1.Write a Python program to check whether the no is even and odd.

```
def check_even_odd(number):
    if number % 2 == 0:
        print(number, "is even.")
    else:
        print(number, "is odd.")

# Example usage
num = int(input("Enter a number: "))
check_even_odd(num)
```

output:

Enter a number: 25

25 is odd.

Enter a number: 12

12 is even.

2.Write a Python program to display Greater no between 3 inputs.

```
def find_greatest_number(a, b, c):
    if a >= b and a >= c:
        return a
    elif b >= a and b >= c:
        return b
    else:
        return c

# Example usage
num1 = int(input("Enter the first number: "))
num2 = int(input("Enter the second number: "))
num3 = int(input("Enter the third number: "))

greatest_num = find_greatest_number(num1, num2, num3)
print("The greatest number is:", greatest_num)
```

output:

```
Enter the first number: 25
Enter the second number: 12
Enter the third number: 36
The greatest number is: 36
```

3.Write a Python program to print prime no between n and m.

```
def is_prime(num):
    if num <= 1:
        return False

    for i in range(2, int(num ** 0.5) + 1):
        if num % i == 0:
            return False
    return True
def print_prime_numbers(n, m):
    print("Prime numbers between", n, "and", m, "are:")
    for num in range(n, m + 1):
        if is_prime(num):
            print(num, end=" ")

# Example usage
lower_limit = int(input("Enter the lower limit: "))
upper_limit = int(input("Enter the upper limit: "))
print_prime_numbers(lower_limit, upper_limit)
```

output:

```
Enter the lower limit: 10
Enter the upper limit: 30
Prime numbers between 10 and 30 are:
11 13 17 19 23 29
```

4. Write a Python program to display Fibonacci series.

```
def fibonacci_series(n):
    fibonacci = [0, 1]  # Initialize the series with the first two numbers

    if n <= 0:
        return fibonacci[:1]  # Return [0] for n <= 0
    elif n == 1:
        return fibonacci  # Return [0, 1] for n = 1

    # Generate the Fibonacci series
    while len(fibonacci) < n:
        next_num = fibonacci[-1] + fibonacci[-2]
        fibonacci.append(next_num)

    return fibonacci

# Example usage
num_terms = int(input("Enter the number of terms in the Fibonacci series: "))

fib_series = fibonacci_series(num_terms)
print("The Fibonacci series up to", num_terms, "terms is:")
for num in fib_series:
    print(num, end=" ")
```

output:

```
Enter the number of terms in the Fibonacci series: 10
The Fibonacci series up to 10 terms is:
0 1 1 2 3 5 8 13 21 34
```

5. Write a Python program factorial of no.

```
def factorial(num):
    if num == 0 or num == 1:
        return 1

    result = 1
    for i in range(2, num + 1):
        result *= i

    return result

# Example usage
num = int(input("Enter a number: "))

fact = factorial(num)
print("The factorial of", num, "is:", fact)
```

output:

Enter a number: 5

The factorial of 5 is: 120

6. Write a Python script to generate and print a dictionary that contains a number (between 1 and n) in the form (x, x*x).

Sample Dictionary (n = 5) :

Expected Output : {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}

```
def generate_number_dict(n):
    number_dict = {}
    for x in range(1, n + 1):
        number_dict[x] = x * x
    return number_dict

# Example usage
n = int(input("Enter a number (n): "))

result_dict = generate_number_dict(n)
print("Generated Dictionary:", result_dict)
```

output:

```
Enter a number (n): 5
Generated Dictionary: {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

7. Write a Python script to concatenate following dictionaries to create a new one.

dic1={1:10, 2:20}

dic2={3:30, 4:40}

dic3={5:50,6:60}

Expected Result : {1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}

```
def concatenate_dictionaries(*dicts):
    result_dict = {}
    for d in dicts:
        result_dict.update(d)
    return result_dict

# Example usage
dic1 = {1: 10, 2: 20}
dic2 = {3: 30, 4: 40}
dic3 = {5: 50, 6: 60}

result_dict = concatenate_dictionaries(dic1, dic2, dic3)
print("Concatenated Dictionary:", result_dict)
```

output:

Concatenated Dictionary: {1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}

8. Write a Python program to perform star pattern.

*

**

***

****

```
def print_star_pattern(n):
    for i in range(1, n + 1):
        print('*' * i)

# Example usage
num_rows = int(input("Enter the number of rows: "))

print("Star Pattern:")
print_star_pattern(num_rows)
```

output:

```
Enter the number of rows: 4
Star Pattern:
*
**
***
****
```

9. Write a Python program to perform star Pattern.

****

***

**

*

```
def print_star_pattern(n):
    for i in range(n, 0, -1):
        print('*' * i)

# Example usage
num_rows = int(input("Enter the number of rows: "))

print("Star Pattern:")
print_star_pattern(num_rows)
```

output:

Enter the number of rows: 4

Star Pattern:

****

***

**

*

10. Write a Python program to construct the following pattern, using a nested loop number.

687954231

87954231

7954231

954231

54231

4231

231

31

1def print_pattern(n):

```
    for i in range(n, 0, -1):
        for j in range(i, n + 1):
            print(j, end='')
        print()
num_rows = int(input("Enter the number of rows: "))
print("Pattern:")
print_pattern(num_rows)
```

output: Enter the number of rows: 8 Pattern:

687954231

87954231

7954231

954231

54231

4231

231

31

1

13. Write a Python program to get a string from a given string where all occurrences of its first char have been changed to '$', except the first char itself

Sample String : 'restart'

Expected Result : 'resta$t'

```
def replace_first_char(string):
    first_char = string[0]
    modified_string = first_char + string[1:].replace(first_char, '$')
    return modified_string

# Example usage
input_string = input("Enter a string: ")

result_string = replace_first_char(input_string)
print("Modified string:", result_string)
```

output:

Enter a string: restart

Modified string: resta$t

14. Write a Python program to remove the characters which have odd index values of a given string

```
def remove_odd_characters(string):
    result = ''
    for i in range(len(string)):
        if i % 2 == 0:
            result += string[i]
    return result

# Example usage
input_string = input("Enter a string: ")

result_string = remove_odd_characters(input_string)
print("Modified string:", result_string)
```

output:

Enter a string: OpenAI

Modified string: OeA

15. Write a Python program to get a string made of the first 2 and the last 2 chars from a given a string. If the string length is less than 2, return instead of the empty string

Sample String : “Welcome to DYPSOMCA”

Expected Result : WTDYP

Sample String : 'We'

Expected Result : 'WeWe'

Sample String : ' w'

Expected Result : Empty String

```
def get_first_last_chars(string):
    if len(string) < 2:
        return ""
    first_chars = string[:2]
    last_chars = string[-2:]
    return first_chars + last_chars
# Example usage
input_string = input("Enter a string: ")
result_string = get_first_last_chars(input_string)
print("Modified string:", result_string)
```

output:

```
Enter a string: Welcome to DYPSOMCA
Modified string: WTDY
Enter a string: We
Modified string: WeWe
Enter a string:  w
Modified string: Empty String
```

16. Write a Python class to implement pow(x, n).

```
class PowerCalculator:
    def pow(self, x, n):
        if n == 0:
            return 1

        if n < 0:
            x = 1 / x
            n = -n
        result = 1
        while n > 0:
            if n % 2 == 1:
                result *= x
            x *= x
            n //= 2
        return result

# Example usage
calculator = PowerCalculator()
base = float(input("Enter the base (x): "))
exponent = int(input("Enter the exponent (n): "))

result = calculator.pow(base, exponent)
print("Result:", result)
```

output:

```
Enter the base (x): 2.5
Enter the exponent (n): 4
Result: 39.0625
```

17. Write a Python class named Circle constructed by a radius and two methods which will compute the area and the perimeter of a circle.

```
import math
class Circle:
    def __init__(self, radius):
        self.radius = radius

    def calculate_area(self):
        area = math.pi * self.radius**2
        return area

    def calculate_perimeter(self):
        perimeter = 2 * math.pi * self.radius
        return perimeter

# Example usage
radius = float(input("Enter the radius of the circle: "))

circle = Circle(radius)
area = circle.calculate_area()
perimeter = circle.calculate_perimeter()
print("Area:", area)
print("Perimeter:", perimeter)
```

output:

```
Enter the radius of the circle: 5
Area: 78.53981633974483
Perimeter: 31.41592653589793
```

18. Write a Python program to get the factorial of a non-negative integer using recursion.

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)

# Example usage
num = int(input("Enter a non-negative integer: "))

result = factorial(num)
print("Factorial:", result)
```

output:

Enter a non-negative integer: 6

Factorial: 720

19. Write a Python program to read first n lines of a file

```
def read_first_n_lines(filename, n):
    with open(filename, 'r') as file:
        lines = file.readlines()
        first_n_lines = lines[:n]
    return first_n_lines

# Example usage
filename = input("Enter the file name: ")
n = int(input("Enter the number of lines to read: "))

lines = read_first_n_lines(filename, n)
print("First", n, "lines of the file:")
for line in lines:
    print(line.strip())
```

output:

```
Enter the file name: example.txt
Enter the number of lines to read: 3
First 3 lines of the file:
Line 1
Line 2
Line 3
```

20. Write a Python program to read a file line by line store it into a variable.

```
def read_file(filename):
    with open(filename, 'r') as file:
        content = file.read()
    return content

# Example usage
filename = input("Enter the file name: ")

file_content = read_file(filename)
print("File content:")
print(file_content)
```

output:

```
Enter the file name: example.txt
File content:
Line 1
Line 2
Line 3
Line 4
```

21. Write a Python program to combine each line from first file with the corresponding line in second file.

```
def combine_files(file1, file2):
    combined_lines = []
    with open(file1, 'r') as f1, open(file2, 'r') as f2:
        lines1 = f1.readlines()
        lines2 = f2.readlines()
        min_lines = min(len(lines1), len(lines2))
        for i in range(min_lines):
            combined_line = lines1[i].strip() + ' ' + lines2[i].strip()
            combined_lines.append(combined_line)
    return combined_lines
# Example usage
file1 = input("Enter the first file name: ")
file2 = input("Enter the second file name: ")
combined_lines = combine_files(file1, file2)
print("Combined lines:")
for line in combined_lines:
    print(line)
```

output:

```
Enter the first file name: file1.txt
Enter the second file name: file2.txt
Combined lines:
Line 1 from file1 Line 1 from file2
Line 2 from file1 Line 2 from file2
Line 3 from file1 Line 3 from file2
```

22. Write a program in which you need to calculate size of rectangle, square, circle and Triangle.(Separate Package need to be created for the same)

rectangle.py

```
class Rectangle:
    def __init__(self, length, width):
        self.length = length
        self.width = width

    def calculate_area(self):
        return self.length * self.width

    def calculate_perimeter(self):
        return 2 * (self.length + self.width)
```

square.py

```
class Square:
    def __init__(self, side):
        self.side = side

    def calculate_area(self):
        return self.side ** 2

    def calculate_perimeter(self):
        return 4 * self.side
```

circle.py

```
import math

class Circle:
    def __init__(self, radius):
        self.radius = radius

    def calculate_area(self):
        return math.pi * self.radius ** 2

    def calculate_circumference(self):
        return 2 * math.pi * self.radius
```

triangle.py

```
class Triangle:
    def __init__(self, base, height):
        self.base = base
        self.height = height

    def calculate_area(self):
        return 0.5 * self.base * self.height

    def calculate_perimeter(self, side1, side2, side3):
        return side1 + side2 + side3
```

main.py

```
from shapes.rectangle import Rectangle
```

```
from shapes.square import Square
from shapes.circle import Circle
from shapes.triangle import Triangle

# Rectangle
rectangle = Rectangle(5, 10)
rectangle_area = rectangle.calculate_area()
rectangle_perimeter = rectangle.calculate_perimeter()

print("Rectangle:")
print("Area:", rectangle_area)
print("Perimeter:", rectangle_perimeter)

# Square
square = Square(7)
square_area = square.calculate_area()
square_perimeter = square.calculate_perimeter()

print("\nSquare:")
print("Area:", square_area)
print("Perimeter:", square_perimeter)

# Circle
circle = Circle(4)
circle_area = circle.calculate_area()
circle_circumference = circle.calculate_circumference()

print("\nCircle:")
print("Area:", circle_area)
```

```
print("Circumference:", circle_circumference)

# Triangle
triangle = Triangle(6, 8)
triangle_area = triangle.calculate_area()
triangle_perimeter = triangle.calculate_perimeter(5, 7, 9)

print("\nTriangle:")
print("Area:", triangle_area)
print("Perimeter:", triangle_perimeter)
```

output:

```
Rectangle:
Area: 50
Perimeter: 30

Square:
Area: 49
Perimeter: 28

Circle:
Area: 50.26548245743669
Circumference: 25.132741228718345

Triangle:
Area: 24.0
Perimeter: 21
```

23. Write a program for Decorator and Generator.

decorator_and_generator.py

```
# Decorator
def uppercase_decorator(func):
    def wrapper(*args, **kwargs):
        result = func(*args, **kwargs)
        return result.upper()
    return wrapper
# Decorated function
@uppercase_decorator
def greet(name):
    return f"Hello, {name}!"
# Generator
def countdown(n):
    while n > 0:
        yield n
        n -= 1
# Main function
def main():
    # Decorator example
    greeting = greet("John")
    print(greeting)  # Output: HELLO, JOHN!
    # Generator example
    countdown_generator = countdown(5)
    for number in countdown_generator:
        print(number)  #
```

Output:

5, 4, 3, 2, 1

```
if __name__ == "__main__":
    main()
```

output:

```
HELLO, JOHN!
5
4
3
2
1
```

24. Write a program to validate URL using regular expression.

```
import re

def validate_url(url):
    # Regular expression pattern for URL validation
    pattern = re.compile(
        r'^(https?://)?'  # Optional http(s):// prefix
        r'([a-zA-Z0-9-]+\.)*[a-zA-Z0-9-]+(\.[a-zA-Z]{2,})'  # Domain name
        r'(/[^\s]*)?$'  # Optional path
    )

    # Check if the URL matches the pattern
    if re.match(pattern, url):
        return True
    else:
        return False

# Testing the URL validation
url1 = 'https://www.example.com'
url2 = 'http://google.com'
url3 = 'ftp://ftp.example.com'
url4 = 'www.example.com'
url5 = 'invalid-url'

print(validate_url(url1))  # Output: True
print(validate_url(url2))  # Output: True
print(validate_url(url3))  # Output: False
print(validate_url(url4))  # Output: False
print(validate_url(url5))  # Output: False
```

output:

True

True

False

False

False

25. Write a program to validate email using regular expression.

```
import re
def validate_email(email):
    # Regular expression pattern for email validation
    pattern = re.compile(
        r'^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$'
    )
    # Check if the email matches the pattern
    if re.match(pattern, email):
        return True
    else:
        return False
# Testing the email validation
email1 = 'test@example.com'
email2 = 'john.doe@gmail.com'
email3 = 'invalid-email'
email4 = 'user@example'

print(validate_email(email1))  # Output: True
print(validate_email(email2))  # Output: True
print(validate_email(email3))  # Output: False
print(validate_email(email4))  # Output: False
```

output:

```
True
True
False
False
```

26. Write a program to validate Password using regular expression.

```
import re
def validate_password(password):
    # Regular expression pattern for password validation
    pattern = re.compile(
        r'^(?=.*[a-z])'  # At least one lowercase letter
        r'(?=.*[A-Z])'  # At least one uppercase letter
        r'(?=.*\d)'  # At least one digit
        r'(?=.*[!@#$%^&*()_\-+=])'  # At least one special character
        r'(?=.{8,})'  # Minimum length of 8 characters
    )
    # Check if the password matches the pattern
    if re.match(pattern, password):
        return True
    else:
        return False
password1 = 'Passw0rd!'
password2 = '12345'
password3 = 'Abcd123'
password4 = 'strongPassword123!'
print(validate_password(password1))  # Output: True
print(validate_password(password2))  # Output: False
print(validate_password(password3))  # Output: False
print(validate_password(password4))  # Output: True
```

output:

```
True
False
False
True
```

27. Write a program that ask for an integer number. Accepted number should be in between 1 to 10 and break the loop if not then generate the exception and print an error message.

```
while True:
   try:
      number = int(input("Enter an integer number between 1 and 10: "))
      if number < 1 or number > 10:
         raise ValueError("Number is out of range!")
      break
   except ValueError as error:
      print("Error:", str(error))

print("Number is:", number)
```

output:

```
Enter an integer number between 1 and 10: 5
Number is: 5
Enter an integer number between 1 and 10: 15
Error: Number is out of range!
Enter an integer number between 1 and 10: 0
Error: Number is out of range!
Enter an integer number between 1 and 10: 7
Number is: 7
```

28. Write a Python Program to perform synchronized Multi-threading.

```
import threading
# Shared variable
counter = 0
# Lock object for synchronization
lock = threading.Lock()
# Function to increment the counter
def increment():
    global counter
    for _ in range(1000000):
        # Acquire the lock
        lock.acquire()
        counter += 1
        # Release the lock
        lock.release()
# Create multiple threads
threads = []
for _ in range(5):
    t = threading.Thread(target=increment)
    threads.append(t)
# Start the threads
for t in threads:
    t.start()
# Wait for all threads to finish
for t in threads:
    t.join()
# Print the final value of the counter
print("Counter:", counter)
```

output: Counter: 5000000

29. Write a Program for Performing basic CRUD operations with MongoDB and python.

```
from pymongo import MongoClient

# Establish a connection to MongoDB
client = MongoClient('mongodb://localhost:27017')

# Access the database
db = client['mydatabase']

# Access the collection
collection = db['mycollection']

# Create operation
data = {'name': 'John Doe', 'age': 30, 'city': 'New York'}
result = collection.insert_one(data)
print('Insert ID:', result.inserted_id)

# Read operation
document = collection.find_one({'name': 'John Doe'})
print('Document:', document)

# Update operation
update_data = {'$set': {'age': 35}}
result = collection.update_one({'name': 'John Doe'}, update_data)
print('Matched Count:', result.matched_count)
print('Modified Count:', result.modified_count)
```

```
# Read operation after update
document = collection.find_one({'name': 'John Doe'})
print('Updated Document:', document)

# Delete operation
result = collection.delete_one({'name': 'John Doe'})
print('Deleted Count:', result.deleted_count)

# Read operation after delete
document = collection.find_one({'name': 'John Doe'})
print('Document after delete:', document)
```

output:

```
Insert ID: 61486c96d8c7c7a3be2d03e1
Document: {'_id': ObjectId('61486c96d8c7c7a3be2d03e1'), 'name': 'John Doe', 'age': 30, 'city': 'New York'}
Matched Count: 1
Modified Count: 1
Updated Document: {'_id': ObjectId('61486c96d8c7c7a3be2d03e1'), 'name': 'John Doe', 'age': 35, 'city': 'New York'}
Deleted Count: 1
Document after delete: None
```

30. Write a python Program Create Numpy Array with Random Values – numpy.random.rand()

```
import numpy as np


# Set the random seed for reproducibility (optional)

np.random.seed(0)


# Generate a 1-dimensional NumPy array with random values

array = np.random.rand(5)


# Print the array

print("Array:", array)
```

output:

```
Array: [0.5488135  0.71518937 0.60276338 0.54488318 0.4236548 ]
```

31. Write a Program Numpy array and Reverse the Array.

```
import numpy as np

# Create a NumPy array
array = np.array([1, 2, 3, 4, 5])

# Print the original array
print("Original Array:", array)

# Reverse the array
reversed_array = np.flip(array)

# Print the reversed array
print("Reversed Array:", reversed_array)
```

output:

```
Original Array: [1 2 3 4 5]
Reversed Array: [5 4 3 2 1]
```

32. Write a Programs for series and data frames.

```
import pandas as pd

# Create a Series
series = pd.Series([10, 20, 30, 40, 50])

# Print the Series
print("Series:")
print(series)
```

output:

```
Series:
0  10
1  20
2  30
3  40
4  50
dtype: int64
```

```
import pandas as pd

# Create a dictionary
data = {'Name': ['John', 'Emma', 'Mike', 'Sophia'],
        'Age': [25, 28, 30, 32],
        'City': ['New York', 'London', 'Paris', 'Sydney']}

# Create a Data Frame
df = pd.DataFrame(data)
```

```
# Print the Data Frame
print("Data Frame:")
print(df)
```

output:

```
Data Frame:
id   Name       Age      City
0   John        25         New York
1   Emma     28         London
2   Mike       30          Paris
3  Sophia    32          Sydney
```

33. Write a Program for data visualization using Matplotlib.

```
import matplotlib.pyplot as plt

# Data

x = [1, 2, 3, 4, 5]

y = [10, 20, 15, 25, 30]

# Plotting

plt.plot(x, y)

# Adding labels and title

plt.xlabel('X-axis')

plt.ylabel('Y-axis')

plt.title('Line Graph')

# Displaying the plot

plt.show()
```

output: