Name: Prashant Jalindar Wayal

Roll No: 21 (AI&ML)

Q 1. Write a python code to find given number is prime or not

In [43]:
```
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
num = 11
if is_prime(num):
    print(f"{num} is a prime number.")
else:
    print(f"{num} is not a prime number.")
```

```
11 is not a prime number.
```

Q 2.Write a python code to find LCM and GCM of a given list

In [41]:
```
import math

def find_lcm(numbers):
    lcm = numbers[0]
    for num in numbers[1:]:
        lcm = lcm * num // math.gcd(lcm, num)
    return lcm
def find_gcd(numbers):
    gcd = numbers[0]
    for num in numbers[1:]:
        gcd = math.gcd(gcd, num)
    return gcd
numbers = [12, 18, 24, 36]
lcm_result = find_lcm(numbers)
gcd_result = find_gcd(numbers)

print(f"LCM of {numbers} is {lcm_result}")
print(f"GCD of {numbers} is {gcd_result}")
```

```
LCM of [12, 18, 24, 36] is 72
GCD of [12, 18, 24, 36] is 6
```

Q 3.Write a python code to find mean and standard deviation of a given list of

numbers

In [2]:
```
import numpy as np
def find_mean_and_stddev(numbers):
    mean = np.mean(numbers)
    stddev = np.std(numbers)
    return mean, stddev

numbers = [13, 28, 14, 26, 22]

mean_result, stddev_result = find_mean_and_stddev(numbers)

print(f"Mean of the numbers is {mean_result:.2f}")
print(f"Standard Deviation is {stddev_result:.2f}")
```

```
Mean of the numbers is 20.60
Standard Deviation is 6.12
```

Q 4.Write a python code to add and delete element from a dictionary using functions

In [13]:
```
def add_element(dictionary, key, value):
    dictionary[key] = value

def delete_element(dictionary, key):
    if key in dictionary:
        del dictionary[key]
    else:
        print(f"Key '{key}' not found in the dictionary.")

my_dict = {'a': 1, 'b': 2, 'c': 3}

add_element(my_dict, 'd', 4)
print("Dictionary after adding 'd':", my_dict)

delete_element(my_dict, 'b')
print("Dictionary after deleting 'b':", my_dict)

delete_element(my_dict, 'e')
```

```
Dictionary after adding 'd': {'a': 1, 'b': 2, 'c': 3, 'd': 4}
Dictionary after deleting 'b': {'a': 1, 'c': 3, 'd': 4}
Key 'e' not found in the dictionary.
```

Q 5.Write a python code to print 10 student details using class and lists

In [3]:
```
class Student:
    def __init__(self, name, roll_number, grade):
        self.name = name
        self.roll_number = roll_number
        self.grade = grade
student_list = []

def add_student(name, roll_number, grade):
    student = Student(name, roll_number, grade)
    student_list.append(student)
```

```
def print_student_details(student):
    print(f"Name: {student.name}")
    print(f"Roll Number: {student.roll_number}")
    print(f"Grade: {student.grade}")
    print()
add_student("Aditya", 101, "A")
add_student("Bhasker", 102, "B")
add_student("Charlie", 103, "A")
add_student("David", 104, "C")
add_student("Evan", 105, "A")
add_student("Frank", 106, "B")
add_student("Grace", 107, "A")
add_student("Hank", 108, "C")
add_student("Ivy", 109, "B")
add_student("Jack", 110, "A")
for student in student_list:
    print_student_details(student)
```

```
Name: Aditya
Roll Number: 101
Grade: A

Name: Bhasker
Roll Number: 102
Grade: B

Name: Charlie
Roll Number: 103
Grade: A

Name: David
Roll Number: 104
Grade: C

Name: Evan
Roll Number: 105
Grade: A

Name: Frank
Roll Number: 106
Grade: B

Name: Grace
Roll Number: 107
Grade: A

Name: Hank
Roll Number: 108
Grade: C

Name: Ivy
Roll Number: 109
Grade: B

Name: Jack
Roll Number: 110
Grade: A
```

Q 6.Write a python code to find student from a given list using class

In [4]:
```
class Student:
    def __init__(self, name, roll_number, grade):
        self.name = name
        self.roll_number = roll_number
        self.grade = grade


student_list = []


def add_student(name, roll_number, grade):
    student = Student(name, roll_number, grade)
    student_list.append(student)


def find_student_by_name(name):
    for student in student_list:
        if student.name == name:
            return student
    return None

def find_student_by_roll_number(roll_number):
    for student in student_list:
        if student.roll_number == roll_number:
            return student
    return None


add_student("Aditya", 101, "A")
add_student("Bhasker", 102, "B")
add_student("Charlie", 103, "A")
add_student("David", 104, "C")


student_to_find_by_name = "Charlie"
found_student_by_name = find_student_by_name(student_to_find_by_name)

if found_student_by_name:
    print(f"Student found by name '{student_to_find_by_name}':")
    print(f"Name: {found_student_by_name.name}")
    print(f"Roll Number: {found_student_by_name.roll_number}")
    print(f"Grade: {found_student_by_name.grade}")
else:
    print(f"Student with name '{student_to_find_by_name}' not found.")


roll_number_to_find = 102
found_student_by_roll_number = find_student_by_roll_number(roll_number_to_find)

if found_student_by_roll_number:
    print(f"Student found by roll number {roll_number_to_find}:")
    print(f"Name: {found_student_by_roll_number.name}")
    print(f"Roll Number: {found_student_by_roll_number.roll_number}")
    print(f"Grade: {found_student_by_roll_number.grade}")
else:
    print(f"Student with roll number {roll_number_to_find} not found.")
```

```
Student found by name 'Charlie':
Name: Charlie
Roll Number: 103
Grade: A
Student found by roll number 102:
Name: Bhasker
Roll Number: 102
Grade: B
```

Q 7 Write a python code to inherit employee class to student class

In [16]:
```python
class Employee:
    def __init__(self, name, employee_id):
        self.name = name
        self.employee_id = employee_id

    def display_info(self):
        print(f"Name: {self.name}")
        print(f"Employee ID: {self.employee_id}")


class Student(Employee):
    def __init__(self, name, employee_id, student_id):
        super().__init__(name, employee_id)
        self.student_id = student_id

    def display_info(self):
        super().display_info()
        print(f"Student ID: {self.student_id}")


student = Student("Alice", "E123", "S456")


student.display_info()
```

```
Name: Alice
Employee ID: E123
Student ID: S456
```

Q.8.Display Fibonacci series up to 10 terms

In [18]:
```python
n_terms = 10
a, b = 0, 1
print(a, end=" ")
print(b, end=" ")
for _ in range(2, n_terms):
    next_term = a + b
    print(next_term, end=" ")
    a, b = b, next_term
```

```
0 1 1 2 3 5 8 13 21 34
```

Q.9.Find the factorial of a given number

In [19]:
```
def factorial(n):
    if n < 0:
        return "Factorial is not defined for negative numbers"
    elif n == 0 or n == 1:
        return 1
    else:
        result = 1
        for i in range(2, n + 1):
            result *= i
        return result
number = 5
result = factorial(number)
print(f"The factorial of {number} is {result}")
```

```
The factorial of 5 is 120
```

Q.10.Write a program to iterate a given list and count the occurrence of each element and create a dictionary to show the count of each element.

In [20]:
```
def count_elements(input_list):
    element_count = {}
    for element in input_list:
        if element in element_count:
            element_count[element] += 1
        else:
            element_count[element] = 1
    return element_count
my_list = [1, 2, 2, 3, 4, 1, 2, 3, 3, 5, 5]
element_count_dict = count_elements(my_list)
for element, count in element_count_dict.items():
    print(f"{element}: {count}")
```

```
1: 2
2: 3
3: 3
4: 1
5: 2
```

Q.11.Find the intersection (common) of two sets and remove those elements from the first set

In [21]:
```
set1 = {1, 2, 3, 4, 5}
set2 = {3, 4, 5, 6, 7}
intersection = set1.intersection(set2)
set1.difference_update(intersection)
print("Original Set 1 after removing common elements:")
print(set1)
```

```
Original Set 1 after removing common elements:
{1, 2}
```

Q.12.Get all values from the dictionary and add them to a list but don't add duplicates

In [22]:
```
my_dict = {
    'a': 1,
    'b': 2,
    'c': 3,
    'd': 2,
    'e': 4,
}
unique_values = []
for value in my_dict.values():
    if value not in unique_values:
        unique_values.append(value)
print("List of unique values from the dictionary:")
print(unique_values)
```

```
List of unique values from the dictionary:
[1, 2, 3, 4]
```

Q.13.Create a Cricle class and intialize it with radius. Make two methods getArea and getCircumference inside this class.

In [23]:
```
class Circle:
    def __init__(self, radius):
        self.radius = radius

    def get_area(self):
        area = 3.14159265359 * self.radius * self.radius
        return area

    def get_circumference(self):
        circumference = 2 * 3.14159265359 * self.radius
        return circumference
circle = Circle(5)

area = circle.get_area()
circumference = circle.get_circumference()

print(f"Radius: {circle.radius}")
print(f"Area: {area:.2f}")
print(f"Circumference: {circumference:.2f}")
```

```
Radius: 5
Area: 78.54
Circumference: 31.42
```

Q.14.Create a Temprature class. Make two methods : 1. convertFahrenheit - It will take celsius and will print it into Fahrenheit. 2. convertCelsius - It will take Fahrenheit and will convert it into Celsius.

In [24]:
```
class Temperature:
    def convert_fahrenheit(self, celsius):
        fahrenheit = (celsius * 9/5) + 32
        print(f"{celsius} degrees Celsius is equal to {fahrenheit:.2f} degrees Fahr

    def convert_celsius(self, fahrenheit):
        celsius = (fahrenheit - 32) * 5/9
        print(f"{fahrenheit} degrees Fahrenheit is equal to {celsius:.2f} degrees C
temp_converter = Temperature()
temp_converter.convert_fahrenheit(25)
temp_converter.convert_celsius(77)
```

```
25 degrees Celsius is equal to 77.00 degrees Fahrenheit
77 degrees Fahrenheit is equal to 25.00 degrees Celsius
```

Q.15.Create a Time class and initialize it with hours and minutes.1. Make a method addTime which should take two time object and add them. E.g.- (2 hour and 50 min)+(1 hr and 20 min) is (4 hr and 10 min) 2. Make a method displayTime which should print the time. 3. Make a method DisplayMinute which should display the total minutes in the Time. E.g.- (1 hr 2 min) should display 62 minute.

In [25]:
```
class Time:
    def __init__(self, hours, minutes):
        self.hours = hours
        self.minutes = minutes

    def add_time(self, other_time):
        total_hours = self.hours + other_time.hours
        total_minutes = self.minutes + other_time.minutes
        if total_minutes >= 60:
            total_hours += total_minutes // 60
            total_minutes %= 60

        return Time(total_hours, total_minutes)

    def display_time(self):
        print(f"{self.hours} hr {self.minutes} min")

    def display_minutes(self):
        total_minutes = (self.hours * 60) + self.minutes
        print(f"{total_minutes} minutes")
time1 = Time(2, 50)
time2 = Time(1, 20)
sum_time = time1.add_time(time2)
print("Sum of times:")
sum_time.display_time()
sum_time.display_minutes()
```

```
Sum of times:
4 hr 10 min
250 minutes
```

Q.16.Write a function “perfect()” that determines if parameter number is a perfect number. Use this function in a program that determines and prints all the perfect numbers between 1 and 1000. [An integer number is said to be “perfect number” if its factors, including 1(but not the number itself), sum to the number. E.g., 6 is a perfect number because 6=1+2+3].

In [26]:
```
def perfect(number):
    divisors_sum = 1

    for i in range(2, number // 2 + 1):
        if number % i == 0:
            divisors_sum += i

    return divisors_sum == number
print("Perfect numbers between 1 and 1000:")
for num in range(1, 1001):
    if perfect(num):
        print(num)
```

```
Perfect numbers between 1 and 1000:
1
6
28
496
```

Q.17.Find whether a given string starts with a given character using Lambda

In [27]:
```
starts_with_char = lambda string, char: string.startswith(char)
input_string = "apple"
start_char = "a"
result = starts_with_char(input_string, start_char)

if result:
    print(f"The string '{input_string}' starts with the character '{start_char}'.")
else:
    print(f"The string '{input_string}' does not start with the character '{start_c
```

```
The string 'apple' starts with the character 'a'.
```

Q.18.Write a Python program to handle a ZeroDivisionErrorexception when dividing

a number by zero.

In [28]:
```
try:
    numerator = 10
    denominator = 0

    result = numerator / denominator

except ZeroDivisionError as e:
    print(f"ZeroDivisionError: {e}")
else:
    print(f"Result: {result}")
```

```
ZeroDivisionError: division by zero
```

Q.19.Write a Python program that executes division and handles an ArithmeticError exception if there is an arithmetic error.

In [29]:
```
try:
    numerator = 10
    denominator = 0

    result = numerator / denominator

except ArithmeticError as e:
    print(f"ArithmeticError: {e}")
else:
    print(f"Result: {result}")
```

```
ArithmeticError: division by zero
```

Q.20.Write a Python program that executes an operation on a list and handles an IndexError exception if the index is out of range.

In [30]:
```
try:
    my_list = [1, 2, 3, 4, 5]
    index = 10

    value = my_list[index]
except IndexError as e:
    print(f"IndexError: {e}")
else:
    print(f"Value at index {index}: {value}")
```

```
IndexError: list index out of range
```

Q.21.Create Address Book in Python – Using Tkinter

In [1]:
```
import tkinter as tk
from tkinter import messagebox
def add_contact():
    name = name_entry.get()
    email = email_entry.get()
    if name and email:
        contact_list.insert(tk.END, f"{name}: {email}")
        name_entry.delete(0, tk.END)
        email_entry.delete(0, tk.END)
    else:
        messagebox.showerror("Error", "Please enter both name and email.")
def delete_contact():
    selected_contact = contact_list.get(tk.ACTIVE)
    if selected_contact:
        contact_list.delete(tk.ACTIVE)
    else:
        messagebox.showerror("Error", "No contact selected for deletion.")
root = tk.Tk()
root.title("Address Book")
name_label = tk.Label(root, text="Name:")
name_label.pack()
name_entry = tk.Entry(root)
name_entry.pack()
email_label = tk.Label(root, text="Email:")
email_label.pack()
email_entry = tk.Entry(root)
email_entry.pack()
add_button = tk.Button(root, text="Add Contact", command=add_contact)
add_button.pack()

delete_button = tk.Button(root, text="Delete Contact", command=delete_contact)
delete_button.pack()
contact_list = tk.Listbox(root)
contact_list.pack()
root.mainloop()
```

Q.22.Write a python code to build simple GUI calculator

In [1]:
```
import tkinter as tk
def button_click(number):
    current = display.get()
    display.delete(0, tk.END)
    display.insert(0, current + str(number))
def clear_display():
    display.delete(0, tk.END)
def calculate():
    expression = display.get()
    try:
        result = eval(expression)
        display.delete(0, tk.END)
        display.insert(0, result)
    except Exception as e:
        display.delete(0, tk.END)
        display.insert(0, "Error")
root = tk.Tk()
root.title("Simple Calculator")
display = tk.Entry(root, width=20, borderwidth=5)
display.grid(row=0, column=0, columnspan=4)
buttons = [
```

```
    '7', '8', '9', '/',
    '4', '5', '6', '*',
    '1', '2', '3', '-',
    '0', 'C', '=', '+'
]

row_val = 1
col_val = 0

for button in buttons:
    if button == '=':
        tk.Button(root, text=button, padx=20, pady=20, command=calculate).grid(row=
    elif button == 'C':
        tk.Button(root, text=button, padx=20, pady=20, command=clear_display).grid(
    else:
        tk.Button(root, text=button, padx=20, pady=20, command=lambda b=button: but

    col_val += 1
    if col_val > 3:
        col_val = 0
        row_val += 1
root.mainloop()
```

Q.23.Write a python code to build web page with student registration form

In [5]:

```
from flask import Flask, render_template, request

app = Flask(__name__)

@app.route("/student/register", methods=["GET", "POST"])
def student_register():

    if request.method == "POST":

        student_name = request.form["student_name"]
        student_email = request.form["student_email"]
        student_phone_number = request.form["student_phone_number"]

        return render_template("student_registration_confirmation.html", student_na

    else:
        return render_template("student_registration_form.html")

@app.route("/")
def index():

    return render_template("index.html")


if __name__ == "__main__":
    app.run(debug=True)
```

```
 * Serving Flask app '__main__'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. U
se a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with watchdog (windowsapi)
```

```
An exception has occurred, use %tb to see the full traceback.

SystemExit: 1
C:\Users\Somnath Gaikwad\anaconda3\Lib\site-packages\IPython\core\interactiveshel
l.py:3513: UserWarning: To exit: use 'exit', 'quit', or Ctrl-D.
  warn("To exit: use 'exit', 'quit', or Ctrl-D.", stacklevel=1)
```

Q.24.Write a python code to build web pages with sign-in and sing-up forms

In [ ]:

```
<!DOCTYPE html>
<html>
<head>
    <title>Sign-In and Sign-Up Forms</title>
    <style>
        body {
            font-family: Arial, sans-serif;
        }

        .container {
            display: flex;
            justify-content: space-around;
            width: 80%;
            margin: 0 auto;
        }

        .form {
            width: 45%;
            border: 1px solid
            padding: 20px;
            border-radius: 5px;
        }

        label {
            display: block;
            font-weight: bold;
        }

        input[type="text"], input[type="password"], input[type="email"] {
            width: 100%;
            padding: 10px;
            border: 1px solid
            border-radius: 5px;
            margin-bottom: 20px;
        }

        input[type="submit"] {
            background-color:
            color: white;
            padding: 10px 20px;
            border: none;
            border-radius: 5px;
            cursor: pointer;
        }

        input[type="submit"]:hover {
            background-color:
        }
    </style>
</head>
```

```
<body>
    <h1>Sign-In and Sign-Up Forms</h1>
    <div class="container">
        <div class="form">
            <h2>Sign In</h2>
            <form action="sign_in.php" method="post">
                <label for="username">Username:</label>
                <input type="text" id="username" name="username" required>
                <label for="password">Password:</label>
                <input type="password" id="password" name="password" required>
                <input type="submit" value="Sign In">
            </form>
        </div>
        <div class="form">
            <h2>Sign Up</h2>
            <form action="sign_up.php" method="post">
                <label for="new_username">Username:</label>
                <input type="text" id="new_username" name="new_username" required>
                <label for="new_password">Password:</label>
                <input type="password" id="new_password" name="new_password" requir
                <label for="email">Email:</label>
                <input type="email" id="email" name="email" required>
                <input type="submit" value="Sign Up">
            </form>
        </div>
    </div>
</body>
</html>
```

Q.25.Write a python code to build Rest api for product

In [ ]:

```
from flask import Flask, request, jsonify

app = Flask(__name)

products = [
    {"id": 1, "name": "Product 1", "price": 10.0},
    {"id": 2, "name": "Product 2", "price": 20.0},
]

@app.route('/products', methods=['GET'])
def get_products():
    return jsonify(products)

@app.route('/products/<int:product_id>', methods=['GET'])
def get_product(product_id):
    product = next((p for p in products if p['id'] == product_id), None)
    if product:
        return jsonify(product)
    return jsonify({"message": "Product not found"}), 404

@app.route('/products', methods=['POST'])
def create_product():
    data = request.get_json()
    new_product = {
        "id": len(products) + 1,
        "name": data['name'],
        "price": data['price']
    }
    products.append(new_product)
```

```
    return jsonify(new_product), 201

@app.route('/products/<int:product_id>', methods=['PUT'])
def update_product(product_id):
    product = next((p for p in products if p['id'] == product_id), None)
    if product:
        data = request.get_json()
        product['name'] = data['name']
        product['price'] = data['price']
        return jsonify(product)
    return jsonify({"message": "Product not found"}), 404

@app.route('/products/<int:product_id>', methods=['DELETE'])
def delete_product(product_id):
    product = next((p for p in products if p['id'] == product_id), None)
    if product:
        products.remove(product)
        return jsonify({"message": "Product deleted"})
    return jsonify({"message": "Product not found"}), 404

if __name__ == '__main__':
    app.run(debug=True)
```

Q.26.Write a python code to build Ajax enabled web application for product

In [ ]:

```
from flask import Flask, render_template, request
import json

app = Flask(__name__)
products = [
    {
        "id": 1,
        "name": "Product 1",
        "price": 100,
    },
    {
        "id": 2,
        "name": "Product 2",
        "price": 200,
    },
    {
        "id": 3,
        "name": "Product 3",
        "price": 300,
    },
]

@app.route("/ajax/products", methods=["GET"])
def ajax_products():

    search_query = request.args.get("search_query", "")
    filtered_products = []
    for product in products:
        if search_query in product["name"]:
            filtered_products.append(product)
    return json.dumps({"products": filtered_products})
@app.route("/")
def index():
    return render_template("index.html")
```

```
if __name__ == "__main__":
    app.run(debug=True)
```

In [ ]: