Name - Bhavya Mahawar
batch - 80
sapid-590029516

1. Check whether the given number is divisible by 3 and 5 both.

```python
def check_divisibility(number):
    if (number % 3 == 0) and (number % 5 == 0):
        return True
    else:
        return False
num1 = 15
print(f"Is {num1} divisible by both 3 and 5? {check_divisibility(num1)}")

num2 = 10
print(f"Is {num2} divisible by both 3 and 5? {check_divisibility(num2)}")

num3 = 9
print(f"Is {num3} divisible by both 3 and 5? {check_divisibility(num3)}")

num4 = 30
print(f"Is {num4} divisible by both 3 and 5? {check_divisibility(num4)}")
```

2. Check whether a given number is multiple of five or not.

```python
def check_multiple_of_five(number):

    if number % 5 == 0:
        return True
    else:
        return False

num1 = 25
print(f"Is {num1} a multiple of five? {check_multiple_of_five(num1)}")

num2 = 12
print(f"Is {num2} a multiple of five? {check_multiple_of_five(num2)}")

num3 = 0
print(f"Is {num3} a multiple of five? {check_multiple_of_five(num3)}")

num4 = 100
print(f"Is {num4} a multiple of five? {check_multiple_of_five(num4)}")
```

3. Find the greatest among the two numbers. If numbers are equal than print “numbers are equal”.

```python
def find_greatest(num1, num2):

    if num1 > num2:
        print(f"The greatest number is: {num1}")
    elif num2 > num1:
        print(f"The greatest number is: {num2}")
    else:
        print("Numbers are equal")

find_greatest(10, 5)
find_greatest(7, 15)
find_greatest(20, 20)
find_greatest(-3, -8)
```

4. Find the greatest among three numbers assuming no two values are same.

```python
def find_greatest_three(num1, num2, num3):
    if num1 > num2 and num1 > num3:
        print(f"The greatest number is: {num1}")
    elif num2 > num1 and num2 > num3:
        print(f"The greatest number is: {num2}")
    else:
        print(f"The greatest number is: {num3}")

find_greatest_three(10, 5, 20)
find_greatest_three(30, 15, 25)
find_greatest_three(7, 12, 9)
find_greatest_three(-1, -5, -10)
```

5. Check whether the quadratic equation has real roots or imaginary roots. Display the roots.

```
import cmath

def solve_quadratic_equation(a, b, c):
 t
    discriminant = (b**2) - 4*(a*c)

    if discriminant >= 0:

        print("The roots are real.")
        root1 = (-b - cmath.sqrt(discriminant)) / (2 * a)
        root2 = (-b + cmath.sqrt(discriminant)) / (2 * a)
    else:

        print("The roots are imaginary (complex).")
        root1 = (-b - cmath.sqrt(discriminant)) / (2 * a)
        root2 = (-b + cmath.sqrt(discriminant)) / (2 * a)

    print(f'The roots are {root1} and {root2}')


print("Equation: x^2 + 5x + 6 = 0 (Real roots)")
solve_quadratic_equation(1, 5, 6)

print("\nEquation: x^2 + 2x + 1 = 0 (Real and equal roots)")
solve_quadratic_equation(1, 2, 1)

print("\nEquation: x^2 + x + 1 = 0 (Imaginary roots)")
solve_quadratic_equation(1, 1, 1)

print("\nEquation: 2x^2 + 4x + 2 = 0 (Real and equal roots)")
solve_quadratic_equation(2, 4, 2)
```

6. Find whether a given year is a leap year or not.

```
def is_leap_year(year):

    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
        return True
    else:
        return False
        #Examples:
print(f"Is 2000 a leap year? {is_leap_year(2000)}")
print(f"Is 2004 a leap year? {is_leap_year(2004)}")
print(f"Is 1900 a leap year? {is_leap_year(1900)}")
print(f"Is 2023 a leap year? {is_leap_year(2023)}")
print(f"Is 2024 a leap year? {is_leap_year(2024)}")
```

7. Write a program which takes any date as input and display next date of the calendar e.g. I/P: day=20 month=9 year=2005 O/P: day=21 month=9 year 2005

```
from datetime import date, timedelta

def get_next_date(day, month, year):

    try:
        current_date = date(year, month, day)
        next_date = current_date + timedelta(days=1)
        return next_date.day, next_date.month, next_date.year
    except ValueError as e:
        return f"Invalid date input: {e}"

input_day = 20
input_month = 9
input_year = 2005

next_day, next_month, next_year = get_next_date(input_day, input_month, input_year)
```

```
    print(f"I/P: day={input_day} month={input_month} year={input_year}")
    print(f"O/P: day={next_day} month={next_month} year={next_year}")

    print("\n--- Testing edge cases ---")

    next_day, next_month, next_year = get_next_date(31, 1, 2023) # Jan 31 -> Feb 1
    print(f"Jan 31, 2023 -> day={next_day} month={next_month} year={next_year}")

    next_day, next_month, next_year = get_next_date(31, 12, 2023) # Dec 31 -> Jan 1 of next year
    print(f"Dec 31, 2023 -> day={next_day} month={next_month} year={next_year}")

    next_day, next_month, next_year = get_next_date(28, 2, 2024) # Feb 28, 2024 (leap year) -> Feb 29, 2024
    print(f"Feb 28, 2024 -> day={next_day} month={next_month} year={next_year}")

    next_day, next_month, next_year = get_next_date(28, 2, 2023) # Feb 28, 2023 (non-leap year) -> Mar 1, 2023
    print(f"Feb 28, 2023 -> day={next_day} month={next_month} year={next_year}")
```

8. Print the grade sheet of a student for the given range of cgpa. Scan marks of five subjects and calculate the percentage. CGPA=percentage/10 CGPA range: 0 to 3.4 -> F 3.5 to 5.0->C+ 5.1 to 6->B 6.1 to 7-> B+ 7.1 to 8-> A 8.1 to 9->A+ 9.1 to 10-> O (Outstanding) Sample Gradesheet Name: Rohit Sharma Roll Number: R17234512 SAPID: 50005673 Sem: 1 Course: B.Tech. CSE AI&ML

Subject name: Marks PDS: 70 Python: 80 Chemistry: 90 English: 60 Physics: 50 Percentage: 70% CGPA:7.0 Grade:

```
def generate_grade_sheet(student_name, roll_number, sapid, semester, course, marks):

    total_marks = sum(marks.values())
    num_subjects = len(marks)
    max_marks_per_subject = 100
    total_possible_marks = num_subjects * max_marks_per_subject

    percentage = (total_marks / total_possible_marks) * 100
    cgpa = percentage / 10

    grade = ""
    if 0 <= cgpa <= 3.4:
        grade = "F"
    elif 3.5 <= cgpa <= 5.0:
        grade = "C+"
    elif 5.1 <= cgpa <= 6.0:
        grade = "B"
    elif 6.1 <= cgpa <= 7.0:
        grade = "B+"
    elif 7.1 <= cgpa <= 8.0:
        grade = "A"
    elif 8.1 <= cgpa <= 9.0:
        grade = "A+"
    elif 9.1 <= cgpa <= 10.0:
        grade = "O (Outstanding)"
    else:
        grade = "Invalid CGPA"

    print(f"Sample Gradesheet")
    print(f"Name: {student_name}")
    print(f"Roll Number: {roll_number}\t\t\tSAPID: {sapid}")
    print(f"Sem: {semester}\t\t\t\tCourse: {course}")
    print("\nSubject name:\tMarks")
    for subject, mark in marks.items():
        print(f"{subject}: \t\t{mark}")

    print(f"Percentage: {percentage:.2f}%")
    print(f"CGPA: {cgpa:.1f}")
    print(f"Grade: {grade}")


student_info = {
    "name": "Rohit Sharma",
    "roll_number": "R17234512",
    "sapid": "50005673",
    "semester": 1,
    "course": "B.Tech. CSE AI&ML"
}

subject_marks = {
    "PDS": 70,
    "Python": 80,
        "Chemistry": 90,
        "English": 60,
        "Physics": 50
    }

    generate_grade_sheet(
        student_info["name"],
        student_info["roll_number"],
        student_info["sapid"],
        student_info["semester"],
        student_info["course"],
        subject_marks
    )

    print("\n--- Another Example ---")
    subject_marks_2 = {
        "Math": 45,
        "Science": 40,
        "History": 30,
        "Art": 55,
        "Music": 38
    }

    generate_grade_sheet(
        "Priya Singh",
        "R17234513",
        "50005674",
        1,
        "B.Tech. ECE",
        subject_marks_2
    )

    print("\n--- Another Example (Outstanding) ---")
    subject_marks_3 = {
        "Data Structures": 95,
        "Algorithms": 98,
        "Networking": 92,
        "OS": 96,
        "Databases": 97
    }

    generate_grade_sheet(
        "Rahul Kumar",
        "R17234514",
        "50005675",
        2,
        "B.Tech. CSE",
        subject_marks_3
    )
```

Double-click (or enter) to edit

Github repo link-https://github.com/mahawarbhavya/pythonexperiments.git