CourseName: Advance Python Programming **Course code:** 22CSH-623

Experiment-1.3

A) Types of Operators

Aim: Write a python program to illustrate the concept of different types of operators.

Tools/Software Required: VS Code, Python

Description:

Operators are used for performing operations on variables and values. Python divides the operators into the following groups:

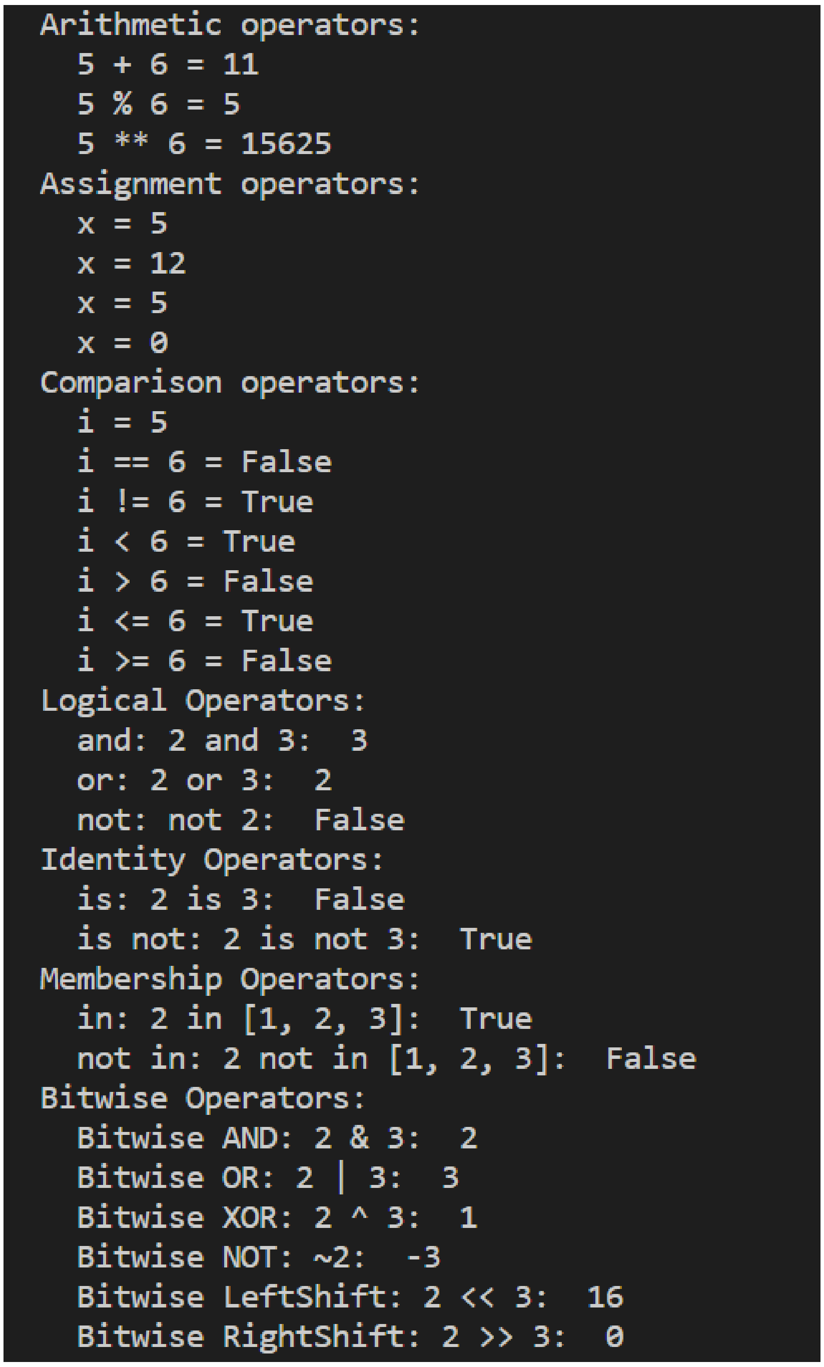
- **Arithmetic operators:** +, -, *, /, %, **, //
- **Assignment operators:** =, +=, -=, *=, /=, %=, //=, **=, &=, |=, ^=, >>=, <<=
- **Comparison operators:** ==, !=, <, >, <=, >=,
- **Logical operators:** and, or, not
- **Identity operators:** is, is not
- **Membership operators:** in, not in
- **Bitwise operators:** &, |, ^, ~, <<, >>

Implementation:

```
print("Arithmetic operators:")
print("  5 + 6 =", 5 + 6)
print("  5 % 6 =", 5 % 6)
print("  5 ** 6 =", 5 ** 6)

print("Assignment operators:")
x = 5
print("  x =", x)
x += 7
print("  x =", x)
x %= 7
print("  x =", x)
```

```
x //= 7
print("  x =", x)
x **= 7

print("Comparison operators:")
i = 5
print("  i =", i)
print("  i == 6 =", i == 6)
print("  i != 6 =", i != 6)
print("  i < 6 =", i < 6)
print("  i > 6 =", i > 6)
print("  i <= 6 =", i <= 6)
print("  i >= 6 =", i >= 6)

print("Logical Operators:")
print("  and: 2 and 3: ", 2 and 3)
print("  or: 2 or 3: ", 2 or 3)
print("  not: not 2: ", not 2)

print("Identity Operators:")
print("  is: 2 is 3: ", 2 is 3)
print("  is not: 2 is not 3: ", 2 is not 3)

print("Membership Operators:")
print("  in: 2 in [1, 2, 3]: ", 2 in [1, 2, 3])
print("  not in: 2 not in [1, 2, 3]: ", 2 not in [1,
2, 3])

print("Bitwise Operators:")
print("  Bitwise AND: 2 & 3: ", 2 & 3)
print("  Bitwise OR: 2 | 3: ", 2 | 3)
print("  Bitwise XOR: 2 ^ 3: ", 2 ^ 3)
print("  Bitwise NOT: ~2: ", ~2)
print("  Bitwise LeftShift: 2 << 3: ", 2 << 3)
print("  Bitwise RightShift: 2 >> 3: ", 2 >> 3)
```

Output:



```
Arithmetic operators:
  5 + 6 = 11
  5 % 6 = 5
  5 ** 6 = 15625
Assignment operators:
  x = 5
  x = 12
  x = 5
  x = 0
Comparison operators:
  i = 5
  i == 6 = False
  i != 6 = True
  i < 6 = True
  i > 6 = False
  i <= 6 = True
  i >= 6 = False
Logical Operators:
  and: 2 and 3:  3
  or: 2 or 3:  2
  not: not 2:  False
Identity Operators:
  is: 2 is 3:  False
  is not: 2 is not 3:  True
Membership Operators:
  in: 2 in [1, 2, 3]:  True
  not in: 2 not in [1, 2, 3]:  False
Bitwise Operators:
  Bitwise AND: 2 & 3:  2
  Bitwise OR: 2 | 3:  3
  Bitwise XOR: 2 ^ 3:  1
  Bitwise NOT: ~2:  -3
  Bitwise LeftShift: 2 << 3:  16
  Bitwise RightShift: 2 >> 3:  0
```

B) Armstrong or Not

Aim: Write a python program to check whether the number is Armstrong or not.

Tools/Software Required: VS Code, Python

Description: Armstrong number is a number that is equal to the sum of cubes of its digits. For example 0, 1, 153, 370, 371, and 407 are the Armstrong numbers.

153 = (1*1*1)+(5*5*5)+(3*3*3)

where:

(1*1*1)=1

(5*5*5)=125

(3*3*3)=27

So:

1+125+27=153

Pseudo Code:

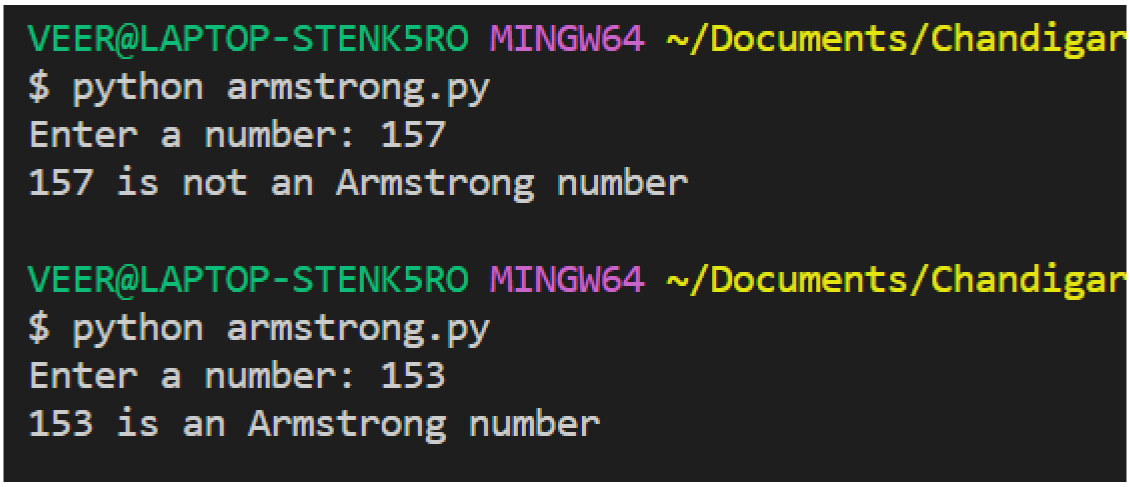
```
1. Take input from the user.
2. Initialize sum = 0
3. Use a while loop to find the sum of the cube of each digit.
4. Display the result.
```

Implementation:

```python
num = int(input("Enter a number: "))
sum = 0
temp = num

while temp > 0:
    digit = temp % 10
    sum += digit ** 3
    temp //= 10
```

```
# display the result
if num == sum:
    print(num, "is an Armstrong number")
else:
    print(num, "is not an Armstrong number")
```

Output:



```
VEER@LAPTOP-STENK5RO MINGW64 ~/Documents/Chandigar
$ python armstrong.py
Enter a number: 157
157 is not an Armstrong number

VEER@LAPTOP-STENK5RO MINGW64 ~/Documents/Chandigar
$ python armstrong.py
Enter a number: 153
153 is an Armstrong number
```

C) Sum of N natural numbers

Aim: Write a python program to find the sum of first n natural numbers.

Tools/Software Required: VS Code, Python

Description:

The sum of the first n natural numbers is given by the formula:

$$sum = n(n+1)/2$$

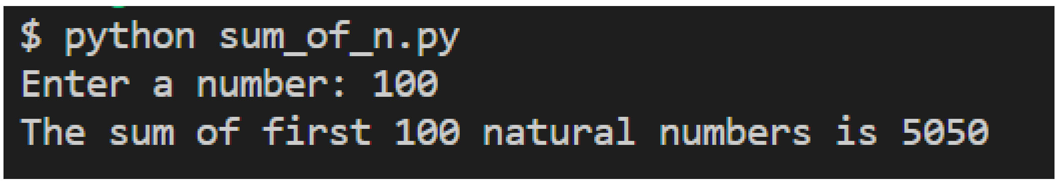
Pseudo Code:

```
1. Take input from the user.
2. Initialize sum = 0
3. Use a while loop to find the sum of the first n natural numbers using the formula.
4. Display the result.
```

Implementation:

```
# Take input from the user
num = int(input("Enter a number: "))
# Initialize sum
sum = 0
# find the sum of the first n natural numbers using
the formula
sum = num * (num + 1) / 2
# display the result
print("The sum of first", num, "natural numbers is",
int(sum))
```

Output:



```
$ python sum_of_n.py
Enter a number: 100
The sum of first 100 natural numbers is 5050
```

D) Palindrome or Not

Aim: Write a python program to check whether the given number is palindrome or not

Tools/Software Required: VS Code, Python

Description:
A number is called a palindrome if the number and its reverse are equal. For example, 121, 131, 34543, 343, 171, and 48984 are the palindrome numbers.

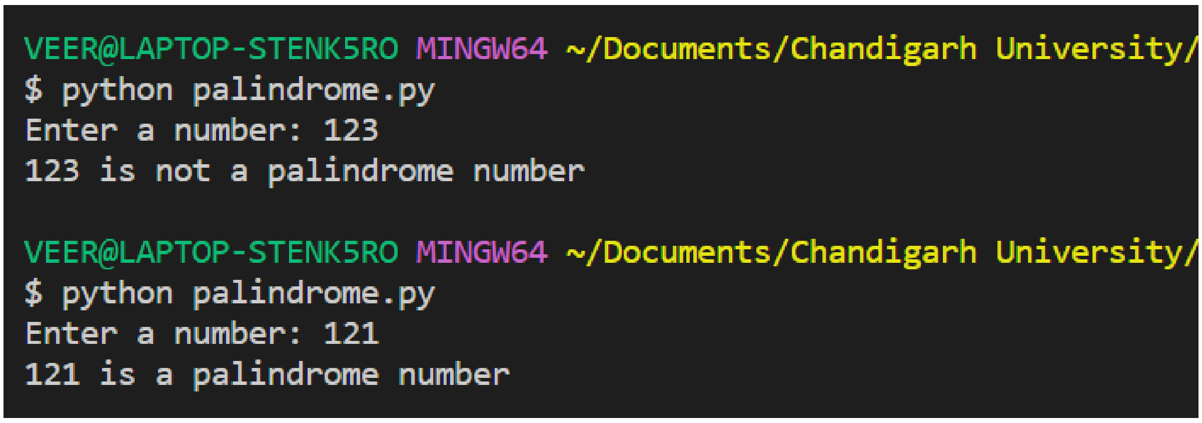
Pseudo Code:

1. Take the value of the integer and store it in a variable.
2. Transfer the value of the integer into another temporary variable.
3. Using a while loop, get each digit of the number and store the reversed number in another variable.
4. Check if the reverse of the number is equal to the one in the temporary variable.
5. Print the final result.
6. Exit.

Implementation:

```python
num = int(input("Enter a number: "))
temp = num
rev = 0
while temp > 0:
    digit = temp % 10
    rev = rev * 10 + digit
    temp //= 10

if num == rev:
    print(num, "is a palindrome number")
else:
    print(num, "is not a palindrome number")
```

Output:



```
VEER@LAPTOP-STENK5RO MINGW64 ~/Documents/Chandigarh University/
$ python palindrome.py
Enter a number: 123
123 is not a palindrome number

VEER@LAPTOP-STENK5RO MINGW64 ~/Documents/Chandigarh University/
$ python palindrome.py
Enter a number: 121
121 is a palindrome number
```