COURSE CODE : R1UC427B

BACHLORES OF ENGINEERING AND TECHNOLOGY

STUDENT NAME : AMAN PATEL

ADMISSION NO : 24SCSE1260003

SEMESTER : IV

SECTION : 50

SCHOOL OF COMPUTER SCIENCE AND ENGNEERING

GALGOTIAS UNIVERSITY, GREATER NOIDA

SUBMITTED TO : DR. SAPNA SHARMA

(PROFESSOR)

python
SOFTWARE FOUNDATION

SCSE

LAB FILE

PYTHON PROGRAMMING

PYTHON PROGRAMMING PRACTICAL – INDEX

S.No	Practical Name	Date	Progress	Teacher's Signature
1	**Practical 1:** Python Data Types & Variables	18th February, 2026	Completed ✅	
2	**Practical 2:** Operators in Python	18th February, 2026	Completed ✅	
3	**Practical 3:** Conditional Statements	25th February, 2026	Completed ✅	
4	**Practical 4:** Looping Statements	25th February, 2026	Completed ✅	
5	**Practical 5:** Pattern Printing			
6	**Practical 6:** Strings			
7	**Practical 7:** Lists			
8	**Practical 8:** Tuples			
9	**Practical 9:** Sets and Dictionary			
10	**Practical 10:** Functions			
11	**Practical 11:** Lambda Function, Modules & Math Module			
12	**Practical 12:** Creating Class and Object			
13	**Practical 13:** Constructors			

14	**Practical 14:** Inheritance & OOP Concepts			
15	**Practical 15:** NumPy & Pandas			
16	**Practical 16:** Mini Projects & Visualization			

EXPERIMENT 1

Aim: To understand Python variables and data types and perform basic variable operations.

Programmes:

1.1 Program to Print "Hello, World"

Aim: To write a basic Python program to print output.

Algorithm:

1. Start the program.
2. Use the print() function to display a message.
3. Stop the program.

Program:

```
print("Hello, World")
print("Aman Patel, 24SCSE1260003")
```

Output:

```
>>> Hello, World
    Aman Patel, 24SCSE1260003
```

1.2 Program to Declare and Print Different Types of Variables.

Aim: To declare variables of different data types and print them.

Algorithm:

1. Start the program.
2. Declare integer, float, string, and Boolean variables.
3. Use print() to display the values.
4. Stop the program.

Program:

```
a = 10 # Integer
b = 20.5 # Float
c = "Python" # String
d = True # Boolean
print("Integer:", a)
print("Float:", b)
print("String:", c)
print("Boolean:", d)
print("Aman Patel, 24SCSE1260003")
```

Sample Output:

```
>>> Integer: 10
    Float: 20
    5 String: Python
    Boolean: True
    Aman Patel, 24SCSE1260003
```

1.3 Program to Take User Input and Display Its Data Type.

Aim: To accept input from the user and display its data type.

Algorithm:

1. Start the program.
2. Use input() to read a value from the user.
3. Display the value.
4. Use type() to display the data type.
5. Stop the program.

Program:

```
x = input("Enter a value: ")
print("Value:", x)
print("Data Type:", type(x))
print("Aman Patel, 24SCSE1260003")
```

Output:

```
>>> Enter a value: 25
    Value: 25
    Data Type: <class 'str'>
    Aman Patel, 24SCSE1260003
```

1.4 Program to Swap Two Variables.

Aim: To swap values of two variables.

Algorithm:

1. Start the program.
2. Assign values to two variables.
3. Display values before swapping.
4. Swap the values using Python multiple assignment.
5. Display values after swapping.
6. Stop the program.

Program:

```
a = 5
b = 10
print("Before swapping : ", a, b)
a, b = b, a
print("After swapping : ", a, b)
print("Aman Patel, 24SCSE1260003")
```

Output:

```
>>> Before swapping : 5 10
    After swapping : 10 5
    Aman Patel, 24SCSE1260003
```

1.5 Program to Assign Multiple Variables in a Single Line.

Aim: To assign multiple variables in a single statement.

Algorithm:

1. Start the program.
2. Assign multiple values to variables in a single line.
3. Print the variable values.
4. Stop the program.

Program:

```
a, b, c = 1, 2, 3
print(a, b, c)
print("Aman Patel, 24SCSE1260003")
```

Output:

```
>>> 1 2 3
    Aman Patel, 24SCSE1260003
```

Result: Thus, the Python programs for printing output, declaring and printing different data types, taking user input and displaying its data type, swapping two variables, and assigning multiple variables in a single line were successfully executed and the outputs were verified.

EXPERIMENT 2

Aim: To understand and use different operators in Python.

Programmes:

2.1 Program Using Arithmetic Operators.

Aim: To perform arithmetic operations using Python operators.

Algorithm:

1. Start the program.
2. Assign values to two variables.
3. Perform addition, subtraction, multiplication, division, and modulus operations.
4. Display the results.
5. Stop the program.

Program:

```
a = 10
b = 5
print("Addition:", a + b)
print("Subtraction:", a - b)
print("Multiplication:", a * b)
print("Division:", a / b)
print("Modulus:", a % b)
print("Aman Patel, 24SCSE1260003")
```

Output:

```
>>> Addition: 15
    Subtraction: 5
    Multiplication: 50
    Division: 2.0
    Modulus: 0
    Aman Patel, 24SCSE1260003
```

2.2 Program Using Relational Operators.

Aim: To compare two values using relational operators.

Algorithm:

1. Start the program.
2. Assign values to two variables.
3. Use relational operators to compare values.
4. Display the comparison results.
5. Stop the program.

Program:

```
a = 10
b = 5
print(a > b)
print(a < b)
print(a == b)
print(a != b)
print("Aman Patel, 24SCSE1260003")
```

Output:

```
>>> True
    False
    False
    True
    Aman Patel, 24SCSE1260003
```

2.3 Program Using Logical Operators.

Aim: To use logical operators in Python.

Algorithm:

1. Start the program.
2. Assign Boolean values to variables.
3. Apply logical AND, OR, and NOT operators.
4. Display the results.
5. Stop the program.

Program:

```
a = True
b = False
print(a and b)
print(a or b)
print(not a)
print("Aman Patel, 24SCSE1260003")
```

Output:

```
>>> False
    True
    False
    Aman Patel, 24SCSE1260003
```

2.4 Program Using Assignment Operators.

Aim: To demonstrate assignment operators in Python.

Algorithm:

1. Start the program.
2. Assign an initial value to a variable.
3. Use assignment operators (+=, -=, *=, /=).
4. Display updated values.
5. Stop the program.

Program:

```
a = 10
a += 5
print(a)
a -= 3
print(a)
a *= 2
print(a)
a /= 4
print(a)
print("Aman Patel, 24SCSE1260003")
```

Output:

```
>>> 15 12 24 6.0
    Aman Patel, 24SCSE1260003
```

Result: Thus, the Python programs using arithmetic, relational, logical, and assignment operators were successfully executed and the outputs were verified.

EXPERIMENT 3

Aim: To study and implement different types of conditional statements in Python.

Programmes:

3.1 Program using simple if statement.

Aim: To write a Python program using a simple if statement.

Algorithm:

1. Start the program.
2. Take an integer input from the user.
3. Check if the number is greater than 0.
4. If the condition is true, print “Number is positive”.
5. Stop the program.

Program:

Practical 3 - Conditional Statements > 10_Simple_If.py > ...

```
num = int(input("Enter a number: "))

if num > 0:
    print("Number is positive")

print(
    "Name : Aman Patel\n"
    "Admission Number : 24SCSE1260003\n"
    "Batch : P2"
)
```

Output:

```
Enter a number: 123
Number is positive
Name : Aman Patel
Admission Number : 24SCSE1260003
Batch : P2
PS C:\Users\User\Desktop\SCARCE\PYTHON PROGRAMMING>
```

Conclusion:

Thus, the simple if statement was successfully implemented to execute a statement based on a given condition.

3.2 Program using if–else statement.

Aim: To write a Python program using if–else statement.

Algorithm:

1. Start the program.
2. Take an integer input from the user.
3. Check whether the number is even or odd.
4. If the number is divisible by 2, print “Even”.
5. Otherwise, print “Odd”.
6. Stop the program.

Program:

Practical 3 - Conditional Statements > 11_If_Else.py > ...

```
1   num = int(input("Enter a number: "))
2
3   if num % 2 == 0:
4       print("Even number")
5   else:
6       print("Odd number")
7
8   print(
9       "Name : Aman Patel\n"
10      "Admission Number : 24SCSE1260003\n"
11      "Batch : P2"
12  )
13
```

Output:

```
Enter a number: 324
Even number
Name : Aman Patel
Admission Number : 24SCSE1260003
Batch : P2
PS C:\Users\User\Desktop\SCARCE\PYTHON PROGRAMMING>
```

Conclusion:

Thus, the if–else statement was successfully implemented to execute one of the two blocks depending on the given condition.

3.3 Program using nested if–else.

Aim: To write a Python program using nested if–else statements.

Algorithm:

1. Start the program.
2. Take an integer input from the user.
3. Check whether the number is greater than or equal to 0.
4. If the number is greater than or equal to 0.
5. Else if the number is negative.
6. Stop the program.

Program:

Practical 3 - Conditional Statements > 12_Nested_If_Else.py > ...

```
num = int(input("Enter a number: "))

if num >= 0:
    if num == 0:
        print("Number is zero")
    else:
        print("Number is positive")
else:
    print("Number is negative")

print(
    "Name : Aman Patel\n"
    "Admission Number : 24SCSE1260003\n"
    "Batch : P2"
)
```

Output:

```
Enter a number: 234
Number is positive
Name : Aman Patel
Admission Number : 24SCSE1260003
Batch : P2
PS C:\Users\User\Desktop\SCARCE\PYTHON PROGRAMMING>
```

Conclusion:

Thus, the nested if–else statement was successfully implemented to check multiple conditions and produce the correct output.

3.4 Program using elif ladder.

Aim: To write a Python program using elif ladder.

Algorithm:

1. Start the program.
2. Take marks as input from the user.
3. Check marks range using if–elif ladder.
4. Print grade according to marks.
5. Stop the program.

Program:

Practical 3 - Conditional Statements > 13_Elif_Ladder.py > ...

```python
marks = int(input("Enter marks: "))

if marks >= 90:
    print("Grade A")
elif marks >= 75:
    print("Grade B")
elif marks >= 50:
    print("Grade C")
else:
    print("Fail")

print(
    "Name : Aman Patel\n"
    "Admission Number : 24SCSE1260003\n"
    "Batch : P2"
)

```

Output:

```
Enter marks: 98
Grade A
Name : Aman Patel
Admission Number : 24SCSE1260003
Batch : P2
PS C:\Users\User\Desktop\SCARCE\PYTHON PROGRAMMING>
```

Result:

Thus, the programs based on conditional statements (simple if, if–else, nested if–else, and elif ladder) were successfully executed and the desired output was obtained.

EXPERIMENT 4

Aim: To study and implement different looping statements in Python.

Programmes:

4.1 Program using for loop.

Aim: To write a Python program using for loop.

Algorithm:

1. Start the program.
2. Use for loop to print numbers from 1 to 5.
3. Stop the program.

Program:

Practical 4 - Looping Statements > 14_For_Loop.py > ...

```
for i in range(1, 6):
    print(i)

print(
    "Name : Aman Patel\n"
    "Admission Number : 24SCSE1260003\n"
    "Batch : P2"
)
```

Output:

```
1
2
3
4
5
Name : Aman Patel
Admission Number : 24SCSE1260003
Batch : P2
PS C:\Users\User\Desktop\SCARCE\PYTHON PROGRAMMING>
```

Conclusion:

Thus, the for loop was successfully implemented to execute statements repeatedly for a fixed range of values.

4.2 Program using while loop.

Aim: To write a Python program using while loop.

Algorithm:

1. Start the program.
2. Initialize a variable with value 1.
3. Use while loop to print numbers up to 5.
4. Increment the variable in each iteration.
5. Stop the program.

Program:

Practical 4 - Looping Statements > 15_While_Loop.py > ...

```
i = 1

while i <= 5:
    print(i)
    i += 1

print(
    "Name : Aman Patel\n"
    "Admission Number : 24SCSE1260003\n"
    "Batch : P2"
)
```

Output:

```
1
2
3
4
5
Name : Aman Patel
Admission Number : 24SCSE1260003
Batch : P2
PS C:\Users\User\Desktop\SCARCE\PYTHON PROGRAMMING>
```

Conclusion:

Thus, the while loop was successfully implemented to execute statements based on the given condition.

4.3 Program using nested for loop.

Aim: To write a Python program using nested for loop.

Algorithm:

1. Start the program.
2. Use outer loop for rows.
3. Use inner loop for columns.
4. Print pattern.
5. Stop the program.

Program:

Practical 4 - Looping Statements > 16_Nested_For_Loop.py > ...

```python
for i in range(1, 4):
    for j in range(1, 4):
        print(i, j)

print(
    "Name : Aman Patel\n"
    "Admission Number : 24SCSE1260003\n"
    "Batch : P2"
)

```

Output:

```
1 1
1 2
1 3
2 1
2 2
2 3
3 1
3 2
3 3
Name : Aman Patel
Admission Number : 24SCSE1260003
Batch : P2
PS C:\Users\User\Desktop\SCARCE\PYTHON PROGRAMMING>
```

Conclusion:

Thus, the nested for loop was successfully implemented to execute multiple loops and generate the required pattern.

4.4 Program using break statement.

Aim: To write a Python program using break statement.

Algorithm:

1. Start the program.
2. Use loop from 1 to 10.
3. If number equals 5, break the loop.
4. Print numbers before break.
5. Stop the program.

Program:

Practical 4 - Looping Statements > 17_Break_Statement.py > ...

```python
for i in range(1, 10):
    if i == 5:
        break
    print(i)

print(
    "Name : Aman Patel\n"
    "Admission Number : 24SCSE1260003\n"
    "Batch : P2"
)
```

Output:

```
1
2
3
4
Name : Aman Patel
Admission Number : 24SCSE1260003
Batch : P2
PS C:\Users\User\Desktop\SCARCE\PYTHON PROGRAMMING>
```

Conclusion:

Thus, the break statement was successfully used to terminate the loop when the specified condition was satisfied.

4.5 Program using continue statement.

Aim: To write a Python program using continue statement.

Algorithm:

1. Start the program.
2. Use loop from 1 to 5.
3. If number equals 3, skip that iteration.
4. Print remaining numbers.
5. Stop the program.

Program:

Practical 4 - Looping Statements > 18_Continue_Statement.py > ...

```
for i in range(1, 6):
    if i == 3:
        continue
    print(i)

print(
    "Name : Aman Patel\n"
    "Admission Number : 24SCSE1260003\n"
    "Batch : P2"
)
```

Output:

```
1
2
4
5
Name : Aman Patel
Admission Number : 24SCSE1260003
Batch : P2
PS C:\Users\User\Desktop\SCARCE\PYTHON PROGRAMMING>
```

Conclusion:

Thus, the continue statement was successfully used to skip a particular iteration and continue the execution of the loop.

Result:

Thus, the programs based on looping statements were successfully executed and the desired output was obtained.