

Tutorial=16

Q.1	Perform 7 pattern using class and object (default constructor).
Code	<pre>1. class Pattern: def __init__(self): print("Generated Pattern:") def display(self, size): for i in range(size): for j in range(i + 1): print("*", end=" ") print() pattern = Pattern() pattern.display(5) 2. class Pattern: def __init__(self): print("Generated Pattern:") def display(self, size): for i in range(size, 0, -1): for j in range(i): print("*", end=" ") print()</pre>

```
pattern = Pattern()

pattern.display(5)

3. class NumberPattern:
    def __init__(self):
        print("Generated Number Pattern:")
    def display(self, size):
        for i in range(1, size + 1):
            for j in range(1, i + 1):
                print(j, end=" ")
            print()
pattern = NumberPattern()
pattern.display(5)

4. class NumberPattern:
    def __init__(self):
        print("Generated Number Pattern:")
    def display(self, size):
        for i in range(1, size + 1):
            for j in range(size, 0, -1):
                print(j, end=" ")
            print()

pattern = NumberPattern()
```

```
pattern.display(5)
```

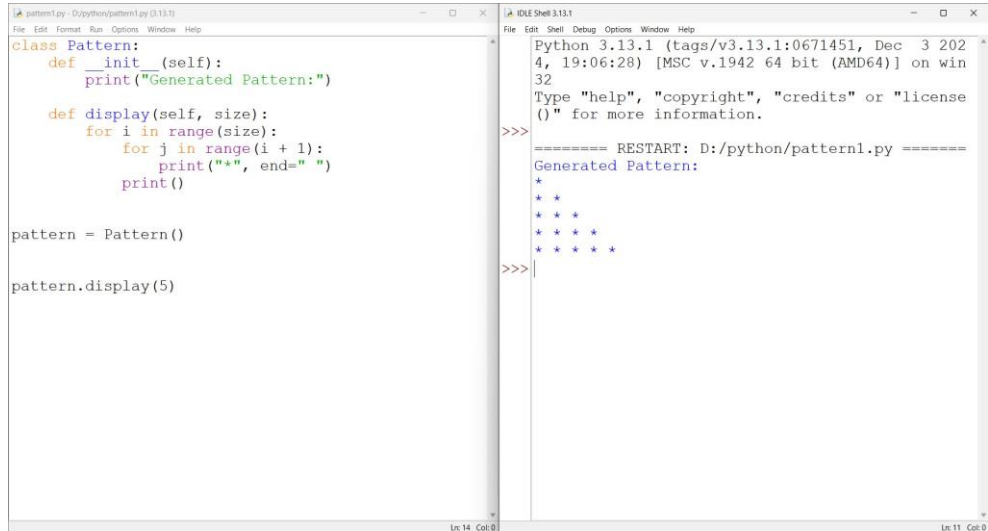
```
5. class NumberPattern:
    def __init__(self):
        print("Generated Number Pattern:")
    def display(self, size):
        for i in range(1, size + 1):
            for j in range(size, size - i, -1):
                print(j, end=" ")
            print()
pattern = NumberPattern()
pattern.display(5)
```

```
6. class NumberPattern:
    def __init__(self):
        print("Generated Reverse Number Pattern:")
    def display(self, size):
        for i in range(size, 0, -1):
            for j in range(size, size - i, -1):
                print(j, end=" ")
            print()
```

```
pattern = NumberPattern()
```

pattern.display(5)

Output



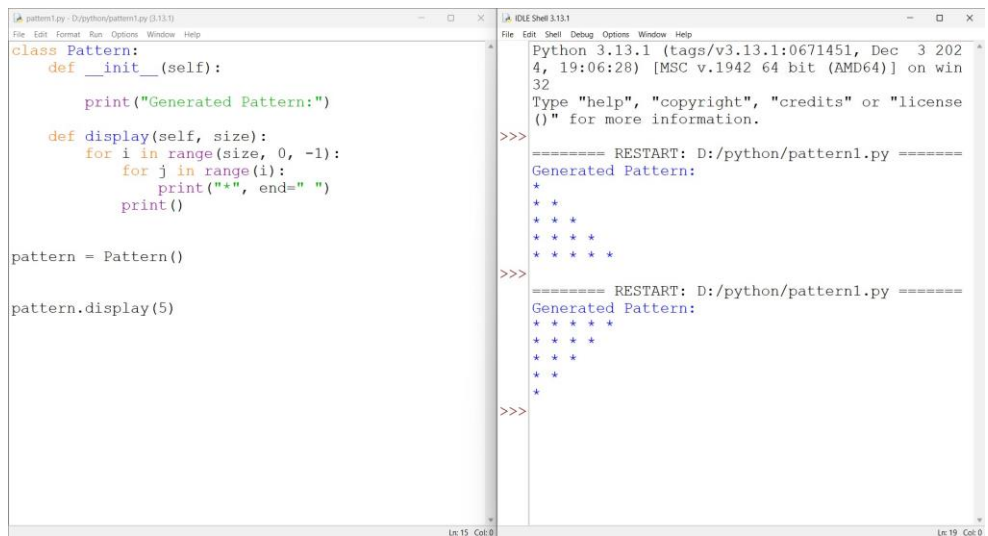
```
pattern1.py - D:\python\pattern1.py (3.13.1)
File Edit Format Run Options Window Help
class Pattern:
    def __init__(self):
        print("Generated Pattern:")

    def display(self, size):
        for i in range(size):
            for j in range(i + 1):
                print("*", end=" ")
            print()

pattern = Pattern()

pattern.display(5)
```

```
Python 3.13.1 (tags/v3.13.1:0671451, Dec 3 2024, 19:06:28) [MSC v.1942 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:/python/pattern1.py =====
Generated Pattern:
*
* *
* * *
* * * *
* * * * *
>>>
```



```
pattern1.py - D:\python\pattern1.py (3.13.1)
File Edit Format Run Options Window Help
class Pattern:
    def __init__(self):
        print("Generated Pattern:")

    def display(self, size):
        for i in range(size, 0, -1):
            for j in range(i):
                print("*", end=" ")
            print()

pattern = Pattern()

pattern.display(5)
```

```
Python 3.13.1 (tags/v3.13.1:0671451, Dec 3 2024, 19:06:28) [MSC v.1942 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:/python/pattern1.py =====
Generated Pattern:
*
* *
* * *
* * * *
* * * * *
>>>
===== RESTART: D:/python/pattern1.py =====
Generated Pattern:
* * * * *
* * * *
* * *
* *
*
>>>
```

```
pattern1.py - D:/python/pattern1.py (3.13.1)
class NumberPattern:
    def __init__(self):
        print("Generated Number Pattern:")

    def display(self, size):
        for i in range(1, size + 1):
            for j in range(1, i + 1):
                print(j, end=" ")
            print()

pattern = NumberPattern()

pattern.display(5)
```

```
IDLE Shell 3.13.1
32
Type "help", "copyright", "credits" or "license
()" for more information.
>>>
===== RESTART: D:/python/pattern1.py =====
Generated Pattern:
*
* *
* * *
* * * *
* * * * *
>>>
===== RESTART: D:/python/pattern1.py =====
Generated Pattern:
* * * * *
* * * *
* * *
* *
*
>>>
===== RESTART: D:/python/pattern1.py =====
Generated Number Pattern:
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
>>>
```

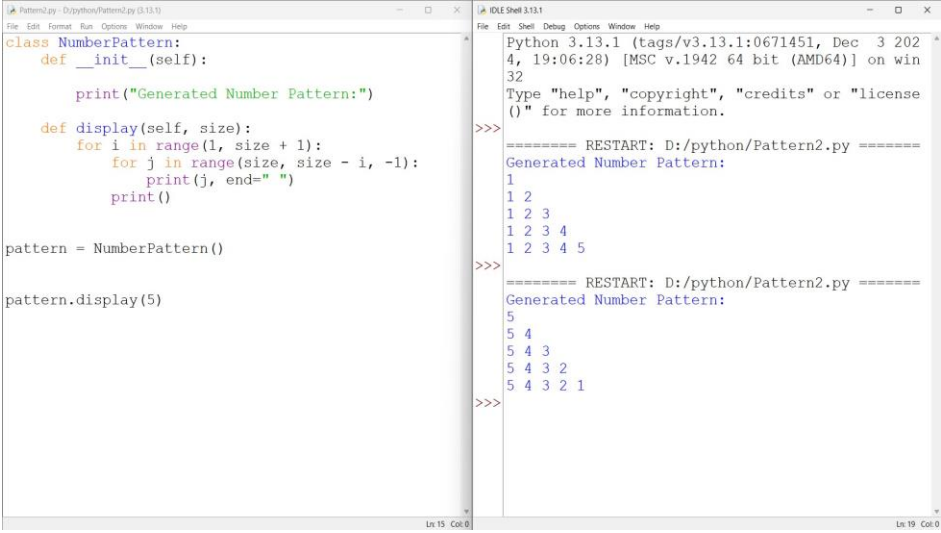
```
pattern1.py - D:/python/pattern1.py (3.13.1)
class NumberPattern:
    def __init__(self):
        print("Generated Number Pattern:")

    def display(self, size):
        for i in range(1, size + 1):
            for j in range(size, 0, -1):
                print(j, end=" ")
            print()

pattern = NumberPattern()

pattern.display(5)
```

```
IDLE Shell 3.13.1
* * * * *
* * * *
* * *
* *
*
>>>
===== RESTART: D:/python/pattern1.py =====
Generated Pattern:
* * * * *
* * * *
* * *
* *
*
>>>
===== RESTART: D:/python/pattern1.py =====
Generated Number Pattern:
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
>>>
===== RESTART: D:/python/pattern1.py =====
Generated Number Pattern:
5 4 3 2 1
5 4 3 2 1
5 4 3 2 1
5 4 3 2 1
5 4 3 2 1
>>>
```

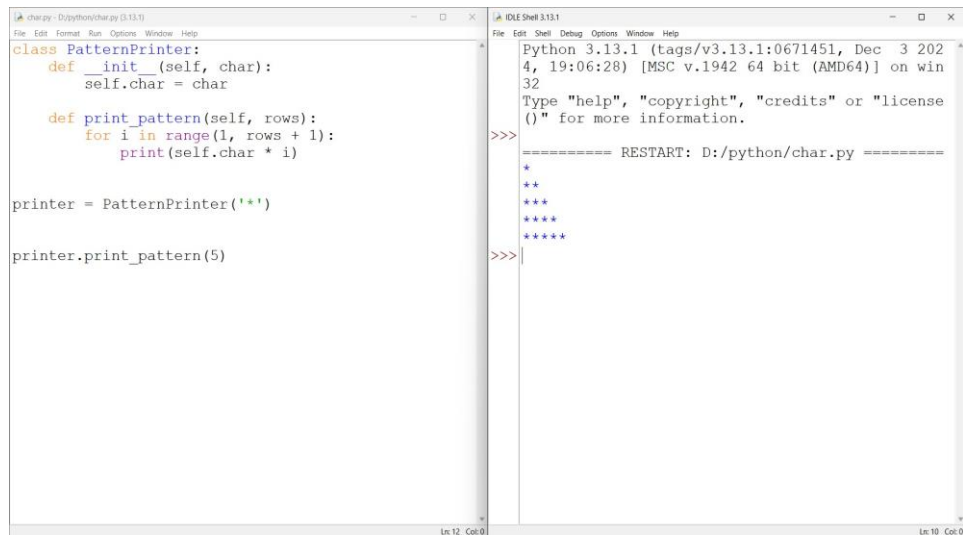
	 <pre> class NumberPattern: def __init__(self): print("Generated Number Pattern:") def display(self, size): for i in range(1, size + 1): for j in range(size, size - i, -1): print(j, end=" ") print() pattern = NumberPattern() pattern.display(5) </pre> <p>Output:</p> <pre> Generated Number Pattern: 1 1 2 1 2 3 1 2 3 4 1 2 3 4 5 </pre>
Q.2	Perform 3 pattern using string using class and object (Member function passing argument).
Code	<pre> 1. class PatternPrinter: def __init__(self, char): self.char = char def print_pattern(self, rows): for i in range(1, rows + 1): </pre>

	<pre>print(self.char * i) printer = PatternPrinter('*') printer.print_pattern(5) 2. class DiamondPattern: def __init__(self, char): self.char = char def print_diamond(self, rows): for i in range(1, rows + 1, 2): print(" " * ((rows - i) // 2) + self.char * i) for i in range(rows - 2, 0, -2): print(" " * ((rows - i) // 2) + self.char * i) printer = DiamondPattern('#') printer.print_diamond(7) 3. class SquarePattern: def __init__(self, char): self.char = char def print_square(self, size): for _ in range(size): print(self.char * size)</pre>
--	--

```
printer = SquarePattern('@')
```

```
printer.print_square(5)
```

Output



The screenshot shows two windows from an IDE. The left window, titled 'char.py - D:/python/char.py (3.13.1)', contains the following code:

```
class PatternPrinter:
    def __init__(self, char):
        self.char = char

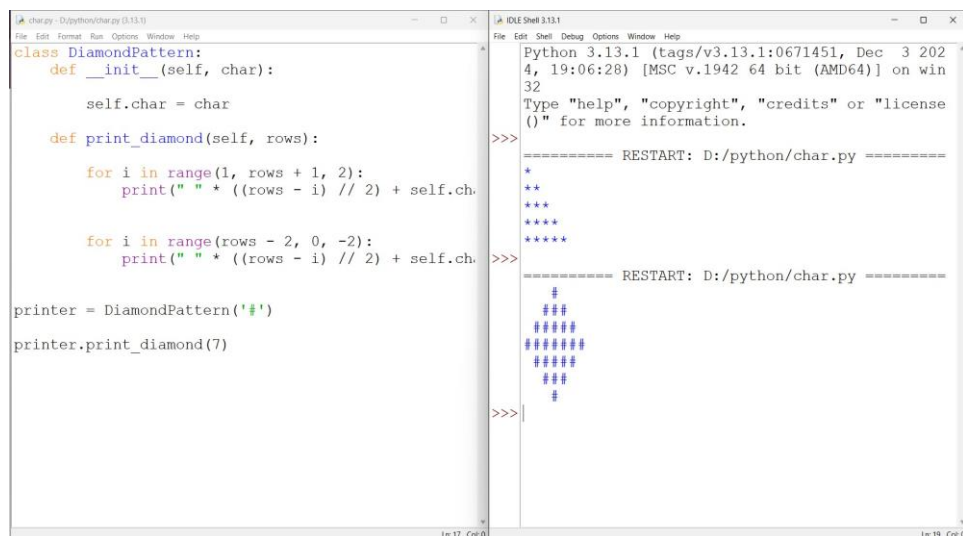
    def print_pattern(self, rows):
        for i in range(1, rows + 1):
            print(self.char * i)

printer = PatternPrinter('*')

printer.print_pattern(5)
```

The right window, titled 'IDLE Shell 3.13.1', shows the output of running the code. It displays the Python version and system information, followed by a restart message and the output of the pattern printer:

```
Python 3.13.1 (tags/v3.13.1:0671451, Dec 3 2024, 19:06:28) [MSC v.1942 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:/python/char.py =====
*
**
***
****
*****
>>>
```



The screenshot shows two windows from an IDE. The left window, titled 'char.py - D:/python/char.py (3.13.1)', contains the following code:

```
class DiamondPattern:
    def __init__(self, char):
        self.char = char

    def print_diamond(self, rows):
        for i in range(1, rows + 1, 2):
            print(" " * ((rows - i) // 2) + self.char * i)

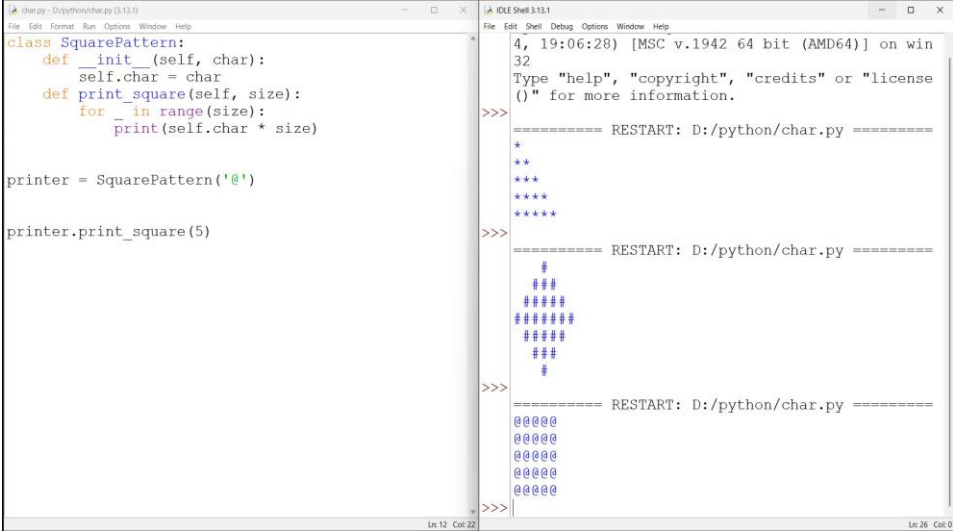
        for i in range(rows - 2, 0, -2):
            print(" " * ((rows - i) // 2) + self.char * i)

printer = DiamondPattern('#')

printer.print_diamond(7)
```

The right window, titled 'IDLE Shell 3.13.1', shows the output of running the code. It displays the Python version and system information, followed by a restart message and the output of the diamond pattern printer:

```
Python 3.13.1 (tags/v3.13.1:0671451, Dec 3 2024, 19:06:28) [MSC v.1942 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:/python/char.py =====
#
##
###
####
#####
####
###
##
#
>>>
```


	 <p>The screenshot shows a Python IDE with two windows. The left window, titled 'char.py - D:\python\char.py (3.13.1)', contains the following code:</p> <pre>class SquarePattern: def __init__(self, char): self.char = char def print_square(self, size): for _ in range(size): print(self.char * size) printer = SquarePattern('@') printer.print_square(5)</pre> <p>The right window, titled 'IDLE Shell 3.13.1', shows the execution output:</p> <pre>4, 19:06:28) [MSC v.1942 64 bit (AMD64)] on win 32 Type "help", "copyright", "credits" or "license ()" for more information. >>> ===== RESTART: D:/python/char.py ===== * * * * * >>> ===== RESTART: D:/python/char.py ===== ##### ##### ##### ##### ##### >>> ===== RESTART: D:/python/char.py ===== @@@@@ @@@@@ @@@@@ @@@@@ @@@@@ >>></pre>
<p>Q.3</p>	<p>Perform program in Class and Object</p> <ul style="list-style-type: none"> ● Fibonacci Series ● Prime Number ● Factorial ● Even and Odd ● Table of 5 ● Userinput (Medical details) ● Student Details ● Bank Details
<p>Code</p>	<pre>class Fibonacci: def generate(self, n): a, b = 0, 1 for _ in range(n): print(a, end=" ") a, b = b, a + b fib = Fibonacci() fib.generate(10)</pre>

```
2. class PrimeCheck:
    def is_prime(self, num):
        if num < 2:
            return False
        for i in range(2, int(num ** 0.5) + 1):
            if num % i == 0:
                return False
        return True

prime = PrimeCheck()
print(prime.is_prime(5))

3. class Factorial:
    def calculate(self, num):
        fact = 1
        for i in range(1, num + 1):
            fact *= i
        return fact

fact_obj = Factorial()
print(fact_obj.calculate(5))

4. class NumberCheck:
    def check(self, num):
        return "Even" if num % 2 == 0 else "Odd"

num_check = NumberCheck()
print(num_check.check(10))

5. class MedicalDetails:
    def __init__(self):
        self.name = input("Enter the Name: ")
```

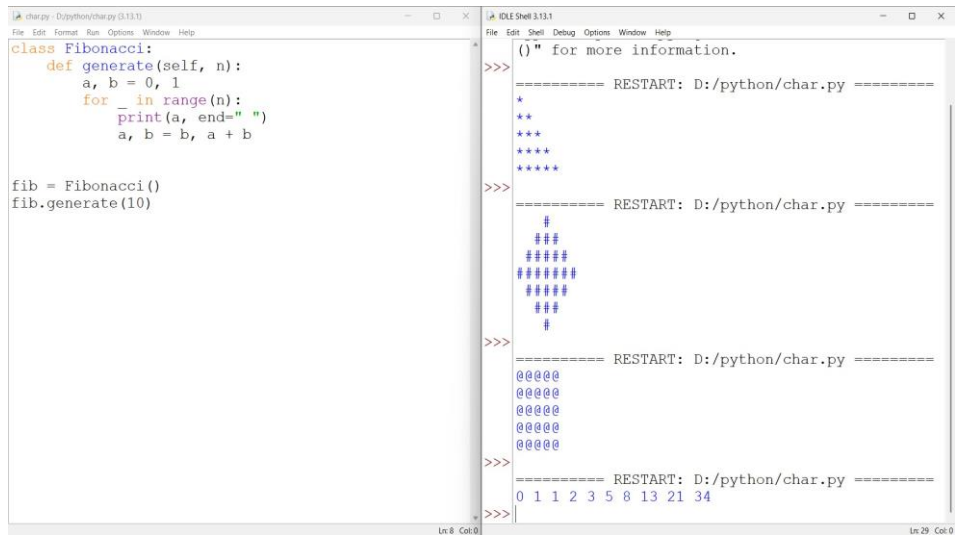
	<pre>self.age = input("Enter the Age: ") self.blood_group = input("Enter the Blood Group: ") def display(self): print(f"Name: {self.name}, Age: {self.age}, Blood Group: {self.blood_group}") person = MedicalDetails() person.display() 6. class Student: def __init__(self): self.name = input("Enter the Name: ") self.roll_no = input("Enter the Roll No: ") self.marks = input("Enter the Marks: ") def display(self): print(f"Name: {self.name}, Roll No: {self.roll_no}, Marks: {self.marks}") student = Student() student.display() 7. class BankAccount: def __init__(self): self.name = input("Enter Account Holder Name: ") self.acc_no = input("Enter Account Number: ") self.balance = input("Enter Balance: ") def display(self): print(f"Account Holder: {self.name}, Account No: {self.acc_no}, Balance: {self.balance}")</pre>
--	--

```
account = BankAccount()
account.display()
```

```
8.class Table:
    def print_table(self, num):
        for i in range(1, 11):
            print(f"{num} x {i} = {num * i}")
```

```
table = Table()
table.print_table(5)
```

Output



The screenshot shows two side-by-side windows from a Python IDE. The left window, titled 'A: charpy - D:\python\charpy (3.11.1)', contains the following code:

```
class Fibonacci:
    def generate(self, n):
        a, b = 0, 1
        for _ in range(n):
            print(a, end=" ")
            a, b = b, a + b

fib = Fibonacci()
fib.generate(10)
```

The right window, titled 'A: IDE Shell 3.11.1', shows the output of the code. It displays three restarts of the program, each showing a different pattern of asterisks and numbers. The first restart shows a single asterisk. The second restart shows a pattern of asterisks forming a triangle. The third restart shows a sequence of numbers: 0 1 1 2 3 5 8 13 21 34.

```
>>>
===== RESTART: D:/python/char.py =====
*
*
*
*
*
>>>
===== RESTART: D:/python/char.py =====
#
###
####
#####
####
###
#
>>>
===== RESTART: D:/python/char.py =====
@@@@@
@@@@@
@@@@@
@@@@@
@@@@@
>>>
===== RESTART: D:/python/char.py =====
0 1 1 2 3 5 8 13 21 34
>>>
```

	<div><div><pre>char.py - D:\python\char.py (3.13.1) File Edit Format Run Options Window Help class PrimeCheck: def is_prime(self, num): if num < 2: return False for i in range(2, int(num ** 0.5) + 1): if num % i == 0: return False return True prime = PrimeCheck() print(prime.is_prime(5))</pre></div><div><pre>IDE Shell 3.13.1 File Edit Shell Debug Options Window Help **** **** >>> ===== RESTART: D:/python/char.py ===== # ### ##### ##### ##### # >>> ===== RESTART: D:/python/char.py ===== ##### ##### ##### ##### >>> ===== RESTART: D:/python/char.py ===== 0 1 1 2 3 5 8 13 21 34 >>> ===== RESTART: D:/python/char.py ===== True >>> ===== RESTART: D:/python/char.py ===== True >>></pre></div></div> <div><div><pre>char.py - D:\python\char.py (3.13.1) File Edit Format Run Options Window Help class Factorial: def calculate(self, num): fact = 1 for i in range(1, num + 1): fact *= i return fact fact_obj = Factorial() print(fact_obj.calculate(5))</pre></div><div><pre>IDE Shell 3.13.1 File Edit Shell Debug Options Window Help ===== RESTART: D:/python/char.py ===== # ### ##### ##### ##### # >>> ===== RESTART: D:/python/char.py ===== ##### ##### ##### ##### >>> ===== RESTART: D:/python/char.py ===== 0 1 1 2 3 5 8 13 21 34 >>> ===== RESTART: D:/python/char.py ===== True >>> ===== RESTART: D:/python/char.py ===== True >>> ===== RESTART: D:/python/char.py ===== 120 >>></pre></div></div>
--	---

```
char.py - D:\python\char.py (3.13.1)
File Edit Format Run Options Window Help

class NumberCheck:
    def check(self, num):
        return "Even" if num % 2 == 0 else "Odd"

num_check = NumberCheck()
print(num_check.check(10))

IDLE Shell 3.13.1
File Edit Shell Debug Options Window Help

#####
#####
#####
#####
#####
#####
>>>
===== RESTART: D:/python/char.py =====
#####
#####
#####
#####
#####
#####
>>>
===== RESTART: D:/python/char.py =====
0 1 1 2 3 5 8 13 21 34
>>>
===== RESTART: D:/python/char.py =====
True
>>>
===== RESTART: D:/python/char.py =====
True
>>>
===== RESTART: D:/python/char.py =====
120
>>>
===== RESTART: D:/python/char.py =====
Even
>>>
```

```
char.py - D:\python\char.py (3.13.1)
File Edit Format Run Options Window Help

class Table:
    def print_table(self, num):
        for i in range(1, 11):
            print(f"{num} x {i} = {num * i}")

table = Table()
table.print_table(5)

IDLE Shell 3.13.1
File Edit Shell Debug Options Window Help

>>>
===== RESTART: D:/python/char.py =====
0 1 1 2 3 5 8 13 21 34
>>>
===== RESTART: D:/python/char.py =====
True
>>>
===== RESTART: D:/python/char.py =====
True
>>>
===== RESTART: D:/python/char.py =====
120
>>>
===== RESTART: D:/python/char.py =====
Even
>>>
===== RESTART: D:/python/char.py =====
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
>>>
```

```
char.py - D:\python\char.py (3.13.1)
class MedicalDetails:
    def __init__(self):
        self.name = input("Enter the Name: ")
        self.age = input("Enter the Age: ")
        self.blood_group = input("Enter the Blood G

    def display(self):
        print(f"Name: {self.name}, Age: {self.age},

person = MedicalDetails()
person.display()
```

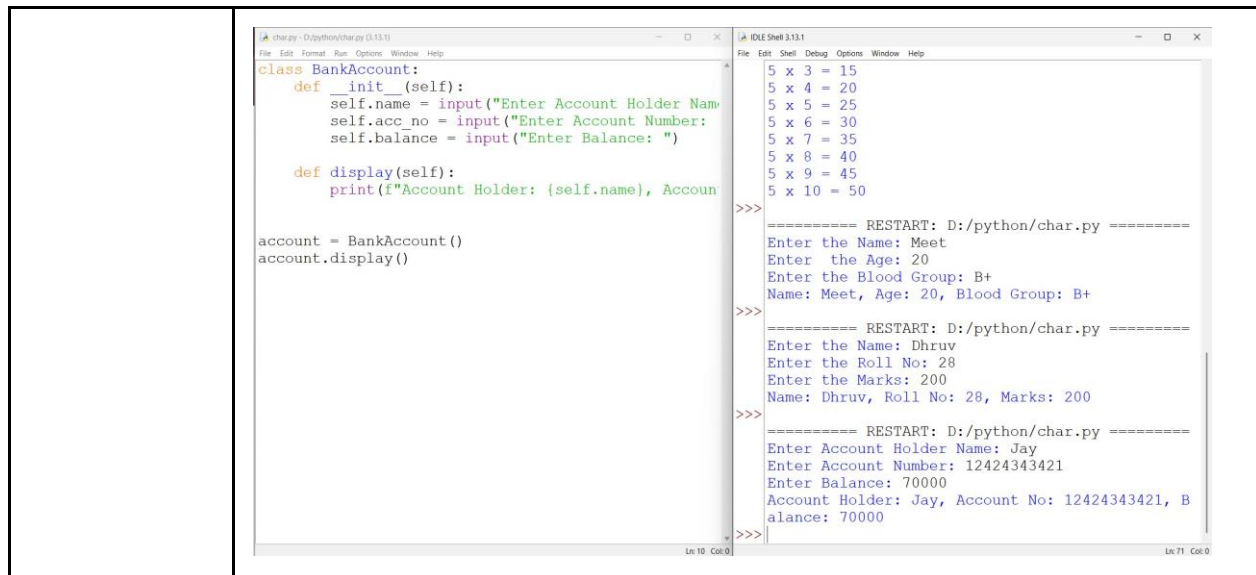
```
IDLE Shell 3.13.1
>>>
===== RESTART: D:/python/char.py =====
True
>>>
===== RESTART: D:/python/char.py =====
120
>>>
===== RESTART: D:/python/char.py =====
Even
>>>
===== RESTART: D:/python/char.py =====
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
>>>
===== RESTART: D:/python/char.py =====
Enter the Name: Meet
Enter the Age: 20
Enter the Blood Group: B+
Name: Meet, Age: 20, Blood Group: B+
>>>
```

```
char.py - D:\python\char.py (3.13.1)
class Student:
    def __init__(self):
        self.name = input("Enter the Name: ")
        self.roll_no = input("Enter the Roll No: ")
        self.marks = input("Enter the Marks: ")

    def display(self):
        print(f"Name: {self.name}, Roll No: {self.r

student = Student()
student.display()
```

```
IDLE Shell 3.13.1
>>>
===== RESTART: D:/python/char.py =====
Even
>>>
===== RESTART: D:/python/char.py =====
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
>>>
===== RESTART: D:/python/char.py =====
Enter the Name: Meet
Enter the Age: 20
Enter the Blood Group: B+
Name: Meet, Age: 20, Blood Group: B+
>>>
===== RESTART: D:/python/char.py =====
Enter the Name: Dhruv
Enter the Roll No: 28
Enter the Marks: 200
Name: Dhruv, Roll No: 28, Marks: 200
>>>
```



The image shows a screenshot of an IDE with two windows. The left window, titled 'char.py - D:/python/char.py (3.13.1)', contains the following Python code:

```
class BankAccount:
    def __init__(self):
        self.name = input("Enter Account Holder Name: ")
        self.acc no = input("Enter Account Number: ")
        self.balance = input("Enter Balance: ")

    def display(self):
        print(f"Account Holder: {self.name}, Account Number: {self.acc no}, Balance: {self.balance}")

account = BankAccount()
account.display()
```

The right window, titled 'IDLE Shell 3.13.1', shows the output of the program after three restarts. The first restart shows a multiplication table from 5x3 to 5x10. The second restart shows user input for Name, Age, and Blood Group, followed by a formatted output string. The third restart shows user input for Account Holder Name, Account Number, and Balance, followed by a formatted output string.

```
>>>
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
>>>
===== RESTART: D:/python/char.py =====
Enter the Name: Meet
Enter the Age: 20
Enter the Blood Group: B+
Name: Meet, Age: 20, Blood Group: B+
>>>
===== RESTART: D:/python/char.py =====
Enter the Name: Dhruv
Enter the Roll No: 28
Enter the Marks: 200
Name: Dhruv, Roll No: 28, Marks: 200
>>>
===== RESTART: D:/python/char.py =====
Enter Account Holder Name: Jay
Enter Account Number: 12424343421
Enter Balance: 70000
Account Holder: Jay, Account No: 12424343421, Balance: 70000
>>>
```