func2()

obj2 = Child2()

obj2.func1() # Inherited from Parent

obj2.func3()

1. Hybrid Inheritance:-

class Parent:

def func1(self):

print("This is Parent")

class Child1(Parent):

def func2(self):

print("This is Child1")

class Child2(Parent):

def func3(self):

print("This is Child2")

class GrandChild(Child1, Child2):

def func4(self):

print("This is GrandChild")

obj = GrandChild()

obj.func1() # From Parent

obj.func2() # From Child1

obj.func3() # From Child2

obj.func4()

Q.15:- Using the super() function to access properties of the parent class.

Ans:-

class Parent:

def show(self):

print("This is Parent class")

class Child(Parent):

def show(self):

super().show() # Calling the parent class method

print("This is Child class")

obj = Child()

obj.show()

Q.16:- Method overloading: defining multiple methods with the same name but different parameters.

Ans:-

class Demo:

def show(self, a=None, b=None):

if a is not None and b is not None:

print(f"Two arguments: {a}, {b}")

elif a is not None:

print(f"One argument: {a}")

else:

print("No arguments")

obj = Demo()

obj.show() # No arguments

obj.show(10) # One argument

obj.show(10, 20) # Two arguments

Q.17:- Method overriding: redefining a parent class method in the child class.

Ans:-

class Parent:

def show(self):

print("This is Parent class")

class Child(Parent):

def show(self): # Overriding the show() method

print("This is Child class")

obj = Child()

obj.show() # Calls the overridden method from Child

Q.18:- Using re.search() and re.match() functions in Python’s re module for pattern matching.

Ans:-

1. re.match():-

import re

text = "Hello World"

match = re.match(r"Hello", text) # Matches because "Hello" is at the start

print(match.group() if match else "No match")

match = re.match(r"World", text) # No match because "World" is not at the start

print(match.group() if match else "No match")

1. re.search():-

import re

text = "Hello World"

search = re.search(r"World", text) # Finds "World" anywhere in the text

print(search.group() if search else "No match")

Q.19:- Difference between search and match.

Ans:-

| **Feature** | **re.match()** | **re.search()** |
| --- | --- | --- |
| **Search Scope** | Only at the **beginning** of the string | Anywhere in the string |
| **Returns** | First match (if at the start) | First match found anywhere |
| **Fails If** | Pattern is not at the start | Pattern is not in the string |
| **Use Case** | Check if a string starts with a pattern | Find a pattern anywhere in the string |