Class Test 1

Q.1

→ A Control flow statement defines the flow of execution of program.

There are 6 different types of control flow statements in Python:-

1. if-else :- If-else is used for decision making when code statements satisfy the condition, then it will return non-zero values as true, or else zero or NONE value as false.

Syntax:

```
if (condition):
    statement 1
    ...
else
    statement 2
    ...
```

2. Nested if-else :- With if... elif... else, elif is a shortened form of else if. It works the same as 'if' statements where the if block condition is false then it checks to elif blocks. If all the blocks are false, then it executes an else statement. There are multiple elif blocks possible for a Nested if...else.

Syntax:

```
if (condition1)
    statement 1
    ...
elif (condition2)
    statement 2
    ...
else :
    statement 3
    ...
```

3. for statement :- The for loop statement has variable iteration in a sequence (list, tuple or string), and executes statements until the loop does not reach the false condition.

Syntax:

```
for value in sequence
    ... body statement of for
```

4. while loop:- A while loop is used in python to iterate over the block of expression which matches to true. Non-zero values are true and zero and negative values are false.

Syntax:

```
while (condition)
    statement 1:.
```

5. Break statement :- The python Break statement is used to break a loop in a certain condition. It terminates the loop. Also, we can use it inside the loop then it will first terminate the innermost loop.

Syntax:

```
for value in sequence
    ... body statement of for
  if (condition):
      break
  ... body statement of for loop.

... body statement outside of for loop
```

6. Continue statement :- A continue statement won't continue the loop, it executes statements until the condition matches true.

Syntax:

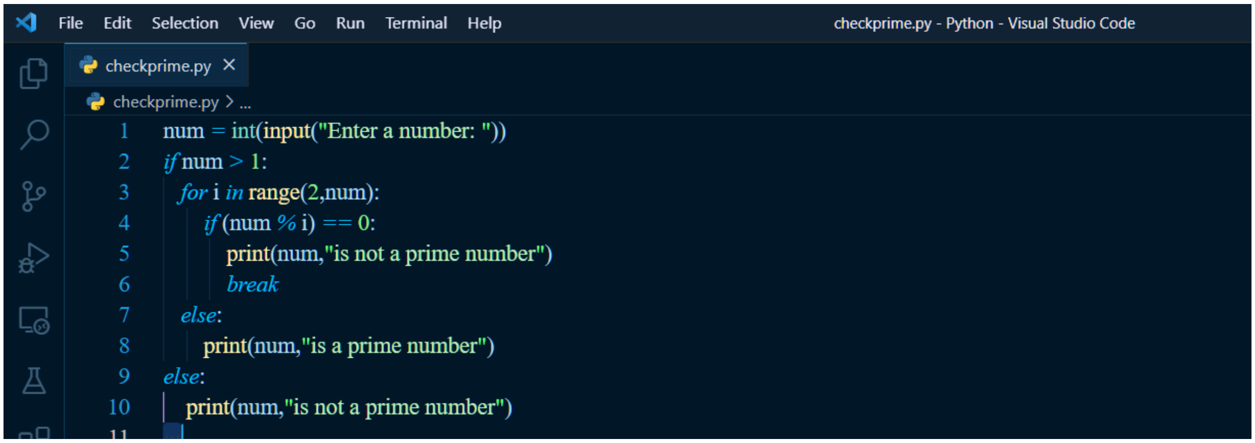
```
for value in sequence:
... body statement of for
    if (condition):
        continue
    ... body statement of for loop

... body statement outside of for loop.
```

Q4.

	Break	Continue
1.	It eliminates the execution of remaining iteration of loop	It will terminate only the current iteration of loop.
2.	'break' will resume control of program to the end of loop enclosing that 'break'.	The 'continue' will resume the control of program to next iteration of loop enclosing 'continue'
3.	It causes early termination of loop	It causes early execution of ~~loop~~ next iteration
4.	The 'break' stop the continuation of loop	The 'continue' does not stop continuation of loop and stops the current
5.	It is used with 'switch', 'label'.	Cannot be executed with switch and labels.

Q2.

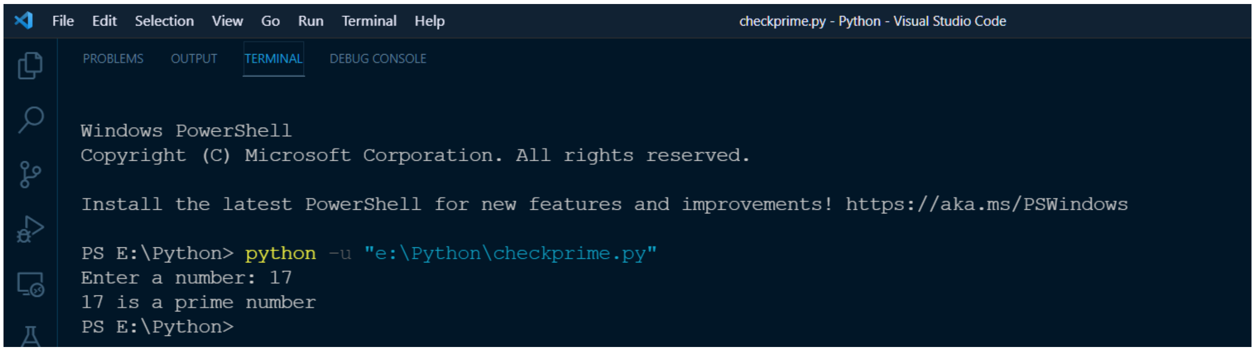
Program:


File Edit Selection View Go Run Terminal Help
checkprime.py - Python - Visual Studio Code
checkprime.py
checkprime.py > ...

```python
num = int(input("Enter a number: "))
if num > 1:
  for i in range(2,num):
    if (num % i) == 0:
      print(num,"is not a prime number")
      break
  else:
    print(num,"is a prime number")
else:
  print(num,"is not a prime number")
```

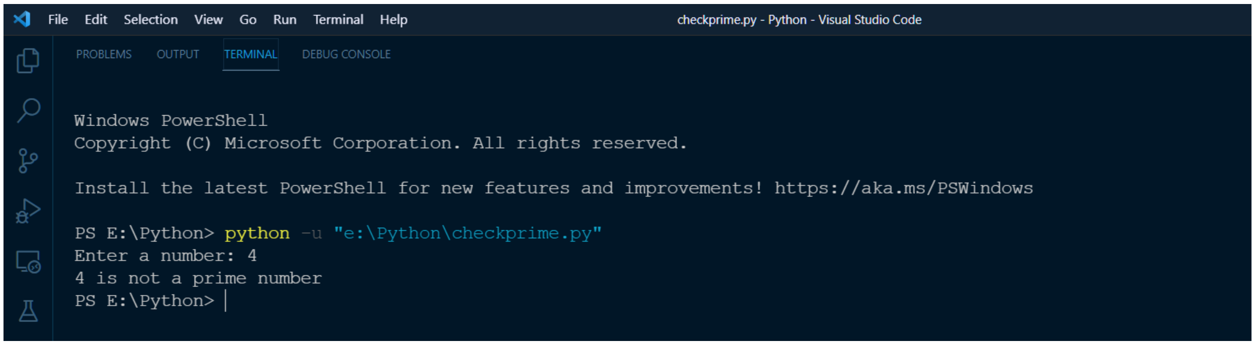
Output:

Case 1 - Number = 17


File Edit Selection View Go Run Terminal Help
checkprime.py - Python - Visual Studio Code
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS E:\Python> python -u "e:\Python\checkprime.py"
Enter a number: 17
17 is a prime number
PS E:\Python>
```

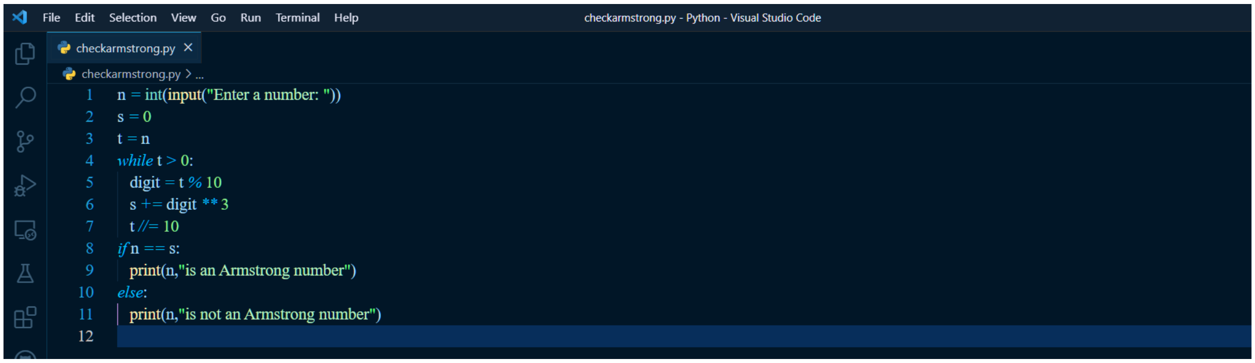
Case 2 - Number = 4


File Edit Selection View Go Run Terminal Help
checkprime.py - Python - Visual Studio Code
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS E:\Python> python -u "e:\Python\checkprime.py"
Enter a number: 4
4 is not a prime number
PS E:\Python>
```

Q3.

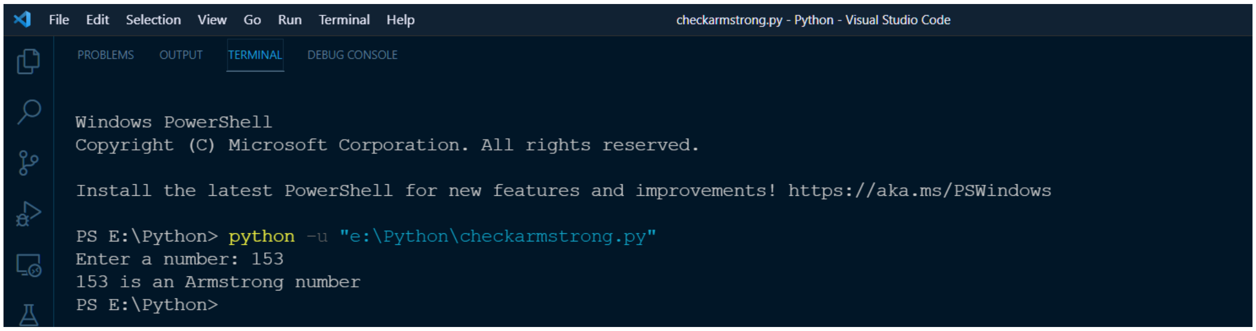
Program:


File Edit Selection View Go Run Terminal Help
checkarmstrong.py - Python - Visual Studio Code
checkarmstrong.py ×
checkarmstrong.py > ...

```python
n = int(input("Enter a number: "))
s = 0
t = n
while t > 0:
    digit = t % 10
    s += digit ** 3
    t //= 10
if n == s:
    print(n,"is an Armstrong number")
else:
    print(n,"is not an Armstrong number")
```

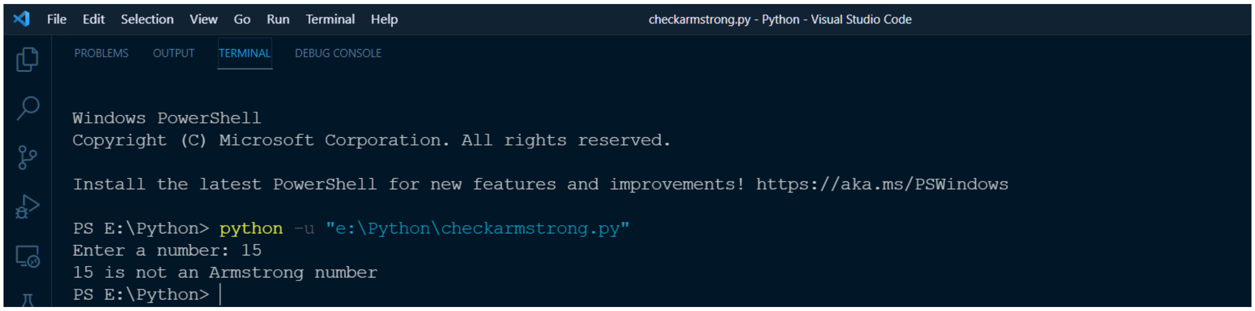
Output:

Case 1 - Number = 153


File Edit Selection View Go Run Terminal Help
checkarmstrong.py - Python - Visual Studio Code
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS E:\Python> python -u "e:\Python\checkarmstrong.py"
Enter a number: 153
153 is an Armstrong number
PS E:\Python>
```

Case 2 - Number = 15


File Edit Selection View Go Run Terminal Help
checkarmstrong.py - Python - Visual Studio Code
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS E:\Python> python -u "e:\Python\checkarmstrong.py"
Enter a number: 15
15 is not an Armstrong number
PS E:\Python>
```