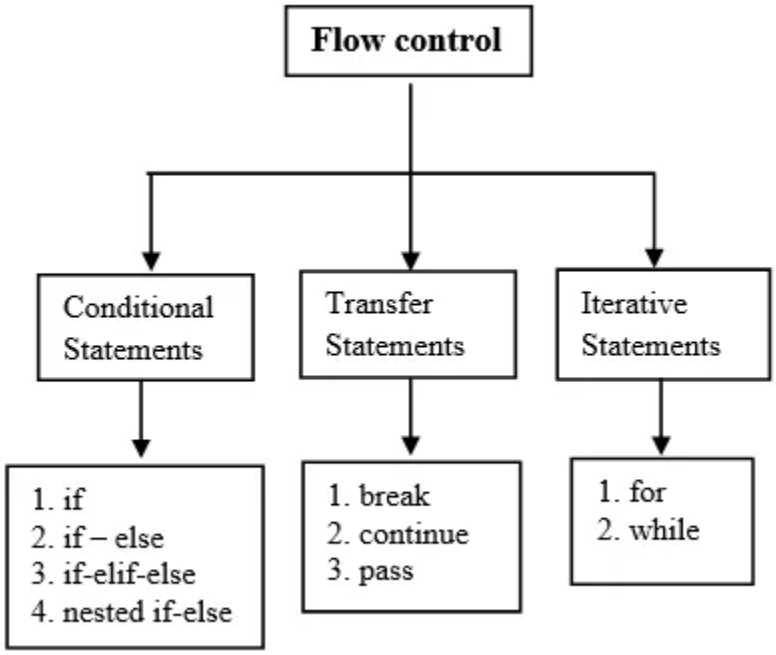
Marwadi University Marwadi Chandarana Group NAAC A+	**Marwadi University** **Faculty of Engineering & Technology** **Department of Information and Communication Technology**	
Subject: Programming With Python (01CT1309)	**Aim:** Develop programs to understand the control structures of python.	
Experiment No: 06	**Date:**	**Enrollment No: 92510133028**

Aim: Develop programs to understand the control structures of python.

IDE:

What is a control structure in Python?

A control structure in Python is a block of code that determines the flow of execution of a program. Control structures enable programmers to create programs that can make decisions based on certain conditions, repeat code blocks multiple times, or execute different code paths based on the values of variables.



There are several types of control structures in Python:

1. Conditional statements: These structures allow the program to execute different code blocks based on the value of a condition. The if statement is the most common conditional statement in Python, and it can be accompanied by elif and else statements.

2. Loops: These structures allow the program to execute a code block repeatedly based on a condition. Python has two main types of loops: while loops and for loops.

3. Exception handling: These structures allow the program to handle errors and exceptions in a controlled manner, preventing the program from crashing. Python has a try-except structure for handling exceptions.

4. Function and method definitions: These structures allow the programmer to define reusable blocks of code that can be called from other parts of the program. Functions and methods can take arguments and return values, and they can also contain other control structures.

By using control structures in Python, programmers can create more complex and powerful programs that can make decisions, repeat actions, and handle errors in a controlled way.

1. Conditional Statements (if, else, elif)

Conditional statements are fundamental to programming and allow us to make decisions based on specific conditions. In Python, we use the keywords if, else, and elif to implement conditional branching. The syntax is clean and straightforward, making Python code highly readable.

1. if Statement

The if statement is used to execute a block of code if a given condition is True. If the condition is False, the code inside the if block is skipped.

Example:

```
x = 10
if x>5:
   print(“x is greter than 5”)
```

Output

```
PS E:\PWP> & C:/Users/keshv/AppData/Local/Programs/Python/Python313/python.exe e:/PWP/harikeshsirexperiment/exp6.py
x is greater than 5
PS E:\PWP>
```

2. elif Statement

The elif statement is used when you have multiple conditions to check. It comes after an if statement and before an optional else statement. If the initial if condition is False, Python evaluates the elif condition. You can have multiple elif conditions.

Marwadi University Marwadi Chandarana Group NAAC A+	**Marwadi University** **Faculty of Engineering & Technology** **Department of Information and Communication Technology**	
Subject: Programming With Python (01CT1309)	**Aim:** Develop programs to understand the control structures of python.	
Experiment No: 06	**Date:**	**Enrollment No: 92510133028**

Example:

```
x = 10
if  x>5
    print("x is greater than 5")
elif x ==5:
   print("x is equal to 5")
else:
   print("x is less than 5")
```

Output

```
PS E:\PWP> & C:/Users/keshv/AppData/Local/Programs/Python/Python313/python.exe e:/PWP/harikeshsirexperiment/exp6.py
x is greater than 5
PS E:\PWP>
```

3. else Statement

The else statement is used to execute a block of code when the conditions specified in the if and elif statements are not met.

Example

```
if x>5:
   print("x is greater than 5")
else:
   print("x is not greater than 5")
```

Output

```
PS E:\PWP> & C:/Users/keshv/AppData/Local/Programs/Python/Python313/python.exe e:/PWP/harikeshsirexperiment/exp6.py
x is greater than 5
PS E:\PWP>
```

Nested if-else statements

Nested if-else statements in Python allow you to test multiple conditions and execute different code blocks based on the results of these tests.

Marwadi University Marwadi Chandarana Group NAAC A+	**Marwadi University** **Faculty of Engineering & Technology** **Department of Information and Communication Technology**	
Subject: Programming With Python (01CT1309)	**Aim:** Develop programs to understand the control structures of python.	
Experiment No: 06	**Date:**	**Enrollment No: 92510133028**

Example

```
age = 35

if age >= 60:
   print("You are a senior citizen.")
else:
   if age >= 18:
      print("You are an adult.")
   else:
      print("You are a teenager.")
```

Output

```
PS E:\PWP> & C:/Users/keshv/AppData/Local/Programs/Python/Python313/python.exe e:/PWP/harikeshsirexperiment/exp6.py
You are an adult.
PS E:\PWP>
```

Example

```
num = 10

if num > 0:
   if num % 2 == 0:
      print("The number is positive and even.")
   else:
      print("The number is positive but odd.")
else:
   print("The number is not positive.")
```

Output

```
PS E:\PWP> & C:/Users/keshv/AppData/Local/Programs/Python/Python313/python.exe e:/PWP/harikeshsirexperiment/exp6.py
The number is positive and even.
PS E:\PWP>
```

Looping Statements

1. for Loop

The for loop is used to iterate over sequences like lists, tuples, strings, and dictionaries. It allows you to perform an action for each item in the sequence.

Marwadi University Marwadi Chandarana Group NAAC A+	**Marwadi University** **Faculty of Engineering & Technology** **Department of Information and Communication Technology**	
Subject: Programming With Python (01CT1309)	**Aim:** Develop programs to understand the control structures of python.	
Experiment No: 06	**Date:**	**Enrollment No: 92510133028**

Example

```
Fruits = ["apple", "banana", "cherry"]
for fruit in fruits:
    print(fruit)
```

Output

```
PS E:\PWP> & C:/Users/keshv/AppData/Local/Programs/Python/Python313/python.exe e:/PWP/harikeshsirexperiment/exp6.py
apple
banana
cherry
```

2. while Loop

The while loop is used to repeatedly execute a block of code as long as a specified condition is True.

Example

```
x = 1
while x<=5:
    print(x)
    x+=1
```

```
PS E:\PWP> & C:/Users/keshv/AppData/Local/Programs/Python/Python313/python.exe e:/PWP/harikeshsirexperiment/exp6.py
1
2
3
4
5
```

Loop Control Statements

Python provides several loop control statements to enhance the functionality of loops.

break: The break statement is used inside a loop (while loop or for loop). When the condition specified in the if statement is true, the break statement is executed, and the control is transferred to the next statement after the loop. This means that the loop is terminated, and the code execution continues from the statement after the loop.

Example

```
for x in range(1,6):
        if x==3:
        break
        print(x)
```

Output

```
PS E:\PWP> & C:/Users/keshv/AppData/Local/Programs/Python/Python313/python.exe e:/PWP/harikeshsirexperiment/exp6.py
3
PS E:\PWP>
```

continue: The continue statement is used to skip a particular iteration of a loop when a specific condition is met. When a continue statement is executed inside a loop, it skips the current iteration of the loop and jumps to the next iteration.

Example

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

Output

```
PS E:\PWP> & C:/Users/keshv/AppData/Local/Programs/Python/Python313/python.exe e:/PWP/harikeshsirexperiment/exp6.py
1
2
4
5
PS E:\PWP>
```

pass: The pass statement is a placeholder in Python. It doesn't do anything but is used when a statement is syntactically required. It is often used as a placeholder for functions, loops, or conditional blocks that will be implemented later.

Example

```
for x in range(1,6):
   if x == 3:
      pass
   print(x)
```

Marwadi University Marwadi Chandarana Group NAAC A+	**Marwadi University** **Faculty of Engineering & Technology** **Department of Information and Communication Technology**	
Subject: Programming With Python (01CT1309)	**Aim:** Develop programs to understand the control structures of python.	
Experiment No: 06	**Date:**	**Enrollment No: 92510133028**

Output

```
PS E:\PWP> & C:/Users/keshv/AppData/Local/Programs/Python/Python313/python.exe e:/PWP/harikeshsirexperiment/exp6.py
1
2
3
4
5
```

Try and Except Statement – Catching Exceptions

- In Python, you may catch and deal with exceptions by using the try and except commands.
- The try and except clauses are used to contain statements that can raise exceptions and statements that handle such exceptions.

Example

```
try:
  number = int(input("Enter a number: "))
  result = 10 / number
  print("The result is:", result)
except ZeroDivisionError:
  print("Division by zero is not allowed.")
except ValueError:
  print("Invalid input. Please enter a valid number.")
```

Python Function:

Python functions are reusable code blocks that carry out particular tasks, helping programmers structure their code and make it easier to read. By preventing duplication, functions make the code more modular and manageable. The 'def' keyword, the function name, and any parameters included in parenthesis define a function. The code to be performed is contained in the function body, and the 'return' statement allows the function to produce a result.

Types of Functions in Python

Python supports various types of functions, each serving different purposes in programming. Here are the main types of functions in Python, along with examples:

1. Built-in Functions

These functions are pre-defined in Python and can be used directly without any further declaration.
Example

```
my_list = [1, 2, 3, 4, 5]
print(len(my_list))
```

2. User-defined Functions
These are functions that users create to perform specific tasks.

Example

```
def add_numbers(a, b):
    return a + b
result = add_numbers(3, 5)
print(result)
```

```
PS E:\PWP> & C:/Users/keshv/AppData/Local/Programs/Python/Python313/python.exe e:/PWP/harikeshsirexperiment/exp6.py
8
PS E:\PWP>
```

3. Anonymous Functions (Lambda Functions)
These are small, unnamed functions defined using the lambda keyword. They are typically used for short, simple operations.
Example

```
add = lambda x, y: x + y
print(add(3, 5))
```

Output

```
PS E:\PWP> & C:/Users/keshv/AppData/Local/Programs/Python/Python313/python.exe e:/PWP/harikeshsirexperiment/exp6.py
8
PS E:\PWP>
```

4. Recursive Functions
These are functions that call themselves within their definition. They help solve problems that can be broken down into smaller, similar problems.

Example

```
def factorial(n):

    if n == 1:
        return 1
    else:
        return n * factorial(n - 1)
print(factorial(5))
```

Output

```
PS E:\PWP> & C:/Users/keshv/AppData/Local/Programs/Python/Python313/python.exe e:/PWP/harikeshsirexperiment/exp6.py
120
PS E:\PWP>
```

5. Higher-Order Functions

These functions can take other functions as arguments or return them as results. Examples include map(), filter(), and reduce().

Example

```
def square(x):
    return x * x
numbers = [1, 2, 3, 4, 5]
squared_numbers = list(map(square, numbers))
print(squared_numbers)
```

```
PS E:\PWP> & C:/Users/keshv/AppData/Local/Programs/Python/Python313/python.exe e:/PWP/harikeshsirexperiment/exp6.py
[1, 4, 9, 16, 25]
PS E:\PWP>
```

6. Generator Functions

These functions yield values one at a time and can produce a sequence of values over time, using the yield keyword.

Example

```
def generate_numbers():
    for i in range(1, 6):
        yield i
for number in generate_numbers():
```

print(number)

```
PS E:\PWP> & C:/Users/keshv/AppData/Local/Programs/Python/Python313/python.exe e:/PWP/harikeshsirexperiment/exp6.py
1
2
3
4
5
PS E:\PWP>
```

Post Lab Exercise:

a. Write a Python program to print all odd numbers between 1 to 100 using a while loop.

```
# Initialize the starting number
num = 1

while num <= 100:
    if num % 2 != 0:  # Check if the number is odd
        print(num)
    num += 1  # Increment the number
```

```
> & C:/Users/keshv/AppData/Local/Programs/Python/Python313/python.exe e:/PWP/harikeshsirexperiment/exp6.py
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
35
37
39
41
43
45
47
49
51
53
55
57
59
61
63
65
67
69
71
```

Marwadi University Marwadi Chandarana Group NAAC A+	**Marwadi University** **Faculty of Engineering & Technology** **Department of Information and Communication Technology**	
Subject: Programming With Python (01CT1309)	**Aim:** Develop programs to understand the control structures of python.	
Experiment No: 06	**Date:**	**Enrollment No: 92510133028**

```
73
75
77
79
81
83
85
87
89
91
93
95
97
99
```

b. Write a Python program to find the sum of all natural numbers between 1 to n.

```
# Take input from user
n = int(input("Enter a positive integer: "))

# Initialize sum
total = 0
num = 1

# Loop to add numbers
while num <= n:
    total += num
    num += 1

# Display result
print(f"The sum of natural numbers from 1 to {n} is: {total}")
```

```
        > & C:/Users/keshv/AppData/Local/Programs/Python/Python313/python.exe e:/PWP/harikeshsirexperiment/exp6.py
Enter a positive integer: -35
The sum of natural numbers from 1 to -35 is: 0
PS E:\PWP>
```

Marwadi University Marwadi Chandarana Group NAAC A+	**Marwadi University** **Faculty of Engineering & Technology** **Department of Information and Communication Technology**	
Subject: Programming With Python (01CT1309)	**Aim:** Develop programs to understand the control structures of python.	
Experiment No: 06	**Date:**	**Enrollment No: 92510133028**

c. Write a Python function program to count a number of digits in a number.

```
def count_digits(num):
    # Convert negative number to positive
    num = abs(num)

    # Special case for 0
    if num == 0:
        return 1

    count = 0
    while num > 0:
        num //= 10  # Remove the last digit
        count += 1
    return count

# Example usage
number = int(input("Enter a number: "))
print(f"The number of digits in {number} is: {count_digits(number)}")
```

```
PS E:\PWP> & C:/Users/keshv/AppData/Local/Programs/Python/Python313/python.exe e:/PWP/harikeshsirexperiment/exp6.py
Enter a number: 21
The number of digits in 21 is: 2
PS E:\PWP>
```

d. Write a Python program to find the first and last digits of a number.

```
# Take input from the user
num = int(input("Enter a number: "))

# Get last digit
last_digit = abs(num) % 10

# Get first digit
first_digit = abs(num)
while first_digit >= 10:
    first_digit //= 10
```

```
print(f"First digit: {first_digit}")
print(f"Last digit: {last_digit}")
```

```
PS E:\PWP> & C:/Users/keshv/AppData/Local/Programs/Python/Python313/python.exe e:/PWP/harikeshsirexperiment/exp6.py
Enter a number: 444
First digit: 4
Last digit: 4
```

e. Write a Python program to swap the first and last digits of a number.

```
# Take input from user
num = int(input("Enter a number: "))

# Convert to string for easy swapping
num_str = str(num)

# If the number has only one digit, no swap needed
if len(num_str) == 1:
    swapped_num = num_str
else:
    swapped_num = num_str[-1] + num_str[1:-1] + num_str[0]

# Convert back to integer
swapped_num = int(swapped_num)

print(f"Number after swapping first and last digits: {swapped_num}")
```

```
PS E:\PWP> & C:/Users/keshv/AppData/Local/Programs/Python/Python313/python.exe e:/PWP/harikeshsirexperiment/exp6.py
Enter a number: 668
Number after swapping first and last digits: 866
```

f. Write a Python program to calculate the product of digits of a number.

```
# Take input from user
num = int(input("Enter a number: "))

# Work with absolute value to handle negatives
num = abs(num)

product = 1
```

Marwadi University Marwadi Chandarana Group NAAC A+	**Marwadi University** **Faculty of Engineering & Technology** **Department of Information and Communication Technology**	
Subject: Programming With Python (01CT1309)	**Aim:** Develop programs to understand the control structures of python.	
Experiment No: 06	**Date:**	**Enrollment No: 92510133028**

```
# Special case: if the number is 0
if num == 0:
    product = 0
else:
    while num > 0:
        digit = num % 10      # Get last digit
        product *= digit      # Multiply to product
        num //= 10            # Remove last digit

print(f"Product of digits: {product}")
```

```
        > & C:/Users/keshv/AppData/Local/Programs/Python/Python313/python.exe e:/PWP/harikeshsirexperiment/exp6.py
Enter a number: 458
Product of digits: 160
```

g. Write a Python program to enter a number and print its reverse

```
# Take input from the user
num = int(input("Enter a number: "))

# Work with absolute value for reversing
is_negative = num < 0
num = abs(num)

reverse_num = 0

while num > 0:
    digit = num % 10          # Get the last digit
    reverse_num = reverse_num * 10 + digit
    num //= 10                # Remove the last digit

# Restore sign if number was negative
```

Marwadi University Marwadi Chandarana Group NAAC A+	**Marwadi University** **Faculty of Engineering & Technology** **Department of Information and Communication Technology**	
Subject: Programming With Python (01CT1309)	**Aim:** Develop programs to understand the control structures of python.	
Experiment No: 06	**Date:**	**Enrollment No: 92510133028**

```
if is_negative:
    reverse_num = -reverse_num

print(f"Reversed number: {reverse_num}")
```

```
PS E:\PWP> & C:/Users/keshv/AppData/Local/Programs/Python/Python313/python.exe e:/PWP/harikeshsirexperiment/exp6.py
Enter a number: 682
Reversed number: 286
```