

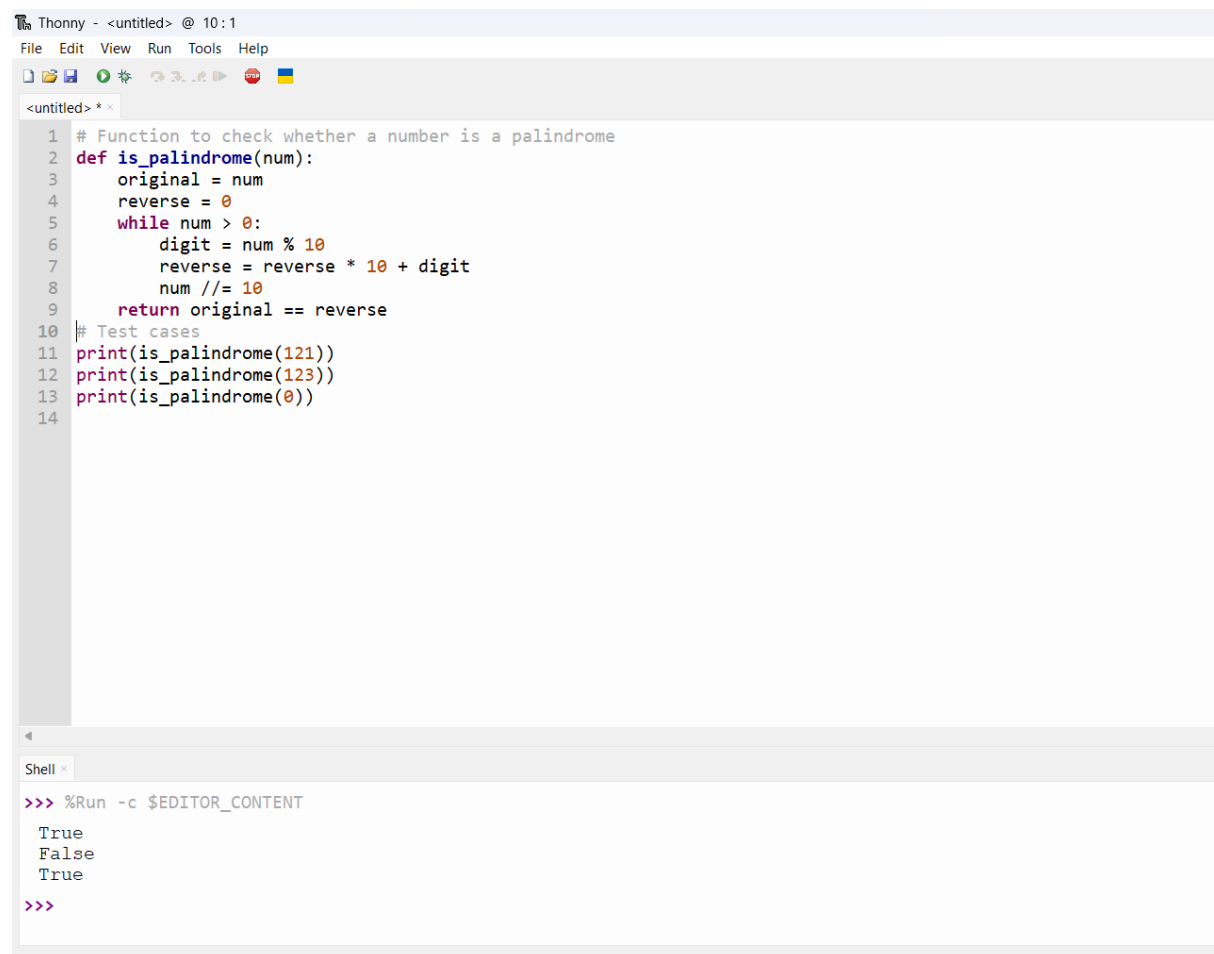
ASSIGNMENT-3

NAME:-N.ROHAN

BATCH-16

HTNO:-2303A51064

Zero-Shot Prompting – Palindrome Number Program:-

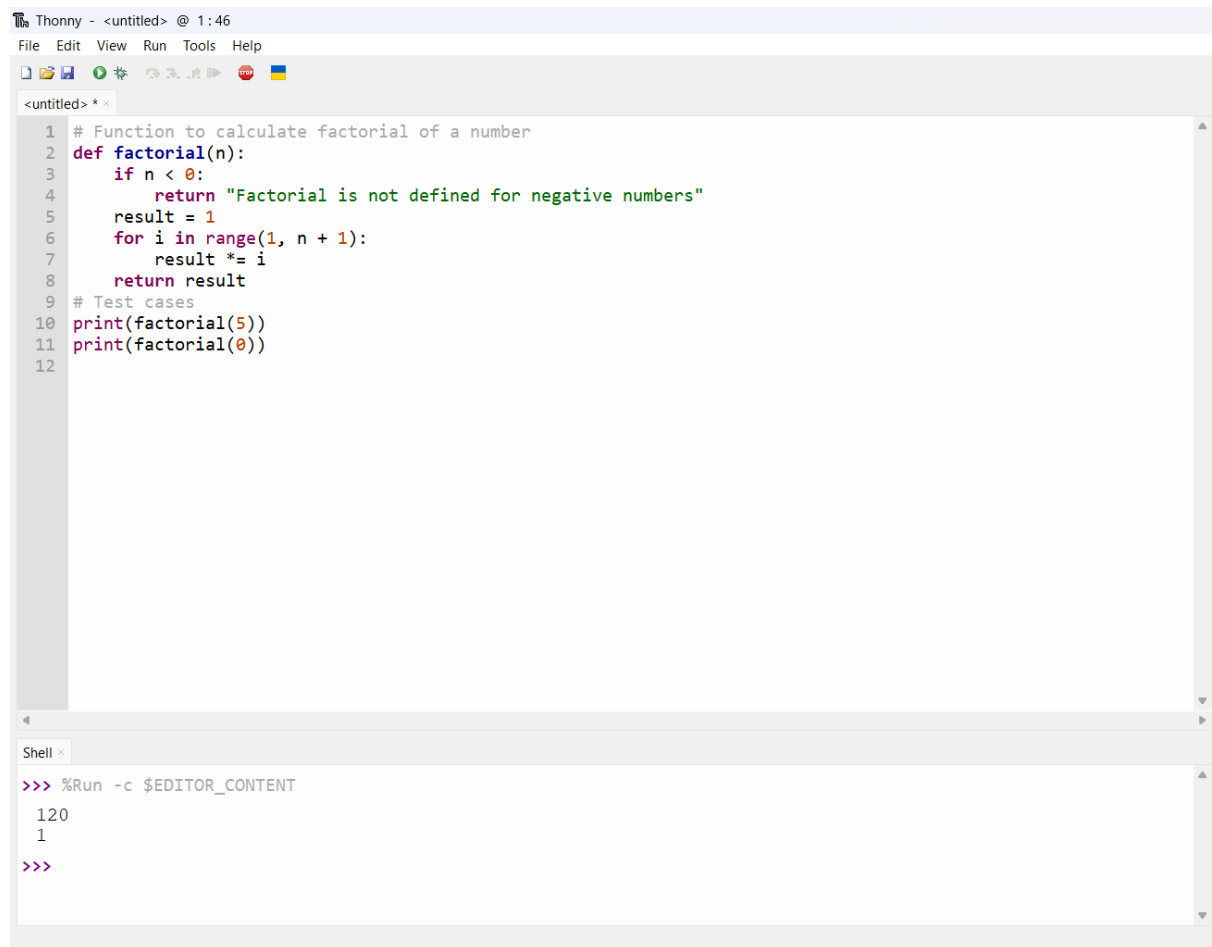


The screenshot displays the Thonny IDE interface. The top pane shows a Python script for checking palindromes. The script defines a function `is_palindrome` that reverses a number and compares it to the original. It then tests the function with 121, 123, and 0. The bottom pane shows the shell output, which prints `True`, `False`, and `True` for the respective inputs.

```
1 # Function to check whether a number is a palindrome
2 def is_palindrome(num):
3     original = num
4     reverse = 0
5     while num > 0:
6         digit = num % 10
7         reverse = reverse * 10 + digit
8         num //= 10
9     return original == reverse
10 # Test cases
11 print(is_palindrome(121))
12 print(is_palindrome(123))
13 print(is_palindrome(0))
14
```

```
>>> %Run -c $EDITOR_CONTENT
True
False
True
>>>
```

One-Shot Prompting – Factorial Calculation:-

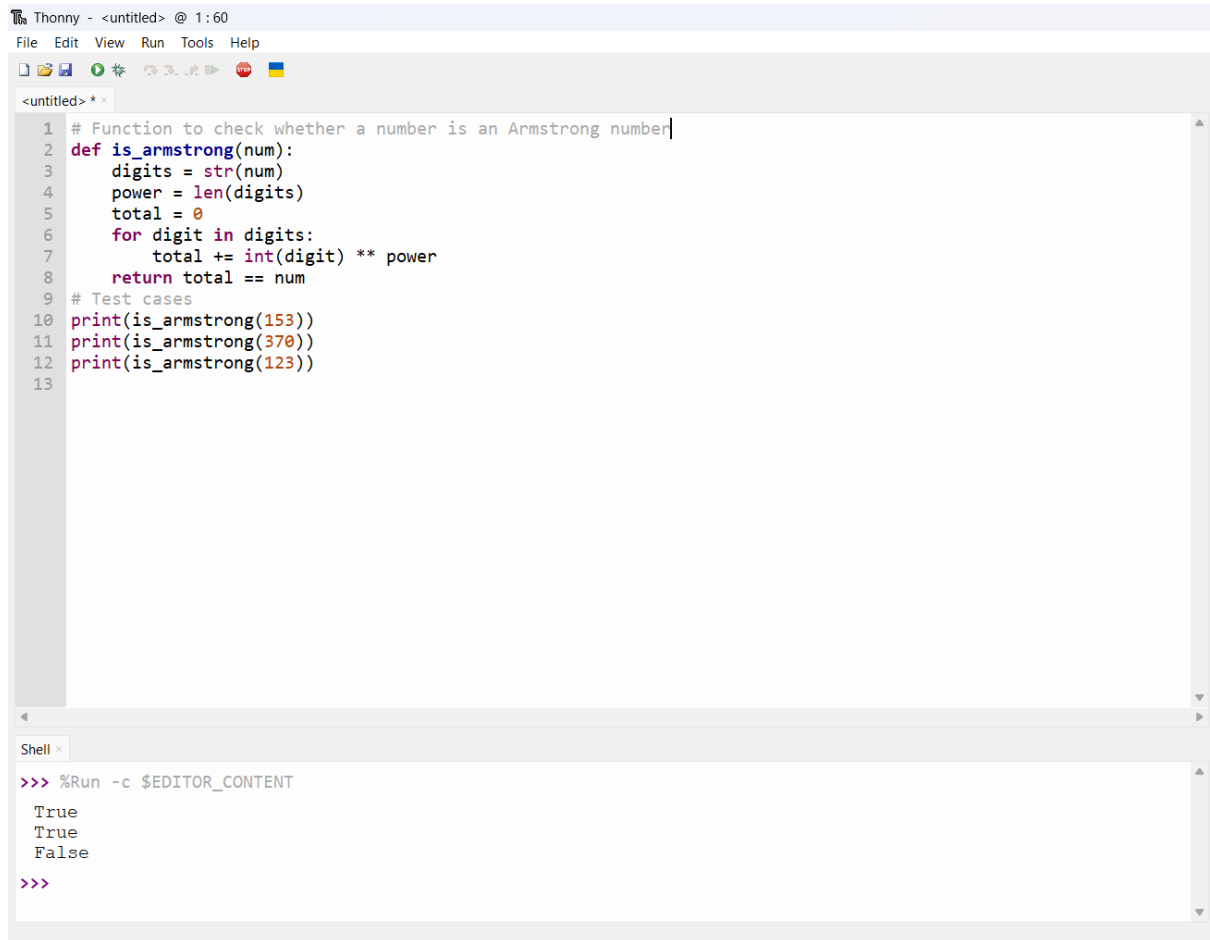


The image shows a screenshot of the Thonny Python IDE. The top window, titled "<untitled> * x", contains a Python script for calculating the factorial of a number. The script defines a function `factorial(n)` that returns an error message for negative numbers and calculates the factorial for non-negative numbers. It also includes test cases for `factorial(5)` and `factorial(0)`. The bottom window, titled "Shell x", shows the output of running the script, displaying the results of the test cases: 120 and 1.

```
1 # Function to calculate factorial of a number
2 def factorial(n):
3     if n < 0:
4         return "Factorial is not defined for negative numbers"
5     result = 1
6     for i in range(1, n + 1):
7         result *= i
8     return result
9 # Test cases
10 print(factorial(5))
11 print(factorial(0))
12
```

```
>>> %Run -c $EDITOR_CONTENT
120
1
>>>
```

Few-Shot Prompting – Armstrong Number Check:-



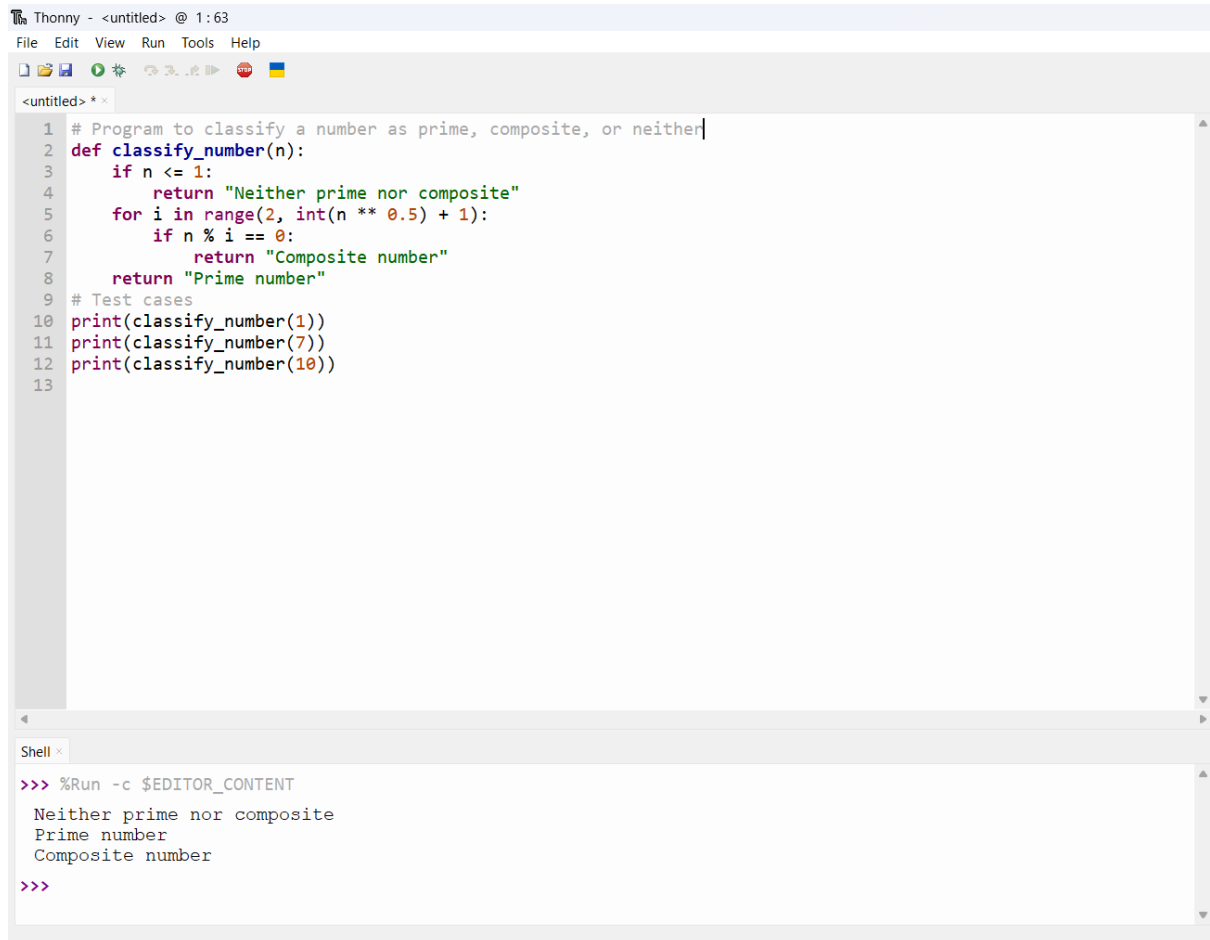
The image shows a screenshot of the Thonny Python IDE. The main editor window displays a Python script with the following code:

```
1 # Function to check whether a number is an Armstrong number
2 def is_armstrong(num):
3     digits = str(num)
4     power = len(digits)
5     total = 0
6     for digit in digits:
7         total += int(digit) ** power
8     return total == num
9 # Test cases
10 print(is_armstrong(153))
11 print(is_armstrong(370))
12 print(is_armstrong(123))
13
```

Below the editor, the Shell window shows the output of the script:

```
>>> %Run -c $EDITOR_CONTENT
True
True
False
>>>
```

Question 4 (Optional): Context-Managed Prompting – Number Classification:-



The image shows a screenshot of the Thonny Python IDE. The main editor window displays a Python script for classifying numbers. The script defines a function `classify_number(n)` that returns "Neither prime nor composite" for `n <= 1`, "Composite number" for numbers with divisors other than 1 and themselves, and "Prime number" for prime numbers. Below the function definition, there are test cases for `1`, `7`, and `10`. The bottom panel, titled "Shell", shows the output of running the script: "Neither prime nor composite", "Prime number", and "Composite number".

```
1 # Program to classify a number as prime, composite, or neither
2 def classify_number(n):
3     if n <= 1:
4         return "Neither prime nor composite"
5     for i in range(2, int(n ** 0.5) + 1):
6         if n % i == 0:
7             return "Composite number"
8     return "Prime number"
9 # Test cases
10 print(classify_number(1))
11 print(classify_number(7))
12 print(classify_number(10))
13
```

Shell

```
>>> %Run -c $EDITOR_CONTENT
Neither prime nor composite
Prime number
Composite number
>>>
```

Question 5: Zero-Shot Prompting – Perfect Number Check:-

The image shows the Thonny Python IDE interface. The top menu bar includes File, Edit, View, Run, Tools, and Help. Below the menu is a toolbar with icons for file operations and execution. The main editor window, titled '<untitled> * x', contains the following Python code:

```
1 # Function to check whether a number is a perfect number
2 def is_perfect(num):
3     if num <= 0:
4         return False
5     total = 0
6     for i in range(1, num):
7         if num % i == 0:
8             total += i
9     return total == num
10 # Test cases
11 print(is_perfect(6))
12 print(is_perfect(28))
13 print(is_perfect(12))
14
```

Below the editor is a Shell window titled 'Shell x'. It shows the command prompt with the command `>>> %Run -c $EDITOR_CONTENT` and the output of the script:

```
>>> %Run -c $EDITOR_CONTENT
True
True
False
>>>
```

Few-Shot Prompting – Even or Odd with Validation:-



<untitled> * x

```
1 # Program to check whether a number is even or odd with input validation
2 def even_or_odd(value):
3     if not isinstance(value, int):
4         return "Invalid input. Please enter an integer."
5     if value % 2 == 0:
6         return "Even"
7     else:
8         return "Odd"
9 # Test cases
10 print(even_or_odd(8))
11 print(even_or_odd(15))
12 print(even_or_odd(0))
13 print(even_or_odd(-7))
14
```

Shell x

>>> %Run -c \$EDITOR_CONTENT

```
Even
Odd
Even
Odd
```

>>>