UNIT 4

Control Structures in Python

Student Learning Outcomes

By the end of this chapter, you will be able to:

- Implement control structures such as decision-making statements and loops in Python.
- Work with Python modules, functions, and built-in data structures like lists.
- Perform different operations on lists.

Introduction

In programming, we often need to control the flow of our program based on different conditions or repeat certain actions multiple times. This is where control structures come into play. Control structures are essential because they help us manage the decision-making process and the repetition of tasks in our programs. There are two main types of control structures, Decision making and Looping:

4.1 Decision Making

Decision making in programming allows the program to choose different actions based on conditions. This is similar to how we make decisions in real life. For example, deciding whether to take an umbrella based on the weather situation. Python provide variety of conditional statements to implement decision making.

if Statement

The **if** statement lets us make decisions based on conditions. If the condition is true, it runs a block of code. In Python indentation defines the structure block or scope of your code. In Python proper indentation is mandatory. Incorrect indentation may cause error in your code (Indentation Error).

```
# Syntax of if statement
if condition:
        code to run if the condition is true
```

Example: If the temperature is above 30 degrees(e.g. It is hot today), we print a message.

```
temperature = 35
if temperature > 30:
    print("It's a hot day")
# Output: It's a hot day
```

if - else Statement

The **if-else** statement allows us to execute one block of code if a condition is true and another block if the condition is false.

 # Syntax of if-else statement

```
if condition:
        code to run if the condition is true:
else:
        code to run if the condition is false
```

Example:

```
temperature = 15
if temperature > 30:
   print("It's a hot day ")
else :
   print("It's not a hot day " )
#Output: It's not a hot day
```

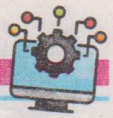
Nested Conditions

Sometimes, we need to check multiple conditions inside another condition. This is called **nesting**.

 # Syntax of nested if statement

```
if condition1:
    if condition 2:
        code to run if both condition1 and condition2 are true
    else:
        code to run if condition 1 is true but condition2 is false
else:
    code to run if condition 1 is false
```

Example: If the weather is rainy and the temperature is below 15 degrees, we wear a raincoat. If it is only rainy, we take an umbrella. If the weather is not rainy, we just enjoy the day.



```
weather = "rainy"
temperature = 10
if weather == "rainy":
        if temperature < 15:
                print("Wear a raincoat")
        else :
                print("Take an umbrella")
else:
        print("Enjoy your day!")
# Output: Wear a raincoat
```

Explanation: In this example, the code checks if the weather is rainy. If it is, it then checks if the temperature is below 15 degrees. If both conditions are true, it prints "Wear a raincoat". If the weather is rainy but the temperature is not below 15 degrees, it prints "Take an umbrella". If the weather is not rainy, it prints "Enjoy your day!".

4.2 Looping Constructs

Loops help us repeat actions, making our code more efficient and easier to read. There are two main types of loops in Python: **while** loops and **for** loops.

while Loop

A **while** loop runs as long as a condition is true. It checks the condition before each iteration and stops running when the condition is no longer true.

```
# Syntax of while loop
while condition:
        code to run while the condition is true
```

Example: Add 1 to a number until it reaches 10.

```
number = 1
while number < 10:
    print(number)
    number += 1
```

```
'''
Output:
1
2
3
4
5
6
7
8
9
'''
```

Explanation: In this example, the code starts with a **number** set to 1. The 'while' loop checks if the **number** is less than 10. If it is, the program prints the current value of the **number** and then adds 1 to it. This loop continues until the **number** reaches 10, at which point the loop stops running.

for Loop

A **for** loop repeats a block of code a specific number of times. It is commonly used to iterate over a sequence (like a list, tuple, or string).

```
# Syntax of for loop
for variable in sequence:
    code to run for each element in the sequence
```

Example: Say "Welcome" to each friend in a list of friends.

```
friends = ["Sami", "Raza", "Moosa"]
for friend in friends:
    print("Welcome to ", friend)
```

```
'''
Output:
Welcome to: Sami
Welcome to: Raza
Welcome to: Moosa
'''
```

Explanation: In this example, the code goes through each friend in the list and prints a greeting message for each one.

range() Function

We can use the **range()** function to generate a sequence of numbers, which is often used in **for** loops.

```
# Syntax of range function
range(stop)
range(start, stop)
range(start, stop, step)
```

Example: Print First 5 Whole Numbers.

```
for i in range (5):
    print (i)
```

```
'''
Output:
0
1
2
3
4
'''
```

end statement

In Python, the end statement is not a separate command, but rather an optional parameter of the print() function. By default, every print() in Python ends with a newline character, but you can change what gets printed at the end using the end parameter.

Example 1: Print two Strings on same line using "end" statement.

```
print("Azhar Shah", end=" ")
print("Ahsan")
# Output:  Azhar Shah Ahsan
print("Ayyan", end="***")
print("Naveed")
# Output: Ayyan***Naveed
```

DO YOU KNOW? Strings in Python are surrounded by either Single quotation' ' marks or double quotation " " marks. Like 'Ayyan' is the same as "Ayyan".

Example 2: Print Numbers from 1 to 5 on the same line using "end" statement.

```
for i in range(1, 6):
  # if you want to print Numbers on same line use following
line
      print(i, end=" ")
  # Output: 1 2 3 4 5
```

Explanation: The above code generates numbers from 0 to 5 and prints each number.

ACTIVITY

1. Write a **for** loop using **range()** to print the even numbers from 2 to 10 on single line.
2. Write a Python program that prints the first 10 multiples of 3 using a **for** loop and the **range()** function.

Nested Loops

Nested loops are loops inside other loop. For each iteration of the outer loop, the inner loop runs completely. We use nested loops when we need to perform repetitive tasks within tasks.

Example1: Nested loops for basic Number Pattern

```
print("=== Basic Number Pattern ===")
for i in range(3):          # Outer loop
    for j in range(4):      # Inner loop
        print(f"i={i}, j={j}")
    print("---")           # Separator after each outer loop
iteration
'''
# Output
=== Basic Number Pattern ===
i=0, j=0
i=0, j=1
i=0, j=2
i=0, j=3
---
i=1, j=0
i=1, j=1
i=1, j=2
i=1, j=3
---
i=2, j=0
i=2, j=1
i=2, j=2
i=2, j=3
--- '''
```

Example2: Nested loops to print stars (asterisk)in Triangular Pattern

```
print("=== Right-Angled Triangle ===")
n = 5
for i in range(1, n+1):
    for j in range(i):          # Inner loop runs 'i' times
        print("*", end=" ")
    print()  # New line after each row
'''
# Output
*
* *
* * *
* * * *
* * * * *
'''
```

4.3 Libraries in Python

Python offers an extensive standard library that includes built-in modules and data structures. A data structure refers to a particular format or method for organizing and storing data. In this section, we will examine the utilization of different libraries in Python.

Using Libraries in Python

In Python, libraries are like toolboxes full of useful tools that help you solve different problems without having to build everything from scratch. Libraries are like pre-built toolkits that you can use without having to write all the code yourself. Let's explore how to import and use different libraries with some simple examples.

Example: Import the random library to generate random numbers.

```
import random
# Generate a random number between 1 to 10
number = random.randint(1, 10)
print("The random number is:", number)
# Output:
# The random number is: 3
```

Example: Import the statistics library to perform statistical calculations.

```
import statistics
# Calculate the mean of a list of numbers
data = [23, 45, 67, 89, 12, 44, 56]
mean_value = statistics.mean(data)
print("The mean value is:", mean_value)
# Output:
# The mean value is: 48
```

Explanation: The statistics library is a great tool for performing basic statistical calculations, such as finding the mean, median, and mode of a set of data. This can be particularly useful in data analysis tasks.

In short by using these libraries, you can save time and effort, allowing you to focus on solving the specific problem at hand rather than reinventing the wheel.

TOOL TIP

When importing libraries, make sure to only import what you need. You can explore the more libraries at https://docs.python.org/3/

4.4 Lists in Python

In Python, a list is a versatile data structure that can hold a collection of items. You can create, access, and modify lists easily.

Creating Lists

A list is created by placing items inside square brackets [ ], separated by commas. Lists can contain items of different types, such as numbers, strings, or even other lists.

Example: Create a list of your favorite fruits.

```
fruits = ["Mango", "Apple","Banana"]
print(fruits)
# Output: ['Mango' 'Apple ' 'Banana ']
```

Explanation: This code creates a list named 'fruits' containing three elements and then prints the list.

Accessing List Items

You can access items in a list by referring to their index, starting from 0.

Example: Access and print the second item from the list of fruits.

```
fruits = ["Mango", "Apple",  ,"Banana"]
print(fruits [1])
# Output: Apple
```

Explanation: The code initializes a list 'fruits' containing 'Mango', 'Apple', and 'Banana', then prints the second item, 'Apple', using the index ' 1 '.

ACTIVITY

Write a Python program that imports the random and statistics libraries. Use random to generate a list of 10 random numbers between 1 and 50. Then, use statistics to calculate and print the mean (average) of those numbers.

Modifying a List

You can modify list items by accessing them via their index and assigning a new value.

Example: Change the first item in the list to "Orange" and add a new fruit "Pineapple".

```
fruits = ["Mango", "Apple", "Banana"]
fruits[0] = "Orange"
fruits.append("Pineapple")
print(fruits)
# Output: ['Orange', 'Apple', 'Banana', 'Pineapple']
```

Explanation: The code modifies the first element of the 'fruits' list to 'Orange', appends 'Pineapple' at the end, and prints the updated list.

Operations on Lists

Python provides several built-in methods to work with lists. Here are a few useful ones:

- append (item) - Adds an item to the end of the list.
- remove (item) - Removes the first occurrence of an item from the list.
- sort () - Sorts the list in ascending order.
- reverse () - Reverses the order of the list.

Example 1: Add a new student to the list of students and print the list.

```
students = ["Ahmed", "Sara", "Ali"]
students.append("Hina")
print(students)
#Output : ['Ahmed', 'Ali', 'Sara', 'Hina']
```

Explanation: The code creates a list of students, adds 'Hina' to it, sorts the list alphabetically.

Example 2: Remove a student from the list of students and print the list.

```
# Remove a student from the list of students
students = ["Ahmed", "Sara", "Ali", "Hina", "Sara"]
students.remove("Sara")
print(students)
# Output: ['Ahmed', 'Ali', 'Hina', 'Sara']
```

Explanation: The code creates a list of students, removes the first occurrence of "Sara" from the list, and prints the updated list. Note that only the first matching item is removed.

Example 3: Sort a list of numbers in ascending order.

```
# Sort a list of numbers in ascending order
numbers = [34, 12, 89, 5, 23]
numbers.sort()
print(numbers)
# Output: [5, 12, 23, 34, 89]
```

Explanation: The code creates a list of numbers, sorts them in ascending order using the sort() method, and prints the sorted list.

Example 4: Reverse the order of fruits in a list.

```
# Reverse the order of fruits in a list
fruits = ["Apple", "Banana", "Orange", "Mango"]
fruits.reverse()
print(fruits)
# Output: ['Mango', 'Orange', 'Banana', 'Apple']
```

Explanation: The code creates a list of fruits, reverses the order of elements using the reverse() method, and prints the reversed list.

4.5 Testing and Debugging in Python

In Python programming, testing and debugging are essential practices to ensure that your code works correctly and efficiently.

Testing

Testing is the process of running your code with various inputs to check if it behaves as expected. The goal is to find and fix any issues before the code is used in real-world applications.

Types of Testing

- **Unit Testing:** Tests individual parts of the code (like functions or classes) in isolation. Python's unit test module is commonly used for this.
- **Integration Testing:** Checks how different parts of the code work together.
- **Functional Testing:** Validates that the software behaves as expected from the user's perspective.
- **Regression Testing:** Ensures that new changes don't affect the existing functionality.

Debugging

Debugging is the process of finding and fixing errors (bugs) in your code. It involves identifying the root cause of problems and making the necessary changes.

Common Debugging Techniques

- **Print Statements:** Adding print statements to check the values of variables at different stages of the code.

- **Debugging Tools:** Using tools like pdb (Python Debugger) to step through the code, inspect variables, and understand the flow of execution.
- **Error Messages:** Reading and interpreting error messages to locate the source of the problem.

Summary

- Decision making in programming allows the program to choose different actions based on conditions.
- The if statement lets us make decisions based on conditions. If the condition is true, it runs a block of code.
- The if-else statement allows us to execute one block of code, if a condition is true and another block if the condition is false.
- Sometimes, we need to check multiple conditions inside another condition. This is called nesting.
- Loops help us repeat actions, making our code more efficient and easier to read.
- A while loop runs as long as a condition is true. It checks the condition before each iteration and stops running when the condition is no longer true.
- A for loop repeats a block of code a specific number of times. It is commonly used to iterate over a sequence.
- We can use the range() function to generate a sequence of numbers, which is often used in for loops.
- In Python, the end statement is not a separate command, but rather an optional parameter of the print() function.
- Nested loops are loops inside other loops. The outer loop runs first. For each iteration of the outer loop, the inner loop runs completely.
- In Python, a list is a versatile data structure that can hold a collection of items. You can create, access, and modify lists easily.
- Testing is the process of running your code with various inputs to check if it behaves as expected.
- Debugging is the process of finding and fixing errors (bugs) in your code. It involves identifying the root cause of problems and making the necessary changes.

EXERCISE

Multiple Choice Questions

1. **What does the range () function do in Python?**

   (a) Generates a list of numbers (b) Creates a sequence of numbers
   (c) Calculates the sum of numbers (d) Prints a range of numbers

2. **What is the output of the below code?**

```
count = 0
for i in range(2):
    for j in range(3):
     count += 1
print(count)
```

   (a) 4 (b) 5
   (c) 6 (d) 7

3. **Which method is used to add an item at the end of the list in Python?**

   (a) insert() (b) append()
   (c) add() (d) extend()

4. **What is the purpose of control structures in programming?**

   (a) Only to make code longer
   (b) To manage decision-making and repetition of tasks
   (c) To create variables
   (d) To import libraries

5. **Which statement is used to execute code when a condition is false?**

   (a) if (b) elif
   (c) else (d) while

6. **How do you prevent print() from adding a newline?**

   (a) Use end="" (b) Use newline=False
   (c) Use break (d) Use continue

7. Which library is used to generate random numbers?

(a) Datetime (b) statistics
(c) random (d) math

8. What does list.append() do?

(a) Removes the first item (b) Adds an item to the end
(c) Sorts the list (d) Reverses the list

9. The type of testing which checks how different parts of the code work together is:

(a) Unit Testing
(b) Integration Testing
(c) Functional Testing
(d) Regression Testing

10. Which debugging technique involves adding output statements?

(a) Unit testing (b) Print statements
(c) Error messages (d) Integration testing

Short Questions

1. Explain the use of the range() function in for loop?
2. What is the difference between the append and remove methods in Python lists?
3. Define debugging.
4. What are the three parameters of the range() function?
5. Name three list methods and their purposes.
6. How do you import a library in Python?
7. What is unit testing and why is it important?
8. How do you access the third element in a list?
9. What is the purpose of the end parameter in print()?
10. How can we assess an element from the list?
11. How do you modify an existing item in a list

Long Questions

1. Explain the different types of decision-making statements in Python with examples for each.
2. Compare while loops and for loops.

3. Describe how to work with Python lists, including creation and modification.
4. Explain the importance of testing and debugging in programming. Discuss different debugging techniques.
5. How do Python libraries enhance programming efficiency? Provide an example.
6. Write a Python program using a loop that prints all the odd numbers between 1 and 20.
7. Write a Python program that performs the following tasks:
   (a) Generates a list of 5 random numbers between 1 and 20.
   (b) Uses the statistics library to calculate and print the mean of the generated numbers.
8. Write a Python program that performs the following operations:
   (a) Create a list of popular Pakistani dishes: ["Biryani", "Nihari", "Karahi", "Seekh Kebabs"].
   (b) Add a new dish "Quorma" to the list.
   (c) Change "Karahi" to "Chicken Karahi".
   (d) Remove "Nihari" from the list.
9. Write a code to print stars (asterisk) in rectangular pattern using nested loops with three rows and five columns.

Answer Key for Multiple Choice Questions

1. **b)** - Creates a sequence of numbers
2. **c)** - 6
3. **b)** - append()
4. **b)** - To manage decision-making and repetition of tasks
5. **c)** - else
6. **a)** - Use end=""
7. **c)** - random
8. **b)** - Adds an item to the end
9. **b)** - To generate a sequence of numbers
10. **b)** - Print statements