Don Bosco Institute of Technology

Department of Electronics and Telecommunication Engg.
SUB: Skill Lab (ECL404)

Experiment on Decision flow control statements.

Expt No:	**02**
Aim:	To make use of 'if and else' statement, loop statements and conditional statements.
Tool used:	Anaconda Navigator + Jupiter Notebook
Theory:	Answer in brief (not more than 6 to7 sentences.) Resource: https://www.w3schools.com/python/

1. List the use of 'if else statement' and explain its declaration in python?

A: The "if-else" statement is a fundamental control flow structure in programming languages, including Python. It allows you to make decisions in your code based on certain conditions.

Syntax: if condition:
    # code to be executed if the condition is True
else:
    # code to be executed if the condition is False

Explanation:

if condition: This is the initial condition. If it evaluates to True, the code inside the associated block is executed. If it's False, the code inside the else block (if present) is executed.

else condition: This is an optional block that is executed if the condition in the if statement is False. It provides an alternative set of instructions.

2. What are 'For loops? Explain with an example.

A: A "for loop" is a control flow statement in programming languages that allows you to iterate over a sequence (such as a list, tuple, string, or range) and execute a block of code for each item in that sequence. It's particularly useful when you want to perform a repetitive task a specific number of times or for each element in a collection.

```
for i in range(5):
    print(f"Number: {i}")
```

```
Number: 0
Number: 1
Number: 2
Number: 3
Number: 4
```

3. With an example, explain the concepts of ‘for loop’ and ‘while loop’.

For Loop:

Used for iterating over a sequence (list, tuple, string, etc.).
The number of iterations is determined by the length of the sequence.
The for loop automatically handles the iteration process.

```
numbers = [1, 2, 3, 4, 5]
sum_result = 0

for num in numbers:
    sum_result += num

print(f"The sum of numbers is: {sum_result}")
```

```
The sum of numbers is: 15
```

While Loop:

Used when the number of iterations is not known beforehand and depends on a condition.
The loop continues until the condition becomes false.
You need to manage the loop control variable manually, as shown in the example with the decrementing number.

```
number = 5
factorial_result = 1

while number > 0:
    factorial_result *= number
    number -= 1

print(f"The factorial of 5 is: {factorial_result}")
```

```
The factorial of 5 is: 120
```

Tasks and outputs:

1. Create a list of numbers from 0 to 10. Display only the even numbers using a for loop.

```
numbers = list(range(11))
print(numbers)
print("Even numbers:")
for num in numbers:
    if num % 2 == 0:
        print(num)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Even numbers:
0
2
4
6
8
10
```

2. A) Print all numbers that are less than 10 using a while loop.

```
number = 0
print("Numbers less than 10:")
while number < 10:
    print(number)
    number += 1
```

```
Numbers less than 10:
0
1
2
3
4
5
6
7
8
9
```

2. B) Use ‘break’ statement in the earlier example to stop the loop at number 7.

```
number = 0
print("Numbers less than 10 (with break at 7):")
while number < 10:
    if number == 8:
        break
    print(number)
    number += 1
```

```
Numbers less than 10 (with break at 7):
0
1
2
3
4
5
6
7
```

2. C) Use continue statement to skip the number 4.

```
number = 0
print("Numbers less than 10 (skip 4 with continue):")
while number < 10:
    if number == 4:
        number += 1
        continue
    print(number)
    number += 1
```

```
Numbers less than 10 (skip 4 with continue):
0
1
2
3
5
6
7
8
9
```

3. Use ‘for loop’, split() and if statement to print out only the words that start with “k” in the given sentence.

```
sentence = "Keep kindness karma and knowledge in your life."
words = sentence.split()
print("Words starting with 'k':")
for word in words:
    if word.startswith('k') or word.startswith('K'):
        print(word)
```

```
Words starting with 'k':
Keep
kindness
karma
knowledge
```

4. Use range to print out odd numbers between 0 and 25.

```
print("Odd numbers between 0 and 25:")
for number in range(26):
    if number % 2 != 0:
        print(number)
```

```
Odd numbers between 0 and 25:
1
3
5
7
9
11
13
15
17
19
21
23
25
```

5. i. Use a comprehensive list to create a list of all numbers between 1 and 90 which are divisible by 3.

```
divisible_by_3 = [num for num in range(1, 91) if num % 3 == 0]
print(divisible_by_3)
```

```
[3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 42, 45, 48, 51, 54, 57, 60, 63, 66, 69, 72, 75, 78, 81, 84, 87, 90]
```

ii. Use a list comprehension to create a list of the first letters of every word in the given string.

```
sentence = "List comprehensions are concise and powerful."
first_letters = [word[0] for word in sentence.split()]
print(first_letters)
```

```
['L', 'c', 'a', 'c', 'a', 'p']
```

iii. Use a list comprehension to convert temperatures in °C to Fahrenheit.

```
temperatures_celsius = [0, 10, 20, 30, 40]
temperatures_fahrenheit = [(9/5) * temp + 32 for temp in temperatures_celsius]
print(temperatures_fahrenheit)
```

```
[32.0, 50.0, 68.0, 86.0, 104.0]
```

6. Use enumerate() to perform an index count on a given word. Display the result.

```
word = "Python"
index_count_result = list(enumerate(word))
print("Index count for each character:")
for index, char in index_count_result:
    print(f"Index {index}: {char}")
```

```
Index count for each character:
Index 0: P
Index 1: y
Index 2: t
Index 3: h
Index 4: o
Index 5: n
```

7. Write python statements to print integers from 1 to 20. But for multiples of 2, print “Skill” instead of the number and for multiples of 3, print ‘Lab’. For multiples of 2 and 3 print “Skill Lab’.

```
for num in range(1, 21):
    if num % 2 == 0 and num % 3 == 0:
        print("Skill Lab")
    elif num % 2 == 0:
        print("Skill")
    elif num % 3 == 0:
        print("Lab")
    else:
        print(num)
```

```
1
Skill
Lab
Skill
5
Skill Lab
7
Skill
Lab
Skill
11
Skill Lab
13
Skill
Lab
Skill
17
Skill Lab
19
Skill
```

8. For a certain string, print the words if the length of the word is even.

```
input_string = "Python is an amazing programming language."
words = input_string.split()
print("Words with even length:")
for word in words:
    if len(word) % 2 == 0:
        print(word)
```

```
Words with even length:
Python
is
an
```

9. Declare 2 numbers a and b. Compare both and display whether a is equal to b, a is greater than b or vice versa. Use If/ elif/ else statements.

```
a = 10
b = 7
if a == b:
    print("a is equal to b")
elif a > b:
    print("a is greater than b")
else:
    print("a is less than b")
```

```
a is greater than b
```

10. Zip two or three lists in python and display the result as a tuple.

```
list1 = [1, 2, 3, 4]
list2 = ['a', 'b', 'c', 'd']
result = list(zip(list1, list2))
print("Zipped result with two lists:")
print(result)
```

```
Zipped result with two lists:
[(1, 'a'), (2, 'b'), (3, 'c'), (4, 'd')]
```

Observation and Conclusion:	Write about those tasks and your algorithm used. In conclusion, the use of 'if and else' statements, loop statements, and conditional statements in Python experiments provides a powerful and flexible way to control the flow of a program and make decisions based on certain conditions. These constructs allow programmers to create dynamic and responsive code that can adapt to various scenarios. 1. The 'if and else' statements are fundamental for implementing branching logic, allowing the program to execute different blocks of code based on whether a certain condition is true or false. This is crucial for making decisions within the code and executing specific actions accordingly. 2. Loop statements, such as 'for' and 'while' loops, provide a mechanism for repeating a certain block of code multiple times. They are essential for iterating through data structures, performing repetitive tasks, and creating more efficient and concise code. 3. Conditional statements, including 'elif' (else if), further enhance the decision-making process by allowing the program to check multiple conditions in a sequential manner. This is particularly useful when dealing with complex scenarios that require multiple branching paths. Together, these constructs empower developers to write more dynamic, efficient, and readable code. They contribute to the overall structure and logic of a program, making it easier to understand, maintain, and modify. Additionally, the combination of 'if and else' statements, loop statements, and conditional statements enables the creation of sophisticated algorithms and solutions to various programming challenges.