LIST OF EXPERIMENTS

**Experiment 1: Assignment Statement**

**Theory:**

Assignment statements are used to assign values to variables.

**Code:**

x = 5

y = "Hello"

**Experiment 2: Conditional Statement (if-else)**

**Theory:**

Conditional statements allow the execution of different code based on conditions.

**Code:**

python

age = 20

if age >= 18:

print("You are an adult.")

else:

print("You are not an adult.")

**Experiment 3: Looping with for**

**Theory:**

for loops iterate over a sequence.

**Code:**

python

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

for fruit in fruits:

print(fruit)

**Experiment 4: Looping with while**

**Theory:**

while loops repeatedly execute a block of code while a condition is True.

**Code:**

python

count = 0

while count < 5:

print(count)

count += 1

**Experiment 5: Break and Continue Statements**

**Theory:**

break and continue alter the flow of loops.

**Code:**

python

for i in range(10):

if i == 5:

break # exits the loop when i equals 5

if i % 2 == 0:

continue # skips even numbers

print(i)

**Experiment 6: Function Definition**

**Theory:**

Functions are defined using the def keyword.

Code:

python

def greet(name):

return f"Hello, {name}!"

message = greet("Alice")

print(message)

**Experiment 7: Return Statement**

**Theory:**

The return statement is used to return a value from a function.

Code:

python

def multiply(x, y):

return x \* y

result = multiply(3, 4)

print(result)

**Experiment 8: Pass Statement**

**Theory:**

pass is a null operation; nothing happens when it executes.

Code:

python

if 1 < 2:

pass # no operation

else:

print("This won't execute.")

**Experiment 9: Try-Except Statement**

**Theory:**

try and except statements handle exceptions.

Code:

python

try:

result = 10 / 0

except ZeroDivisionError:

print("Cannot divide by zero.")

**Experiment 10: Class Definition**

**Theory:**

Classes are used to create objects in Python.

Code:

python

class Car:

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

self.brand = brand

def display\_info(self):

print(f"This car is from {self.brand}.")

car1 = Car("Toyota")

car1.display\_info()

**Experiment 11: Class Inheritance**

**Theory:**

Inheritance allows one class to inherit attributes and methods from another.

Code:

python

class ElectricCar(Car):

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

super().\_\_init\_\_(brand)

self.battery = battery

def display\_battery\_info(self):

print(f"This electric car has a {self.battery} kWh battery.")

electric\_car = ElectricCar("Tesla", 75)

electric\_car.display\_info()

electric\_car.display\_battery\_info()

**Experiment 12: Class Method**

**Theory**:

Methods within classes take the instance as their first argument.

Code:

python

class Person:

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

self.name = name

@classmethod

def create(cls, name):

return cls(name)

person = Person.create("John")

print(person.name)

**Experiment 13: Using in Membership Test**

**Theory:**

in checks for membership in a sequence.

Code:

python

my\_list = [1, 2, 3, 4, 5]

if 3 in my\_list:

print("3 is present in the list.")

**Experiment 14: Multiple Assignments**

**Theory:**

Multiple assignments allow assigning multiple variables in a single line.

Code:

python

x, y, z = 10, 20, 30

print(x, y, z)

**Experiment 15: Enumerate Statement**

**Theory:**

enumerate is used to iterate over indices and elements of a sequence simultaneously.

Code:

python

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

for index, fruit in enumerate(fruits):

print(f"Index {index}: {fruit}")

**Experiment 16: Using range() Function**

**Theory:**

range() generates a sequence of numbers.

Code:

python

for i in range(1, 6):

print(i)

**Experiment 17: String Methods**

**Theory:**

Strings have various built-in methods for manipulation.

Code:

python

my\_string = "Hello, World!"

print(my\_string.upper())

print(my\_string.split(','))

**Experiment 18: List Methods**

**Theory:**

Lists have various built-in methods for manipulation.

Code:

python

my\_list = [1, 2, 3]

my\_list.append(4)

print(my\_list)

**Experiment 19: Dictionary Methods**

**Theory:**

Dictionaries have various built-in methods for manipulation.

**Code:**

python

my\_dict = {'name': 'Alice', 'age': 30}

my\_dict.update({'city': 'New York'})

print(my\_dict)

**Experiment 20: Set Methods**

**Theory:**

Sets have various built-in methods for manipulation.

**Code:**

python

my\_set = {1, 2, 3}

my\_set.add(4)

print(my\_set)