

Name: ALAN GEORGE MATHEWS

KTU-ID: MDL20CS016

Assignment 1: Programming in Python CST 362

- Date of submission: 27-Feb-2023 before 10am
- Learning outcome: Python basics, operators, modules/packages, control statements

1. Python is developed by.....How Python got its name

Ans: Python was developed by Guido van Rossum. Monty Python's Flying Circus was a very famous comedy BBC TV series from the 70s. It was rated so well that it was like a must-watch series at the time and also was known to be very unpredictable, creative, and random; basically, it talked about everything. The Python programming language was very famous since it is easy to learn and use and was considered to be the must-know programming language. Guido van Rossum wanted to give his programming language a name that was unique, mysterious, and short, and thus, Python was named after Monty Python's Flying Circus.

2. Write the output of the statement $2^{**}3^{**}2$

Ans: 512

3. What is the difference between / and // operators. Write one example

Ans: The / operator performs regular division, which means it returns a floating-point number that represents the quotient of the division operation. The // operator performs integer division, which means it returns the quotient of the division operation as an integer by discarding any fractional part. Eg: Input: 10/4 -> Output: 2.5 ----- Input: 10//4 -> Output: 2

4. How the expression $2+3*2^{**}2/3$ is evaluated show the order of evaluation and final output

Ans: The expression will be evaluated as follows: $2 + ((3 * (2 ** 2)) / 3)$ $2 + ((3 * 4) / 3)$ $2 + (12 / 3)$ $2 + 4$ Final output will be 6.

5. How single line and multiline comments are added in Python script

Ans: In Python, we use the hash symbol (#) to add single-line comments and triple quotes (""" """ or ''' ''') to add multi-line comments. Single-line comments are used to add a brief description or explanation of code on the same line. Multi-line comments are often used to provide a description of a module, function, or class in Python.

6. Write the logical operators

Ans: In Python, the logical operators are AND, OR, and NOT. These operators are used to perform logical operations on Boolean values. and: Returns True if both operands are True, otherwise returns False. or: Returns True if at least one operand is True, otherwise returns False. not: Returns the opposite of the operand's value. If the operand is True, it returns False, and if the operand is False, it returns True.

7. What are bit wise operators

Ans: Bitwise operators perform operations on the binary representation of the operands at the bit level. The bitwise operators are: &: Bitwise AND operator |: Bitwise OR operator ^: Bitwise XOR operator ~: Bitwise NOT operator <<: Left shift operator >>: Right shift operator

8. $x=0xAA$, $y=0o16$, $z=0b10110$, find $x^y|z$

Ans. 182

9. How to get the last bit of a number. Write the bitwise operation

e/g: $x=2$ o/p:1 $x=3$ o/p:1

Ans.

Num =3

last_bit = num & 1

10. $x=12$, what is the output of $x<<2$ justify your answer

Ans: In Python, the $<<$ operator is the left shift operator, which shifts the bits of a number to the left by a certain number of positions. Each shift to the left doubles the value of the number. Thus, result for this problem would mean x becomes 48 x in binary: 00001100(12)
Result: 00110000(48)

11. What are Boolean data types

Ans: In Python, the Boolean data type is a built-in data type that represents the two truth values True and False. These values are used to represent the logical values of True and False in Python programs. Boolean values are often used in conditional statements and loops to control program flow.

- **Basic Scripting**

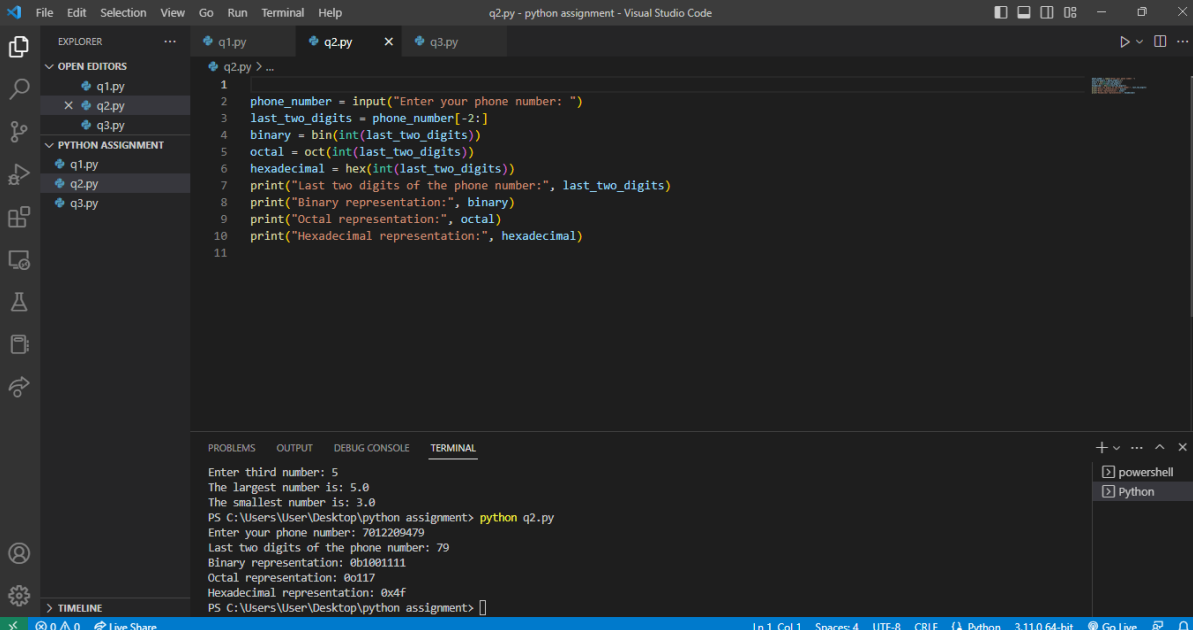
1. Write a Python script which will read two complex numbers and find their sum, difference and product.

```
python.py x  Extension: Python
python.py > ...
1 print("Enter the first complex number:")
2 real1 = float(input("Real part: "))
3 imag1 = float(input("Imaginary part: "))
4 num1 = complex(real1, imag1)
5
6
7 print("\nEnter the second complex number:")
8 real2 = float(input("Real part: "))
9 imag2 = float(input("Imaginary part: "))
10 num2 = complex(real2, imag2)
11
12 sum = num1 + num2
13 diff = num1 - num2
14 prod = num1 * num2
15
16 print("\nSum of the two complex numbers:", sum)
17 print("Difference of the two complex numbers:", diff)
18 print("Product of the two complex numbers:", prod)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Installing collected packages: pycodestyle, autopep8
Successfully installed autopep8-2.0.1 pycodestyle-2.10.0

[notice] A new release of pip available: 22.3.1 -> 23.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip
PS C:\Users\User\Desktop\python assignment> python python.py
Sum of the two complex numbers: (5+5j)
Difference of the two complex numbers: (-1+1j)
Product of the two complex numbers: 13j
PS C:\Users\User\Desktop\python assignment>
```

2. Read your phone number and print the last two digit in binary, octal and hex.



```
File Edit Selection View Go Run Terminal Help
q2.py - python assignment - Visual Studio Code

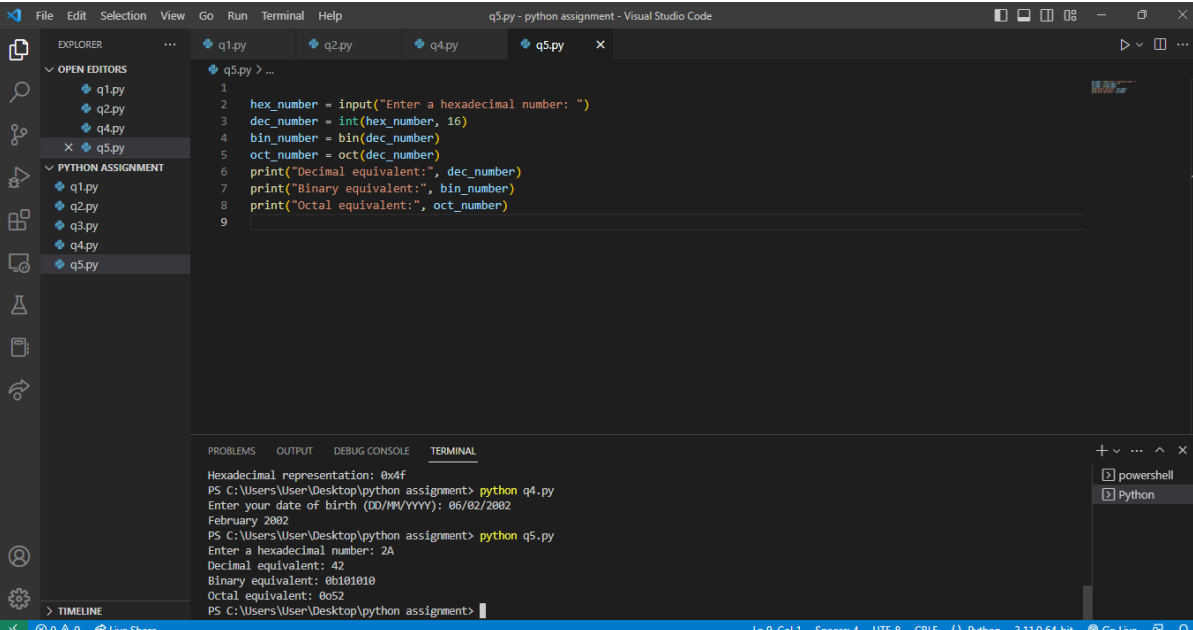
EXPLORER
  OPEN EDITORS
    q1.py
    q2.py
    q3.py
  PYTHON ASSIGNMENT
    q1.py
    q2.py
    q3.py

q2.py > ...
1
2 phone_number = input("Enter your phone number: ")
3 last_two_digits = phone_number[-2:]
4 binary = bin(int(last_two_digits))
5 octal = oct(int(last_two_digits))
6 hexadecimal = hex(int(last_two_digits))
7 print("Last two digits of the phone number:", last_two_digits)
8 print("Binary representation:", binary)
9 print("Octal representation:", octal)
10 print("Hexadecimal representation:", hexadecimal)
11

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Enter third number: 5
The largest number is: 5.0
The smallest number is: 3.0
PS C:\Users\User\Desktop\python assignment> python q2.py
Enter your phone number: 7812289479
Last two digits of the phone number: 79
Binary representation: 0b1001111
Octal representation: 0o117
Hexadecimal representation: 0x4f
PS C:\Users\User\Desktop\python assignment>

Ln 1, Col 1 Spaces: 4 UTF-8 CRUF Python 3.11.0 64-bit Go Live
```

3. Read a hexa decimal number and print the dec, bin and octal equivalent.



```
File Edit Selection View Go Run Terminal Help
q5.py - python assignment - Visual Studio Code

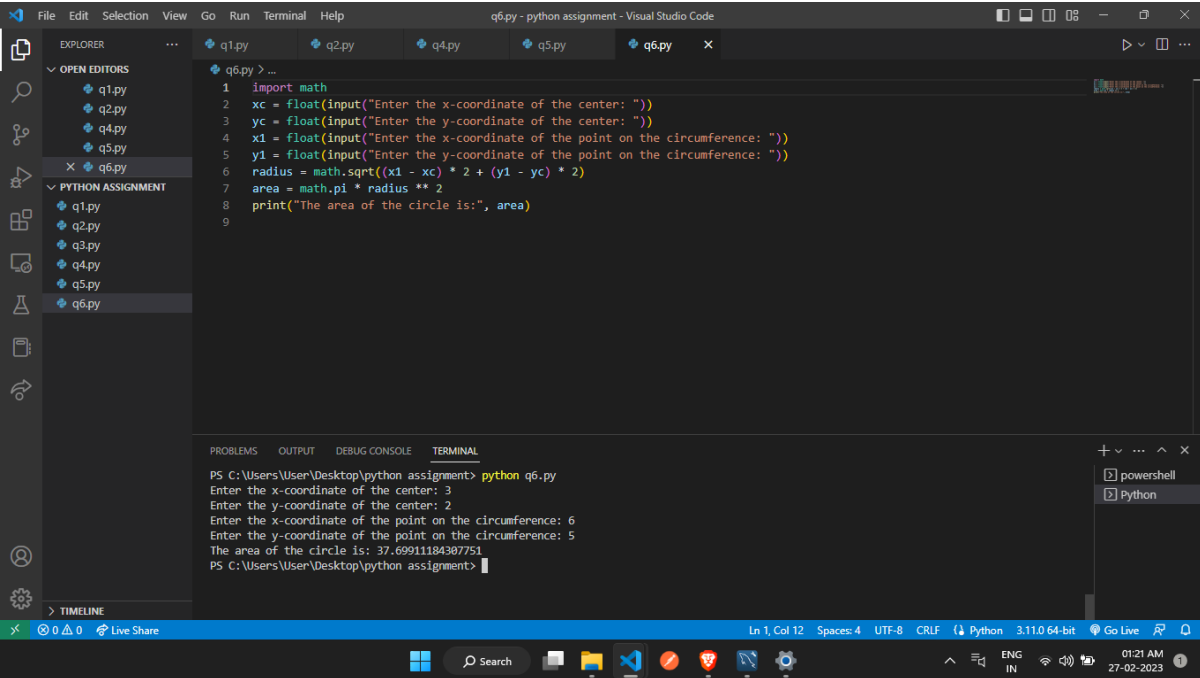
EXPLORER
  OPEN EDITORS
    q1.py
    q2.py
    q4.py
    q5.py
  PYTHON ASSIGNMENT
    q1.py
    q2.py
    q3.py
    q4.py
    q5.py

q5.py > ...
1
2 hex_number = input("Enter a hexadecimal number: ")
3 dec_number = int(hex_number, 16)
4 bin_number = bin(dec_number)
5 oct_number = oct(dec_number)
6 print("Decimal equivalent:", dec_number)
7 print("Binary equivalent:", bin_number)
8 print("Octal equivalent:", oct_number)
9

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Hexadecimal representation: 0x4f
PS C:\Users\User\Desktop\python assignment> python q4.py
Enter your date of birth (DD/MM/YYYY): 06/02/2002
February 2002
PS C:\Users\User\Desktop\python assignment> python q5.py
Enter a hexadecimal number: 2A
Decimal equivalent: 42
Binary equivalent: 0b101010
Octal equivalent: 0o52
PS C:\Users\User\Desktop\python assignment>

Ln 9, Col 1 Spaces: 4 UTF-8 CRUF Python 3.11.0 64-bit Go Live
```

4. Given the center (xc,yc) and a point on the circle(x1,y1). Find the area

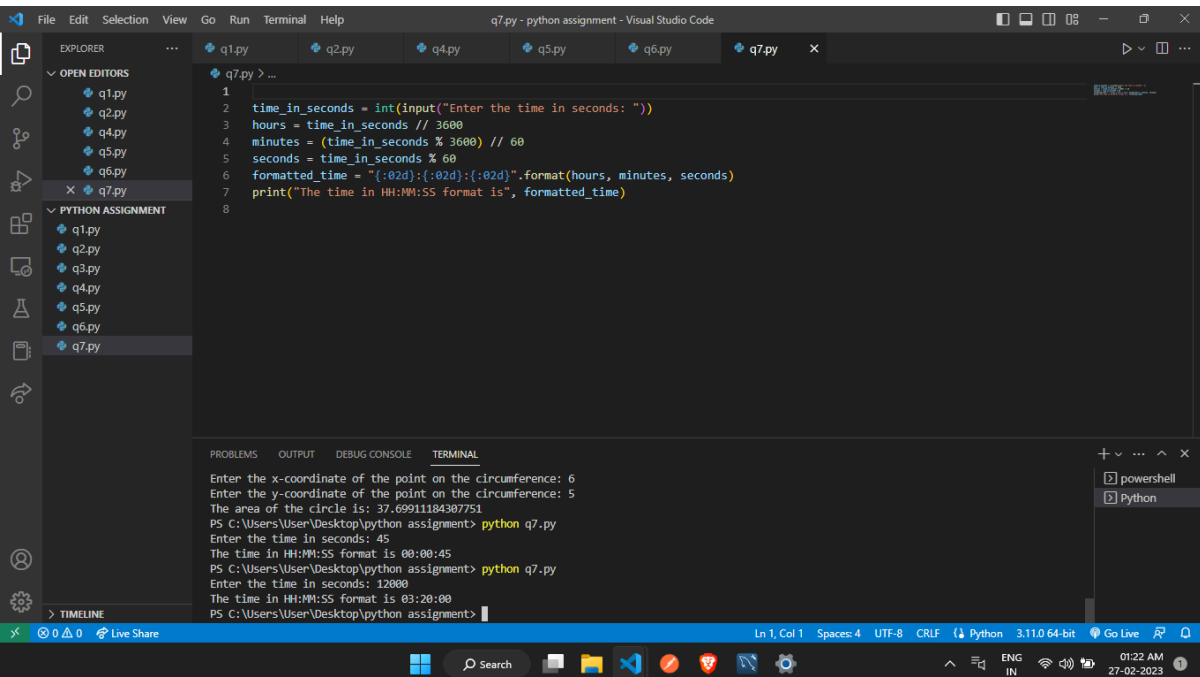


The screenshot shows the Visual Studio Code interface with a Python file named q6.py open. The code calculates the area of a circle by taking the center coordinates (xc, yc) and a point on the circumference (x1, y1) as input. It uses the formula $radius = \sqrt{(x1 - xc)^2 + (y1 - yc)^2}$ and then calculates the area as $area = \pi * radius^2$. The terminal output shows the program being executed with inputs 3, 2, 6, and 5, resulting in an area of 37.69911184387751.

```
1 import math
2 xc = float(input("Enter the x-coordinate of the center: "))
3 yc = float(input("Enter the y-coordinate of the center: "))
4 x1 = float(input("Enter the x-coordinate of the point on the circumference: "))
5 y1 = float(input("Enter the y-coordinate of the point on the circumference: "))
6 radius = math.sqrt((x1 - xc) * 2 + (y1 - yc) * 2)
7 area = math.pi * radius ** 2
8 print("The area of the circle is:", area)
9
```

```
PS C:\Users\User\Desktop\python assignment> python q6.py
Enter the x-coordinate of the center: 3
Enter the y-coordinate of the center: 2
Enter the x-coordinate of the point on the circumference: 6
Enter the y-coordinate of the point on the circumference: 5
The area of the circle is: 37.69911184387751
PS C:\Users\User\Desktop\python assignment>
```

5. Write a Python program to read time in seconds and Print in HH:MM:SS format. i/p in seconds :1000 o/p:00:06:40



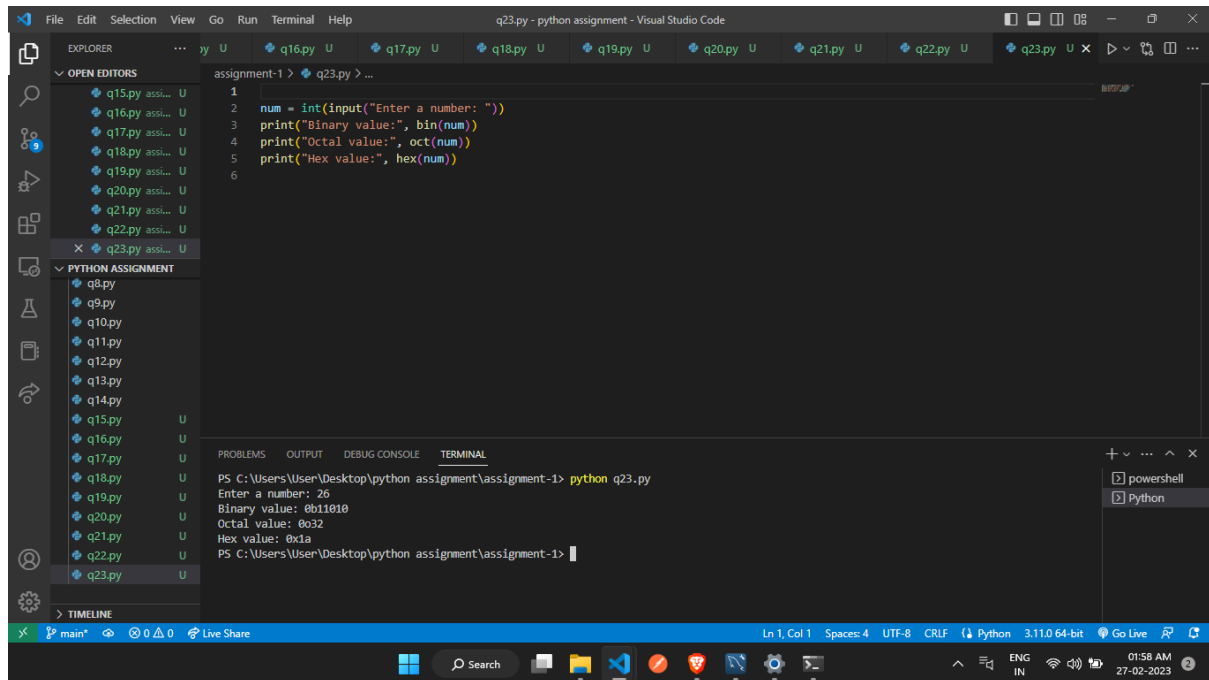
The screenshot shows the Visual Studio Code interface with a Python file named q7.py open. The code reads time in seconds and converts it to HH:MM:SS format using integer division and modulo operations. The terminal output shows the program being executed with inputs 45 and 12000, resulting in formatted times 00:00:45 and 03:20:00.

```
1
2 time_in_seconds = int(input("Enter the time in seconds: "))
3 hours = time_in_seconds // 3600
4 minutes = (time_in_seconds % 3600) // 60
5 seconds = time_in_seconds % 60
6 formatted_time = "{:02d}:{:02d}:{:02d}".format(hours, minutes, seconds)
7 print("The time in HH:MM:SS format is", formatted_time)
8
```

```
Enter the x-coordinate of the point on the circumference: 6
Enter the y-coordinate of the point on the circumference: 5
The area of the circle is: 37.69911184387751
PS C:\Users\User\Desktop\python assignment> python q7.py
Enter the time in seconds: 45
The time in HH:MM:SS format is 00:00:45
PS C:\Users\User\Desktop\python assignment> python q7.py
Enter the time in seconds: 12000
The time in HH:MM:SS format is 03:20:00
PS C:\Users\User\Desktop\python assignment>
```

- Use built in functions

1. Read a number and Print the corresponding binary, oct, hex



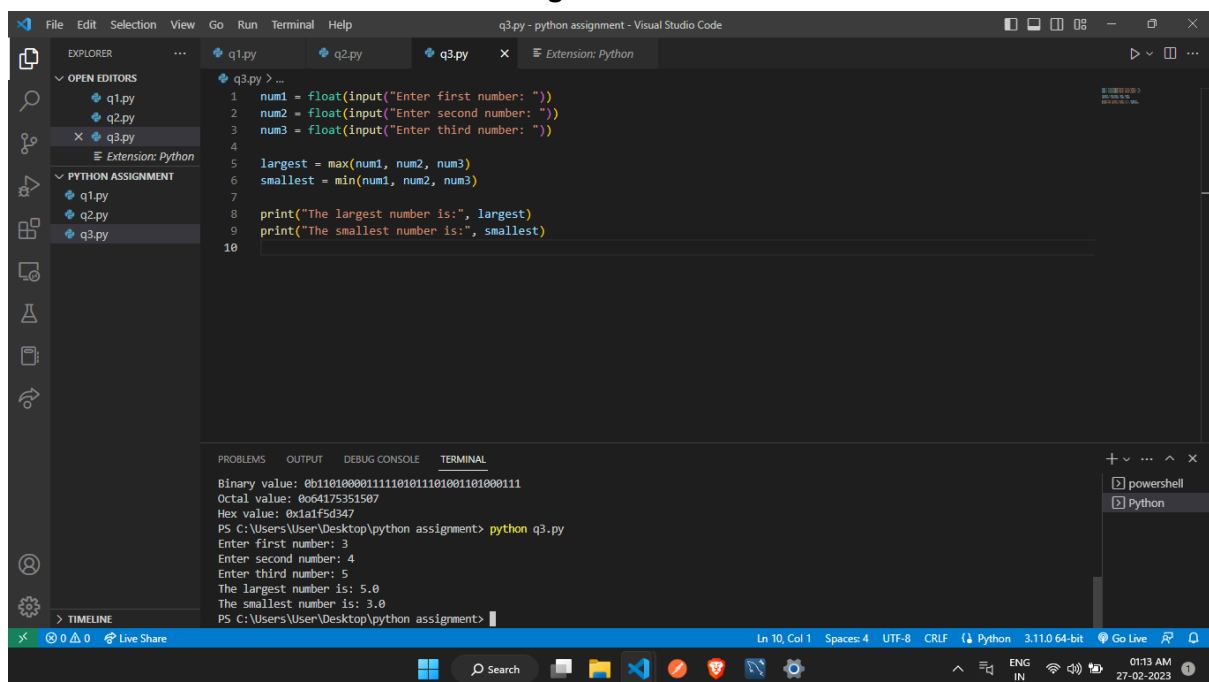
```
File Edit Selection View Go Run Terminal Help
q23.py - python assignment - Visual Studio Code

EXPLORER
OPEN EDITORS
q15.py assi... U
q16.py assi... U
q17.py assi... U
q18.py assi... U
q19.py assi... U
q20.py assi... U
q21.py assi... U
q22.py assi... U
q23.py assi... U
PYTHON ASSIGNMENT
q8.py
q9.py
q10.py
q11.py
q12.py
q13.py
q14.py
q15.py U
q16.py U
q17.py U
q18.py U
q19.py U
q20.py U
q21.py U
q22.py U
q23.py U
TIMELINE
main* 0 0 0 Live Share
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python 3.11.0 64-bit Go Live

1
2 num = int(input("Enter a number: "))
3 print("Binary value:", bin(num))
4 print("Octal value:", oct(num))
5 print("Hex value:", hex(num))
6

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS C:\Users\User\Desktop\python assignment\assignment-1> python q23.py
Enter a number: 26
Binary value: 0b11010
Octal value: 0o32
Hex value: 0x1a
PS C:\Users\User\Desktop\python assignment\assignment-1>
```

2. Read 3 numbers and find the largest and smallest



```
File Edit Selection View Go Run Terminal Help
q3.py - python assignment - Visual Studio Code

EXPLORER
OPEN EDITORS
q1.py
q2.py
q3.py
Extension: Python
PYTHON ASSIGNMENT
q1.py
q2.py
q3.py
TIMELINE
0 0 0 Live Share
Ln 10, Col 1 Spaces: 4 UTF-8 CRLF Python 3.11.0 64-bit Go Live

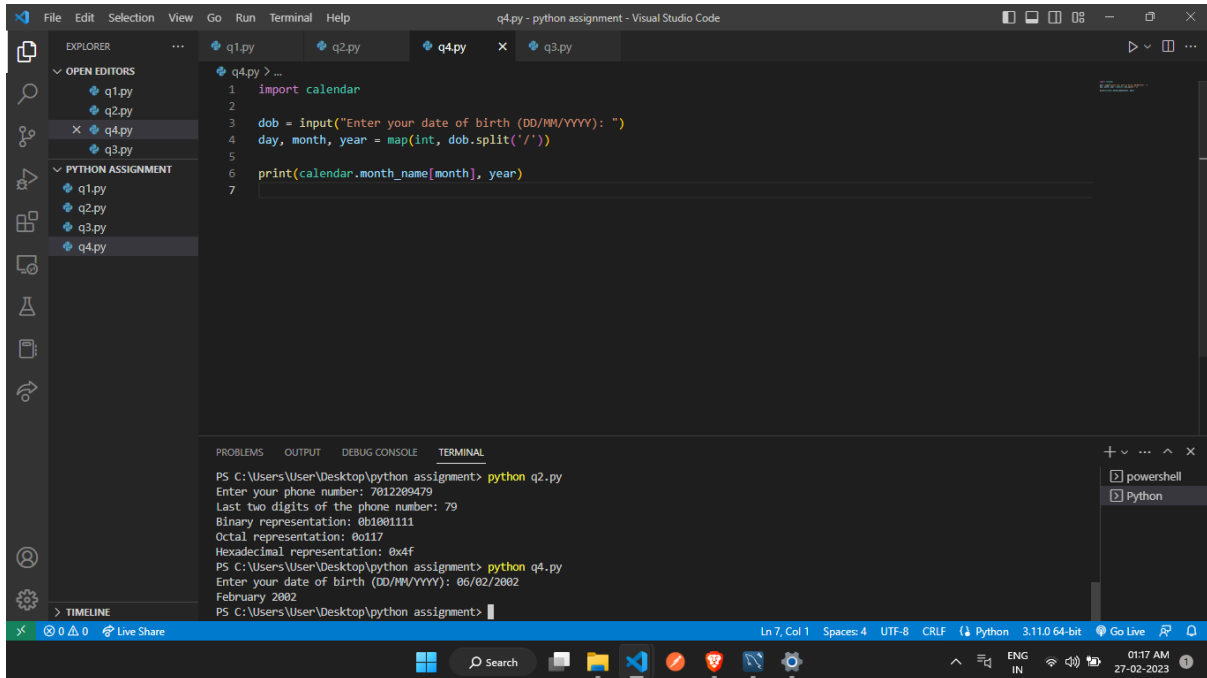
1 num1 = float(input("Enter first number: "))
2 num2 = float(input("Enter second number: "))
3 num3 = float(input("Enter third number: "))
4
5 largest = max(num1, num2, num3)
6 smallest = min(num1, num2, num3)
7
8 print("The largest number is:", largest)
9 print("The smallest number is:", smallest)
10

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Binary value: 0b11010000111110101110100110100111
Octal value: 0o64175351507
Hex value: 0xa1af5d347
PS C:\Users\User\Desktop\python assignment> python q3.py
Enter first number: 3
Enter second number: 4
Enter third number: 5
The largest number is: 5.0
The smallest number is: 3.0
PS C:\Users\User\Desktop\python assignment>
```

- Use modules/packages

1. Print the month calendar depending on your DOB.

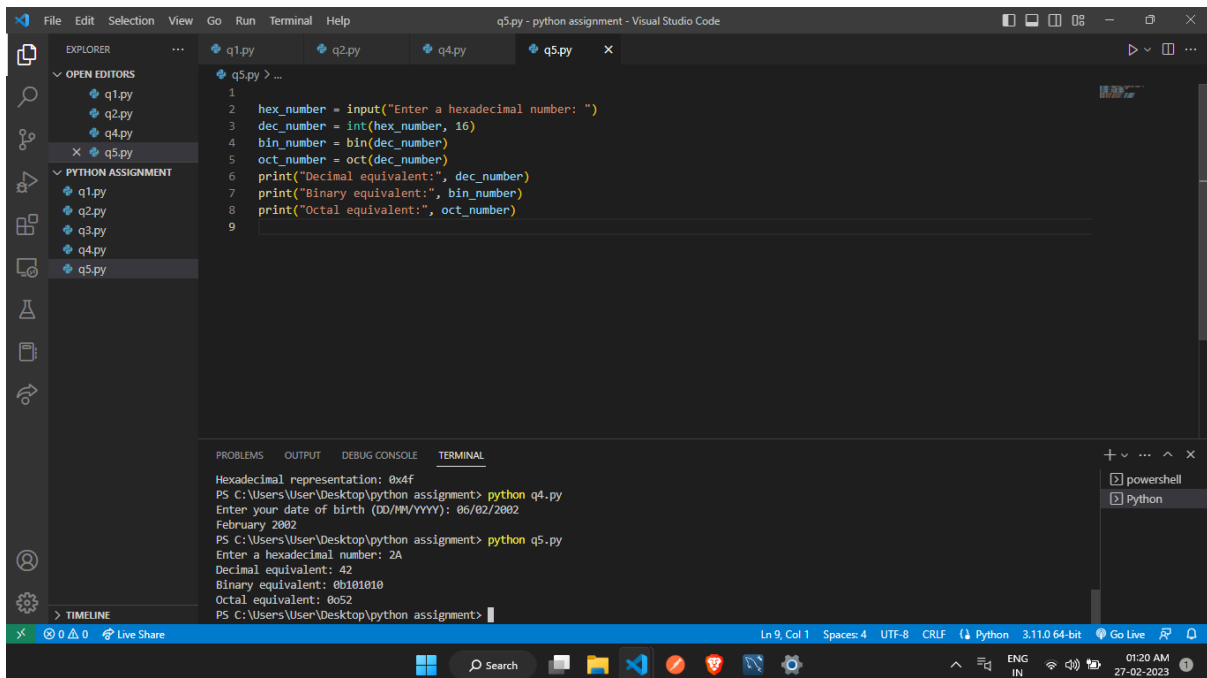
Eg: if the dob is 10/10/2004 the program should print 2004 October month calendar



```
q4.py > ...
1 import calendar
2
3 dob = input("Enter your date of birth (DD/MM/YYYY): ")
4 day, month, year = map(int, dob.split('/'))
5
6 print(calendar.month_name[month], year)
7
```

```
PS C:\Users\User\Desktop\python assignment> python q2.py
Enter your phone number: 7812289479
Last two digits of the phone number: 79
Binary representation: 0b1001111
Octal representation: 0o117
Hexadecimal representation: 0x4f
PS C:\Users\User\Desktop\python assignment> python q4.py
Enter your date of birth (DD/MM/YYYY): 06/02/2002
February 2002
PS C:\Users\User\Desktop\python assignment>
```

2. Find the number of digits in the factorial of a given number



```
q5.py > ...
1
2 hex_number = input("Enter a hexadecimal number: ")
3 dec_number = int(hex_number, 16)
4 bin_number = bin(dec_number)
5 oct_number = oct(dec_number)
6 print("Decimal equivalent:", dec_number)
7 print("Binary equivalent:", bin_number)
8 print("Octal equivalent:", oct_number)
9
```

```
Hexadecimal representation: 0x4f
PS C:\Users\User\Desktop\python assignment> python q4.py
Enter your date of birth (DD/MM/YYYY): 06/02/2002
February 2002
PS C:\Users\User\Desktop\python assignment> python q5.py
Enter a hexadecimal number: 2A
Decimal equivalent: 42
Binary equivalent: 0b101010
Octal equivalent: 0o52
PS C:\Users\User\Desktop\python assignment>
```

3. Find the sqrt of first and last digit of a number

The screenshot shows the Visual Studio Code interface with a Python file named `q22.py` open. The code is as follows:

```
1 import math
2
3 n = int(input("Enter a number: "))
4
5 first_digit = n
6 while first_digit >= 10:
7     first_digit //= 10
8 last_digit = n % 10
9
10
11 sqrt_first = math.sqrt(first_digit)
12 sqrt_last = math.sqrt(last_digit)
13
14 print(
15     f"The square root of the first digit ({first_digit}) is {sqrt_first:.2f}."
16 )
17 print(f"The square root of the last digit ({last_digit}) is {sqrt_last:.2f}.")
```

The terminal output shows the execution of the script with the input 49:

```
PS C:\Users\User\Desktop\python assignment> python q22.py
Enter a number: 49
The square root of the first digit (4) is 2.00.
The square root of the last digit (9) is 3.00.
PS C:\Users\User\Desktop\python assignment>
```

- Using if

1. Check whether the given year is leap year or not.

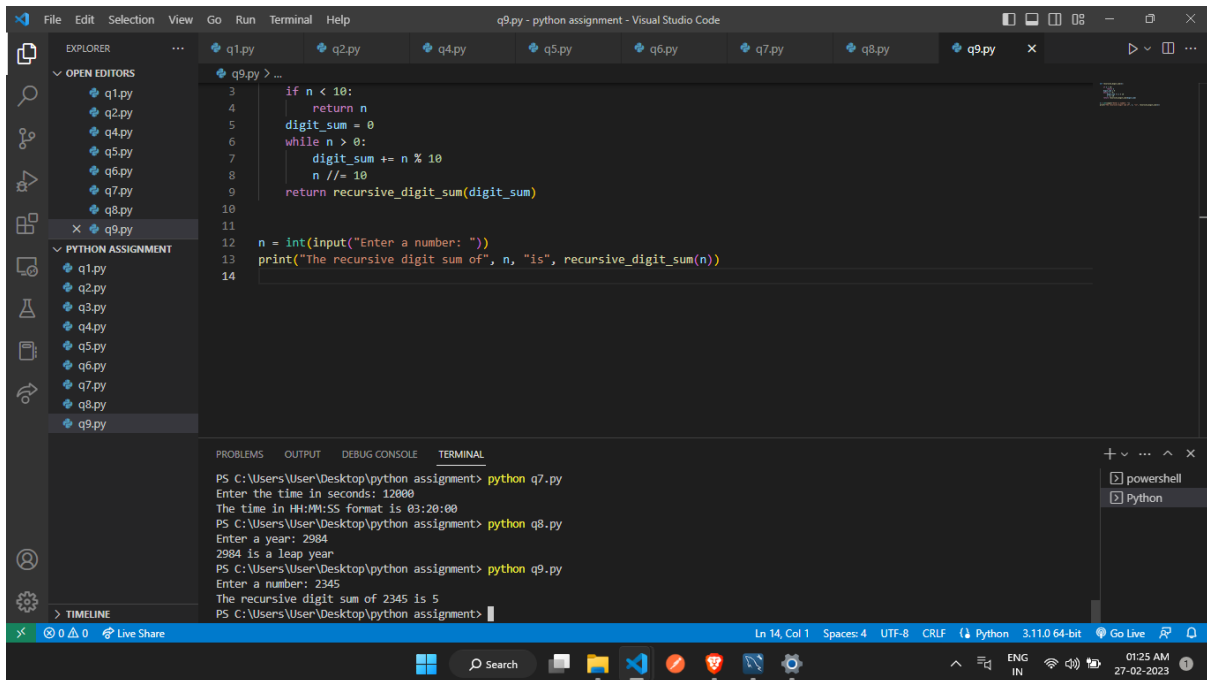
The screenshot shows the Visual Studio Code interface with a Python file named `q8.py` open. The code is as follows:

```
1 year = int(input("Enter a year: "))
2
3 if year % 4 == 0 and (year % 100 != 0 or year % 400 == 0):
4     print(year, "is a leap year")
5 else:
6     print(year, "is not a leap year")
7
```

The terminal output shows the execution of the script with the input 2984:

```
PS C:\Users\User\Desktop\python assignment> python q7.py
Enter the time in seconds: 45
The time in HH:MM:SS format is 00:00:45
PS C:\Users\User\Desktop\python assignment> python q7.py
Enter the time in seconds: 12000
The time in HH:MM:SS format is 03:20:00
PS C:\Users\User\Desktop\python assignment> python q8.py
Enter a year: 2984
2984 is a leap year
PS C:\Users\User\Desktop\python assignment>
```

2. Write a python program to read a number and recursively add the digits in it
Eg: i/p:123 o/p:6 i/p:78
o/p:6



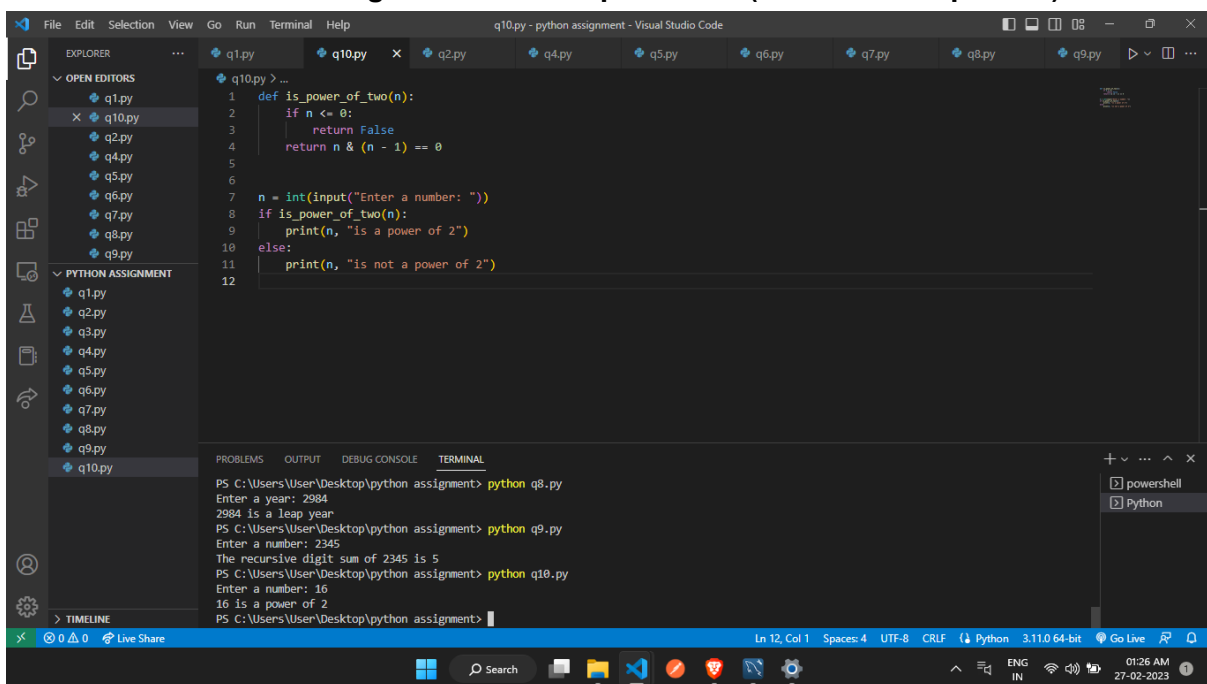
The screenshot shows the Visual Studio Code interface with a Python file named q9.py open. The code defines a recursive function to calculate the sum of digits of a number. The terminal shows the execution of several Python programs, including q9.py, which demonstrates the recursive digit sum calculation for the input 2345, resulting in 5.

```
q9.py > ...
3     if n < 10:
4         return n
5     digit_sum = 0
6     while n > 0:
7         digit_sum += n % 10
8         n //= 10
9     return recursive_digit_sum(digit_sum)
10
11
12 n = int(input("Enter a number: "))
13 print("The recursive digit sum of", n, "is", recursive_digit_sum(n))
14
```

Terminal Output:

```
PS C:\Users\User\Desktop\python assignment> python q7.py
Enter the time in seconds: 12000
The time in HH:MM:SS format is 03:20:00
PS C:\Users\User\Desktop\python assignment> python q8.py
Enter a year: 2984
2984 is a leap year
PS C:\Users\User\Desktop\python assignment> python q9.py
Enter a number: 2345
The recursive digit sum of 2345 is 5
PS C:\Users\User\Desktop\python assignment>
```

3. Check whether the given number is power of 2 (use bitwise operator)



The screenshot shows the Visual Studio Code interface with a Python file named q10.py open. The code defines a function to check if a number is a power of 2 using a bitwise operator. The terminal shows the execution of several Python programs, including q10.py, which demonstrates the power of 2 check for the input 16, resulting in "16 is a power of 2".

```
q10.py > ...
1 def is_power_of_two(n):
2     if n <= 0:
3         return False
4     return n & (n - 1) == 0
5
6
7 n = int(input("Enter a number: "))
8 if is_power_of_two(n):
9     print(n, "is a power of 2")
10 else:
11     print(n, "is not a power of 2")
12
```

Terminal Output:

```
PS C:\Users\User\Desktop\python assignment> python q8.py
Enter a year: 2984
2984 is a leap year
PS C:\Users\User\Desktop\python assignment> python q9.py
Enter a number: 2345
The recursive digit sum of 2345 is 5
PS C:\Users\User\Desktop\python assignment> python q10.py
Enter a number: 16
16 is a power of 2
PS C:\Users\User\Desktop\python assignment>
```


4. Write a program to check the quadrant of a given point(x,y)(University question)

```
q11.py > ...
7 print("The point is on the y-axis.")
8 elif y == 0:
9 print("The point is on the x-axis.")
10 elif x > 0 and y > 0:
11 print("The point is in the first quadrant.")
12 elif x < 0 and y > 0:
13 print("The point is in the second quadrant.")
14 elif x < 0 and y < 0:
15 print("The point is in the third quadrant.")
16 else:
17 print("The point is in the fourth quadrant.")
18
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Enter a number: 2345
The recursive digit sum of 2345 is 5
PS C:\Users\User\Desktop\python assignment> python q10.py
Enter a number: 16
16 is a power of 2
PS C:\Users\User\Desktop\python assignment> python q11.py
Enter the x-coordinate of the point: 3
Enter the y-coordinate of the point: 2
The point is in the first quadrant.
PS C:\Users\User\Desktop\python assignment>

5. Write a program to get the absolute value of a number without using the abs() function.(university question)

```
q12.py U X
assignment-1 > q12.py > absolute_value
1 def absolute_value(n):
2     if n < 0:
3         return -n
4     else:
5         return n
6 n = float(input("Enter a number: "))
7 abs_value = absolute_value(n)
8 print("The absolute value of", n, "is", abs_value)
9
10
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\User\Desktop\python assignment\assignment-1> python q12.py
Enter a number: -12
The absolute value of -12.0 is 12.0
PS C:\Users\User\Desktop\python assignment\assignment-1>

6. Find the roots of a quadratic equation

The screenshot shows the Visual Studio Code interface with a Python file named `q13.py` open. The code calculates the discriminant and prints the roots of a quadratic equation based on its value. The terminal shows the execution of the program with input values 1, 2, and 1, resulting in one real root: -1.0.

```
2
3 a = float(input("Enter the coefficient of x^2 (a): "))
4 b = float(input("Enter the coefficient of x (b): "))
5 c = float(input("Enter the constant term (c): "))
6
7 discriminant = b**2 - 4*a*c
8
9 if discriminant < 0:
10     print("The quadratic equation has no real roots.")
11 elif discriminant == 0:
12     root = -b / (2*a)
13     print("The quadratic equation has one root:", root)
14 else:
15     root1 = (-b + math.sqrt(discriminant)) / (2*a)
16     root2 = (-b - math.sqrt(discriminant)) / (2*a)
17     print("The quadratic equation has two roots:", root1, "and", root2)
18
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\python_assignment> python q13.py
Enter the coefficient of x^2 (a): 1
Enter the coefficient of x (b): 2
Enter the constant term (c): 1
The quadratic equation has one root: -1.0
PS C:\Users\User\Desktop\python_assignment>
```

7. Write a program that accepts the length of three sides of a triangle as input and determine whether or not the triangle is a right triangle.(university question)

The screenshot shows the Visual Studio Code interface with a Python file named `q14.py` open. The code checks if three input sides can form a triangle and then determines if it is a right triangle using the Pythagorean theorem. The terminal shows the execution of the program with input values 3, 5, and 7, resulting in the output: "The triangle is not a right triangle."

```
1 a = float(input("Enter the length of side a: "))
2 b = float(input("Enter the length of side b: "))
3 c = float(input("Enter the length of side c: "))
4
5
6 if a + b <= c or a + c <= b or b + c <= a:
7     print("Invalid triangle.")
8 else:
9
10     sides = [a, b, c]
11     sides.sort()
12
13
14 if sides[0]**2 + sides[1]**2 == sides[2]**2:
15     print("The triangle is a right triangle.")
16 else:
17     print("The triangle is not a right triangle.")
18
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\python_assignment> python q14.py
Enter the length of side a: 3
Enter the length of side b: 5
Enter the length of side c: 7
The triangle is not a right triangle.
PS C:\Users\User\Desktop\python_assignment>
```

- Using while/for

1. Generate the Fibonacci series 0 1 1 2 3 5 8.....n (use while..read n)

The screenshot shows the Visual Studio Code interface with a Python file named q15.py open. The code is as follows:

```
1 n = int(input("Enter the number of terms in the series: "))
2 a, b = 0, 1
3 print(a)
4 print(b)
5 i = 2
6 while i < n:
7     c = a + b
8     a, b = b, c
9     print(c)
10    i += 1
11
```

The terminal output shows the execution of the script:

```
PS C:\Users\User\Desktop\python assignment\assignment-1> python q15.py
Enter the number of terms in the series: 7
0
1
1
2
3
5
8
```

2. Reverse a Number (i/p: 123 o/P 321)

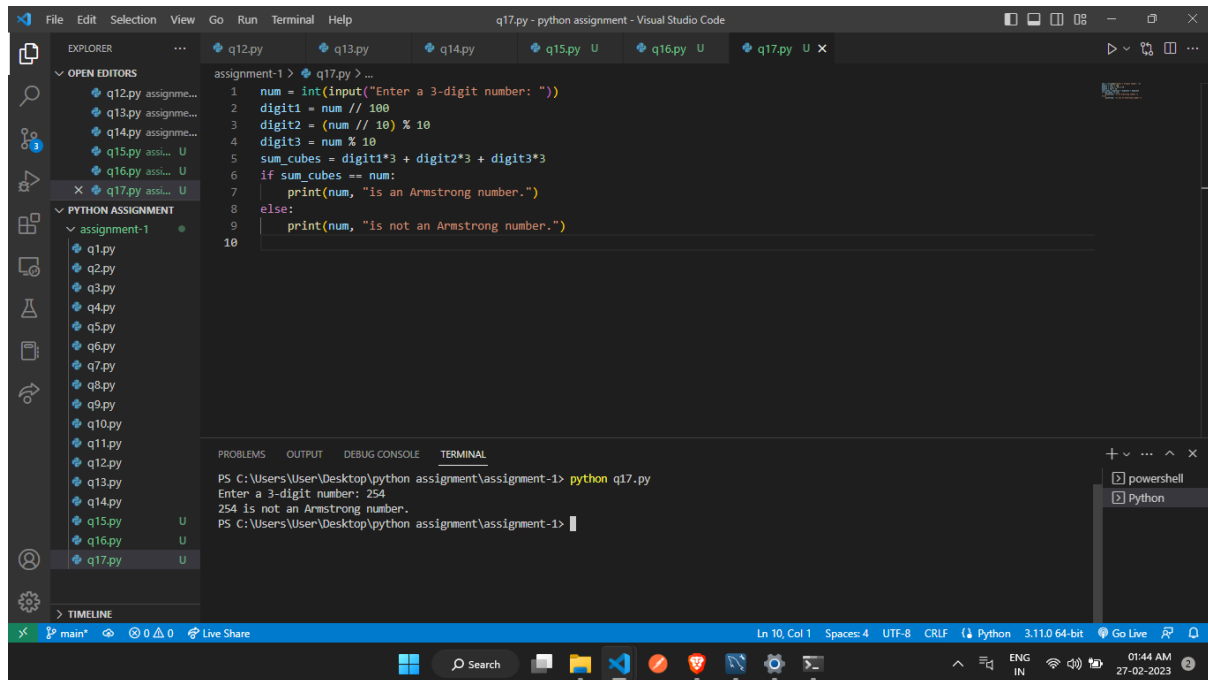
The screenshot shows the Visual Studio Code interface with a Python file named q16.py open. The code is as follows:

```
2 rev_num = 0
3 while num > 0:
4
5     digit = num % 10
6     rev_num = rev_num * 10 + digit
7     num = num // 10
8
9 print("The reversed number is:", rev_num)
10
```

The terminal output shows the execution of the script:

```
PS C:\Users\User\Desktop\python assignment\assignment-1> python q16.py
Enter a number: 456
The reversed number is: 654
PS C:\Users\User\Desktop\python assignment\assignment-1>
```

3. Check whether the given 3 digit number is Armstrong Number.



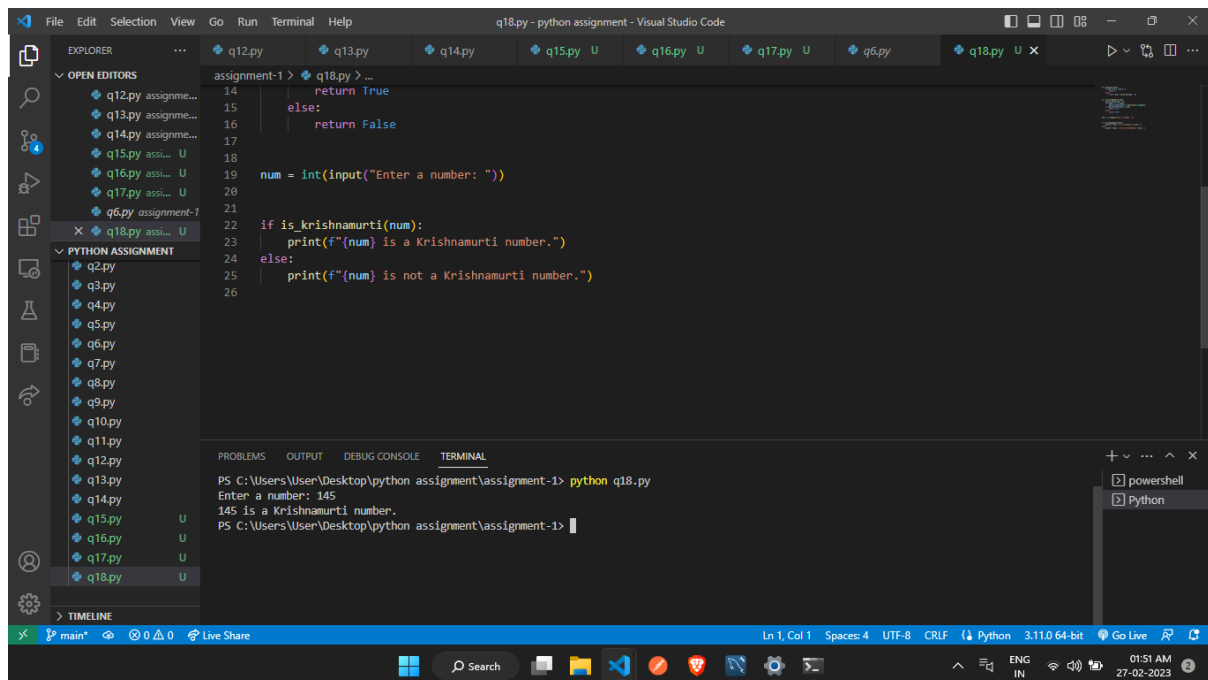
The screenshot shows a Visual Studio Code window with a Python file named `q17.py` open. The code implements a function to check if a 3-digit number is an Armstrong number. The logic is as follows: it takes an input number, extracts its three digits, calculates the sum of their cubes, and compares it to the original number. If they are equal, it's an Armstrong number; otherwise, it's not.

```
1 num = int(input("Enter a 3-digit number: "))
2 digit1 = num // 100
3 digit2 = (num // 10) % 10
4 digit3 = num % 10
5 sum_cubes = digit1*3 + digit2*3 + digit3*3
6 if sum_cubes == num:
7     print(num, "is an Armstrong number.")
8 else:
9     print(num, "is not an Armstrong number.")
10
```

The terminal output shows the program being run with the input 254, which is correctly identified as not an Armstrong number.

```
PS C:\Users\User\Desktop\python assignment\assignment-1> python q17.py
Enter a 3-digit number: 254
254 is not an Armstrong number.
PS C:\Users\User\Desktop\python assignment\assignment-1>
```

4. Check whether the given number is a Krishnamurti number(Krishnamurthy Number: It is a number which is equal to the sum of the factorials of all its digits. Usefactorial() function from math For example : $145 = 1! + 4! + 5! = 1 + 24 + 120 = 145$ which is a Krishnamurti Number.



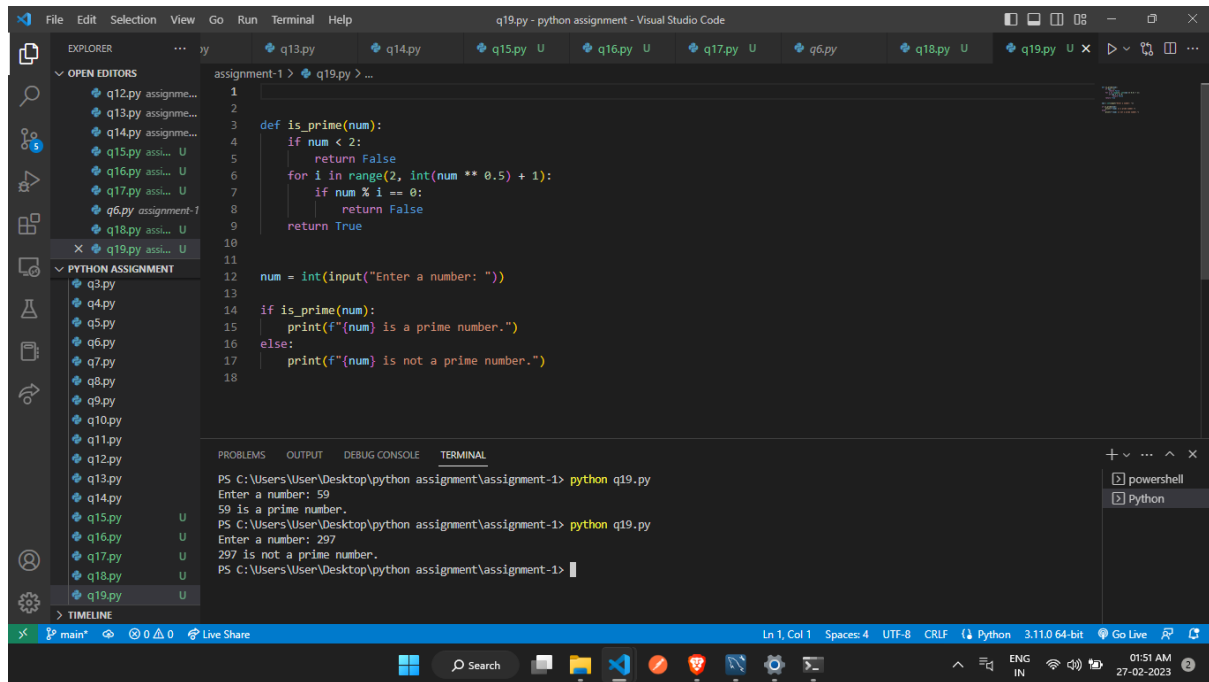
The screenshot shows a Visual Studio Code window with a Python file named `q18.py` open. The code implements a function to check if a number is a Krishnamurti number. It uses a helper function `is_krishnamurti` that iterates through the digits of the number, calculates their factorials, and sums them. The main program prompts the user for a number and prints the result.

```
14 return True
15 else:
16     return False
17
18 num = int(input("Enter a number: "))
19
20
21
22 if is_krishnamurti(num):
23     print(f"{num} is a Krishnamurti number.")
24 else:
25     print(f"{num} is not a Krishnamurti number.")
26
```

The terminal output shows the program being run with the input 145, which is correctly identified as a Krishnamurti number.

```
PS C:\Users\User\Desktop\python assignment\assignment-1> python q18.py
Enter a number: 145
145 is a Krishnamurti number.
PS C:\Users\User\Desktop\python assignment\assignment-1>
```

5. Check whether the given number is Prime or not



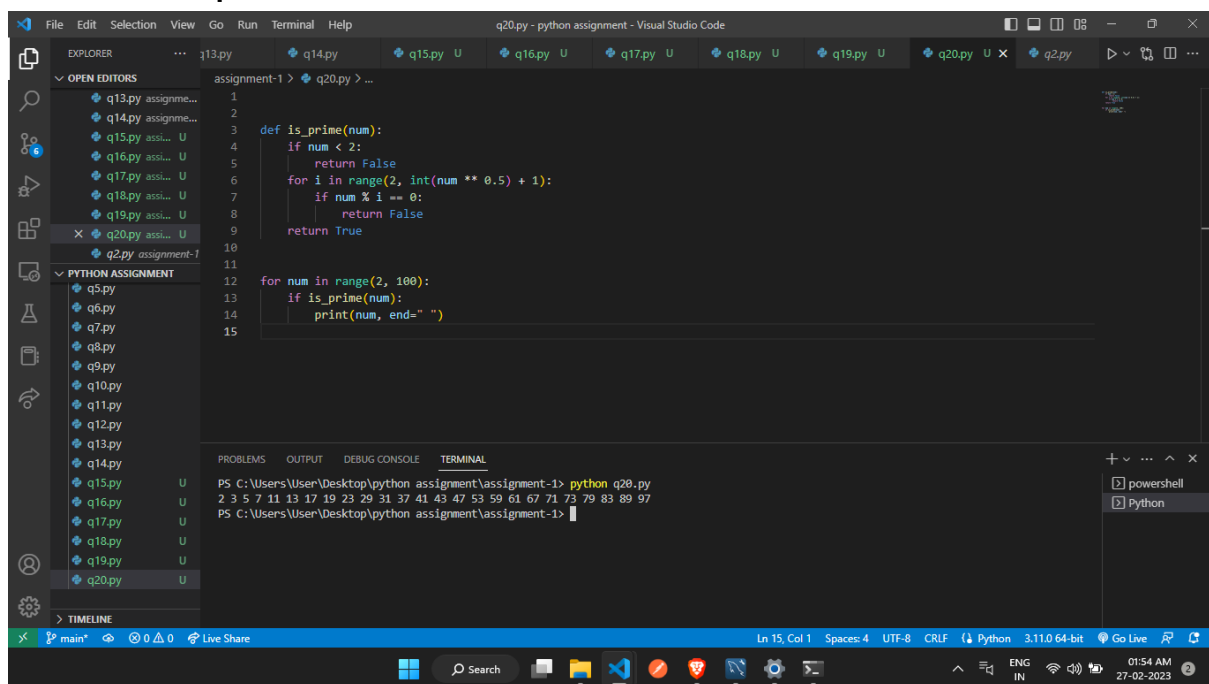
```
1
2
3 def is_prime(num):
4     if num < 2:
5         return False
6     for i in range(2, int(num ** 0.5) + 1):
7         if num % i == 0:
8             return False
9     return True
10
11
12 num = int(input("Enter a number: "))
13
14 if is_prime(num):
15     print(f"{num} is a prime number.")
16 else:
17     print(f"{num} is not a prime number.")
18
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\User\Desktop\python assignment\assignment-1> python q19.py
Enter a number: 59
59 is a prime number.
PS C:\Users\User\Desktop\python assignment\assignment-1> python q19.py
Enter a number: 297
297 is not a prime number.
PS C:\Users\User\Desktop\python assignment\assignment-1>

- Nested Loops(for/while)

1. Print all prime numbers less than 100.

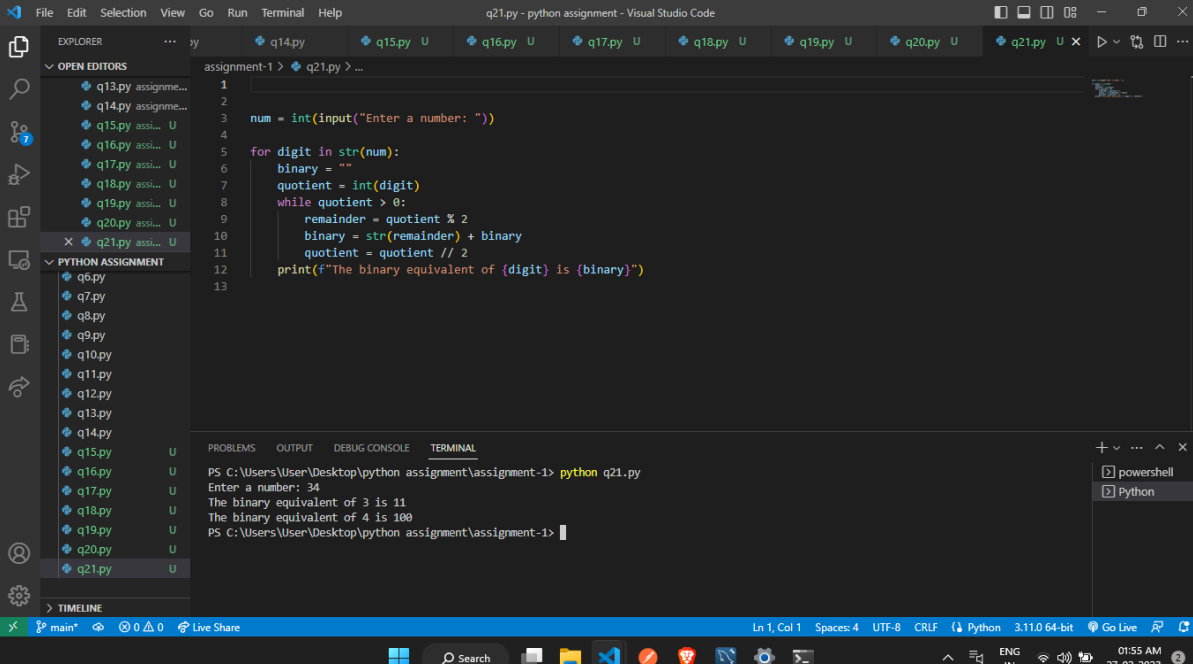


```
1
2
3 def is_prime(num):
4     if num < 2:
5         return False
6     for i in range(2, int(num ** 0.5) + 1):
7         if num % i == 0:
8             return False
9     return True
10
11
12 for num in range(2, 100):
13     if is_prime(num):
14         print(num, end=" ")
15
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\User\Desktop\python assignment\assignment-1> python q20.py
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
PS C:\Users\User\Desktop\python assignment\assignment-1>

2. Print the binary equivalent of each digit of the given number



The screenshot shows the Visual Studio Code interface with a Python file named `q21.py` open. The code in the editor is as follows:

```
1
2
3 num = int(input("Enter a number: "))
4
5 for digit in str(num):
6     binary = ""
7     quotient = int(digit)
8     while quotient > 0:
9         remainder = quotient % 2
10        binary = str(remainder) + binary
11        quotient = quotient // 2
12    print(f"The binary equivalent of {digit} is {binary}")
13
```

The left sidebar shows the Explorer view with a folder named "PYTHON ASSIGNMENT" containing files `q6.py` through `q21.py`. The bottom panel shows the TERMINAL view with the following output:

```
PS C:\Users\User\Desktop\python assignment\assignment-1> python q21.py
Enter a number: 34
The binary equivalent of 3 is 11
The binary equivalent of 4 is 100
PS C:\Users\User\Desktop\python assignment\assignment-1>
```

The status bar at the bottom indicates the file is at line 1, column 1, with 4 spaces, in UTF-8 encoding, using the CRUF line ending, with the Python interpreter set to 3.11.0 64-bit.