COMPUTER SCIENCE — 10

UNIT 04

CONTROL STRUCTURES IN PYTHON

Topics Covered in this Unit
1. Decision Making
2. Nested Conditions
3. Looping Constructs
4. Libraries in Python
5. Lists in Python
6. Testing and Debugging in Python

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.

Decision Making

Q.1 Explain the different types of decision-making statements in Python with examples for each. *10504001*

Ans. ★ *Key Points: Decision Making Statements | if | if-else | Nested Conditions*

Decision-Making Structure

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 conditions. Python provides a variety of conditional statements to implement decision-making.

Types of Decision-making Structure

In Python, decision-making structures refer to control flow constructs that allow a program to make decisions based on conditions. These structures determine the execution path of the code. There are three main types of decision-making structures in Python:

1. 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. Proper indentation is mandatory. Incorrect indentation may cause an error (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*

2. 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*

3. Nested Conditions

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

Syntax of nested if statement:

```
if condition 1:
    if condition 2:
        code to run if both condition 1 and condition 2 are true
    else:
        code to run if condition 1 is true but condition 2 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!".

Looping Constructs

Q.2 What is a looping structure in programming? Explain its types with examples. *10504002*

Ans. ★ *Key Points: Looping Structure | while loop | for loop*

A looping structure in programming allows a set of instructions to be executed repeatedly until a specific condition is met. It helps the programmer to reduce the process of writing similar code again and again.

Types of Looping Structure

There are two main types of loops in Python:

- while loop
- for loop

1. 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 1: Print numbers from 1 to 9 OR (Add 1 to a number until it reaches 10)

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

Output (left): 1 2 3 4 5 6 7 8 9

Output (right): 1 2 3 4 5 6 7 8 9 10

Explanation:

- The program prints numbers from 1 to 9. It starts with number = 1 and repeatedly prints the value and adds 1 until number reaches 10, then the loop stops.
- In the second version, the while loop checks if the number is less than or equal to 10. If it is, the program prints the current value and then adds 1 to it. This continues until number exceeds 10, at which point the loop stops.

Example 2: Write a Python program that prints even numbers and counts the odd numbers from 1 to 20 using a while loop.

```
num=1
odd_count=0
while num<=20:
    if num%2==0:
        print(f"Even: {num}")
    else:
        odd_count +=1
    num +=1
print("Total odd numbers:",odd_count)
```

Output: *Even:2 Even:4 Even:6 Even:8 Even:10 Even:12 Even:14 Even:16 Even:18 Even:20 Total odd numbers:10*

2. 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
```

SMART TIPS

List: A list is an ordered collection of items that can be of different data types. It is always enclosed in square brackets. e.g. list[0] = 10

Tuple: A tuple is an ordered, immutable collection of items similar to a list but with a fixed size. It is enclosed in parenthesis (). e.g. (1,2,3)

String: A string is a sequence of characters used to represent text data. Always enclosed in single or double quotes. e.g. string = "Faisal_Zia"

Example 1: Say "Hello" to each friend in a list of friends.

```
friends=["Arshad","Ayaan","Areeda"]
for friend in friends:
    print(" Hello",friend)
```

Output: *Hello Arshad Hello Ayaan Hello Areeda*

Explanation:

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

Q.3 Compare while loops and for loops. *10504003*

Ans. ★ *Key Point: Comparison between while & for loop*

Feature	While Loop	For Loop
Definition	Executes a block of code repeatedly as long as a condition is true.	Executes a block of code for a specific number of times or over each element in a sequence.
Control	Controlled by a condition (Boolean expression).	Controlled by a sequence (like range, list, string, or tuple).
Use Case	Used when the number of iterations is not known in advance.	Used when the number of iterations is known or you are iterating over a sequence.
Syntax Example	i=0 while i<5: print(i) i+=1	for i in range(1,5): print(i)
Risk	Can result in infinite loops if the condition never becomes False.	Less risk of infinite loops; iterates over a fixed range or sequence.
Increment / Step	Must be manually updated inside the loop (i += 1).	Automatically handled by the sequence or range() function.
Common Uses	Repeating until a condition changes, like waiting for user input.	Iterating over lists, strings, or generating sequences of numbers.

Q.4 What is the range() function in Python? Write its syntax with an example. *10504004*

Ans. ★ *Key Points: Use of range() function | Syntax of range() function*

In Python, the range() function is used to generate a sequence of numbers. It is commonly used in for loops to repeat actions a specific number of times.

Syntax of range() function:

- range(stop)
- range(start, stop)
- range(start, stop, step)

Example: Print the numbers from 0 to 4 OR (Print First 5 whole numbers)

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

Output: *0 1 2 3 4*

Explanation:

- The above code uses range(5), which generates a sequence of numbers starting from 0 up to (but not including) 5. Each number is printed using the for loop.

Q.5 Explain End statement.

Ans. ★ *Key Points: Use of end statement | Example of 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 the same line using "end" statement.

```
print("Ayaan",end="***")
print("Shah")
```

Output: *Ayaan***Shah*

Code:

```
print("Arshad Mehmood",end="")
print("Shah")
```

Output: *Arshad Mehmood Shah*

DID YOU KNOW?

Q. How are strings represented in Python? Give an example. (10504005a)

Ans. Strings in Python are surrounded by either single quote ' ' or double quotes " ". Both are treated the same.

Example: 'Ayaan' and "Ayaan" — both represent the same string "Ayaan".

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

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

Output: *1 2 3 4 5*

Note: Using end="" prints the numbers without spaces. If you use end=" " it would print: 1 2 3 4 5 with spaces between numbers.

Q.6 What is nested loop? Explain with examples.

Ans. ★ *Key Points: Use of nested loops | Examples of nested loops*

Nested loops are loops inside another 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.

Example 1: Nested loops for basic Number Pattern.

```
print("===Basic Number Pattern===")
for i in range(3):
    for j in range(4):
        print(f"i={i}, j={j}")
    print("--------")
```

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*

Explanation:

- The outer loop (i) controls the rows.
- The inner loop (j) controls the columns/iterations inside each row.
- print("") adds a blank line between each set of i values for clarity.

Example 2: 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):
        print("*",end="")
    print()
```

Output: ** / ** / *** / **** / ***** (one row per line)*

Explanation:

- Outer loop (i) controls the number of rows.
- Inner loop (j) prints the * symbols for each row.
- end="" keeps the stars on the same line.
- print() moves to a new line after finishing each row.

Libraries in Python

Q.7 How do Python libraries enhance programming efficiency? Provide an example. *10504007*

Ans. ★ *Key Points: Use of Python Libraries | Examples of Python Libraries*

Python libraries enhance programming efficiency by providing built-in functions and modules that help perform tasks quickly. Python offers an extensive standard library that includes built-in modules and data structures.

Importing and Using Libraries

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
number=random.randint(1,10)
print("The random number is:",number)
```

Output: *The random number is:3*

Explanation:

- The random library helps you generate random numbers, which can be useful in games, simulations, or even to pick a winner in a lucky draw.

Example: Import the statistics library to perform statistical calculations.

```
import statistics
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:48.0*

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. 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.

DID YOU KNOW?

Q. Why should you be careful when importing libraries in Python, and where can you find more information about Python libraries? (10504007a)

Ans. When importing libraries in Python, you should only import what you need to avoid unnecessary memory usage and keep your code efficient. You can explore more Python libraries and their documentation at: https://docs.python.org/3/

Lists in Python

Q.8 What is lists? Also define Creating, Accessing and Modifying lists. *10504008*

OR Describe how to work with Python lists, including creation and modification.

Ans. ★ *Key Points: Python Lists | Creation | Accessing | Modification*

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

1. Creating List

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.

2. 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'.

3. 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.

Q.9 What are some common Operations on Lists? *10504009*

Ans. ★ *Key Points: Operations on Lists | append() | remove() | sort() | reverse() | Examples*

Python provides several built-in methods to work with lists:

Method	Description	Example
append(item)	Adds an item to the end of the list.	my_list.append(5) adds 5 to the end of the list.
remove(item)	Deletes the first time that item appears in the list.	my_list.remove(3) removes the first 3 it finds.
sort()	Arranges the list in order, from smallest to biggest or ascending order.	my_list.sort() sorts numbers like 1, 2, 3, etc.
reverse()	Flips the list so the last item comes first.	my_list.reverse() turns [1, 2, 3] into [3, 2, 1].

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

```
students=["Ahmed","Sara","Ali"]
students.append("Hina") #Add "Hina" to the list
students.sort() #Sort the list alphabetically
print(students)
```

Output: *['Ahmed','Ali','Hina','Sara']*

Explanation:

- A list students is created.
- .append("Hina") adds the new student to the end of the list.
- .sort() arranges the names alphabetically.

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

```
students=["Ahmed","Sara","Ali","Hina","Sara"]
students.remove("Sara") #Removes the first occurrence of "Sara"
print(students)
```

Output: *['Ahmed','Ali','Hina','Sara']*

Explanation:

- .remove("Sara") deletes the first occurrence of "Sara" in the list.
- The remaining elements stay in their order.

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

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

Output: *[5,12,23,34,89]*

Explanation:

- .sort() arranges the numbers from smallest to largest.

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

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

Output: *['Mango','Orange','Banana','Apple']*

Explanation:

- The list fruits is created with four elements.

- The reverse() method reverses the order of the list in place.
- print(fruits) displays the reversed list.

Testing and Debugging in Python

Q.10 Write a note on Testing and Debugging. *10504010*

OR Explain the importance of testing and debugging in programming. Discuss different debugging techniques.

Ans. ★ *Key Points: Use of Testing | Types of Testing | Use of Debugging | Types of Debugging*

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

1. 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.

a. Types of Testing

- Unit Testing: Tests individual parts of the code (like functions or classes) in isolation. Python's unittest 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 break existing functionality.

b. Importance

- Helps identify and fix issues before the code is used in real-world applications.
- Ensures that the software functions reliably and as intended.

2. 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.

a. 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.

b. Importance

- Helps to identify the root cause of problems.
- Ensures that the program runs correctly and avoids unexpected crashes.

TOPIC WISE SHORT QUESTIONS (ADDITIONAL)

Decision Making

Q.1 What is a decision-making structure in programming? *10504011*

Ans. 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. Python provides a variety of conditional statements to implement decision-making.

Q.2 What are decision-making structures in Python? *10504012*

Ans. In Python, decision-making structures refer to control flow constructs that allow a program to make decisions based on conditions. These structures determine the execution path of the code.

Q.3 What are the main types of decision-making structures in Python? *10504013*

Ans. The main types are: 1. if Statement 2. if-else Statement 3. Nested Conditions

Q.4 What is an if statement and its syntax? *10504014*

Ans. 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. Proper indentation is mandatory. Incorrect indentation may cause an error (Indentation Error). Syntax: if condition: / code to run if the condition is true

Q.5 What is an if-else statement and its syntax? *10504015*

Ans. 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. It is also known as a two-way decision-making structure. Syntax: if condition: / code if true / else: / code if false

Q.6 What are nested conditions in Python? *10504016*

Ans. Nested conditions occur when one condition is placed inside another condition. They are used when you need to check multiple related conditions, allowing more precise control over the program's flow. Syntax: if cond1: / if cond2: code if both true / else: code if 1 true 2 false / else: code if cond1 false

Looping Constructs

Q.7 What is a looping structure in programming? *10504017*

Ans. A looping structure allows a set of instructions to be executed repeatedly until a specific condition is met. It helps reduce code repetition and is commonly used when performing repetitive tasks.

Q.8 How many types of Looping Structure? *10504018*

Ans. There are two types of loops: while loop and for loop.

Q.9 Define while loop. *10504019*

Ans. 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.

Q.10 How does a while loop differ from a for loop? *10504020*

Ans. 'while' repeats as long as a condition is true (e.g., counting to 10), while 'for' iterates over a sequence (e.g., list items).

Q.11 Define for loop. *10504021*

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

Q.12 What is the range() function used in for loops? *10504022*

Ans. In Python, the range() function is used to generate a sequence of numbers. It is commonly used in for loops to repeat actions a specific number of times. Syntax: range(stop) / range(start, stop) / range(start, stop, step)

Q.13 What are the uses of end statement in Python? *10504023*

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

Q.14 How are strings represented in Python? Give example. *10504024*

Ans. Strings in Python are surrounded by either single quotes ' ' or double quotes " ". Both are treated the same. Example: name1='Arshad' # Output: Arshad ; name2="Faisal" # Output: Faisal

Q.15 What is a nested loop? *10504025*

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

Libraries in Python

Q.16 What are Python libraries? *10504026*

Ans. In Python, libraries are like toolboxes full of useful tools that help you solve different problems without having to build everything from scratch.

Q.17 How do you import and use libraries in Python? *10504027*

Ans. Libraries are like pre-built toolkits that you can use without having to write all the code yourself. Example: import random / number=random.randint(1,10) / print("The random number is:", number)

Q.18 Why should you be careful when importing libraries, and where can you find more information? *10504028*

Ans. You should only import what you need to avoid unnecessary memory usage and keep your code efficient. You can explore more Python libraries and documentation at: https://docs.python.org/3/

Lists in Python

Q.19 What is a list in Python? *10504029*

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

Q.20 How do you create a list? Give an example. *10504030*

Ans. A list is created by placing items inside square brackets [ ], separated by commas. Example: fruits = ["Mango","Apple","Banana"] / print(fruits) → Output: ['Mango', 'Apple', 'Banana']

Q.21 How do you access items in a list? Give an example. *10504031*

Ans. You access items by their index, starting from 0. Example: fruits = ["Mango","Apple","Banana"] / print(fruits[1]) → Output: Apple

Q.22 How do you modify items in a list? Give an example. *10504032*

Ans. You can modify list items by accessing them via their index and assigning a new value. Example: fruits[0]="Orange"; fruits.append("Pineapple") → Output: ['Orange', 'Apple', 'Banana', 'Pineapple']

Q.23 What are some common operations/methods on lists? *10504033*

Ans. append(item): adds an item to the end. remove(item): deletes the first matching item. sort(): arranges items in ascending order. reverse(): flips the list so the last item comes first.

Q.24 Write the example of adding and sorting a list of students. *10504034*

Ans. students = ["Ahmed","Sara","Ali"]; students.append("Hina"); students.sort(); print(students) → Output: ['Ahmed','Ali','Hina','Sara']

Q.25 Write the example of removing a student from a list. *10504035*

Ans. students = ["Ahmed","Sara","Ali","Hina","Sara"]; students.remove("Sara"); print(students) → Output: ['Ahmed','Ali','Hina','Sara']

Q.26 Write the example of sorting a list of numbers. *10504036*

Ans. numbers=[34,12,89,5,23]; numbers.sort(); print(numbers) → Output: [5,12,23,34,89]

Q.27 Write the example of reversing a list. *10504037*

Ans. fruits = ["Apple","Banana","Orange","Mango"]; fruits.reverse(); print(fruits) → Output: ['Mango','Orange','Banana','Apple']

Testing and Debugging in Python

Q.28 What is testing in Python programming? *10504038*

Ans. 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.

Q.29 Write the main types of testing in Python. *10504039*

Ans. Unit Testing: tests individual code parts in isolation. Integration Testing: checks how parts work together. Functional Testing: validates behaviour from a user's perspective. Regression Testing: ensures new changes don't break existing functionality.

Q.30 What is debugging in Python? *10504040*

Ans. 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.

Q.31 Write some common debugging techniques in Python. *10504041*

Ans. Print Statements: check variable values at different stages. Debugging Tools: e.g. pdb (Python Debugger) to step through code. Error Messages: reading and interpreting them to locate the problem's source.

PREDICT THE OUTPUT

Q.32 10504042

Ans. Output: Yes

```
x=7
if x>5:
    print("Yes")
else:
    print("No")
```

Q.33 10504043

Ans. Output: No

```
x=3
if x>5:
    print("Yes")
else:
    print("No")
```

Q.34 10504044

Ans. Output: Even

```
x=10
if x%2==0:
    print("Even")
```

Q.35 10504045

Ans. Output: Odd

```
x=9
if x%2==0:
    print("Even")
else:
    print("Odd")
```

Q.36 10504046

Ans. Output: X

```
x=4
y=6
if x<y:
    print("X")
```

Q.37 10504047

Ans. Output: Y

```
x=8
y=3
if x<y:
    print("X")
else:
    print("Y")
```

Q.38 10504048

Ans. Output: In Range

```
x=5
if x>2:
    if x<10:
        print("In Range")
```

Q.39 10504049

Ans. Output: Even Big

```
x=6
if x>5:
    if x%2==0:
        print("Even Big")
```

Q.40 10504050

Ans. Output: Odd Big

```
x=7
if x>5:
    if x%2==0:
        print("Even Big")
    else:
        print("Odd Big")
```

Q.41 10504051

Ans. Output: B

```
x=3
if x>5:
    print("A")
else:
    if x>1:
        print("B")
```

Q.42 10504052

Ans. Output: False

```
x=0
if x >= 1:
    print("True")
else:
    print("False")
```

Q.43 10504053

Ans. Output: True

```
x=1
if x ==1:
    print("True")
```

Q.44 10504054

Ans. Output: Not Positive

```
x=-1
```

Q.45 10504055

Ans. Output: Equal

```
x=5
```

```
if x>0:
    print("Positive")
else:
    print("Not Positive")
```

```
if x==5:
    print("Equal")
```

Q.46 10504056

Ans. Output: Not Equal

```
x=4
if x!=5:
    print("Not Equal")
```

Q.47 10504057

Ans. Output: High

```
x=10
if x>5:
    if x>8:
        print("High")
    else:
        print("Medium")
```

Q.48 10504058

Ans. Output: Medium

```
x=6
if x>5:
    if x>8:
        print("High")
    else:
        print("Medium")
```

FIND THE ERRORS

Q.No.	Code (with error)	Error Identified	Fix
Q.49 (10504059)	x=5 if x>3 print("A")	Syntax Error: Missing colon after if	x=5 if x>3: print("A")
Q.50 (10504060)	x=2 if x>3: print("A")	Indentation Error: print not indented	x=2 if x>3: print("A")
Q.51 (10504061)	x=7 if x>5 print("Yes") else: print("No")	Syntax Error & Indentation Error: missing colon and wrong indentation	x=7 if x>5: print("Yes") else: print("No")
Q.52 (10504062)	x=10 if x%2=0: print("Even")	Syntax Error: used '=' instead of '=='	x=10 if x%2==0: print("Even")
Q.53 (10504063)	x=4 if x<5 print("Less") else print("More")	Syntax Error: missing colons after 'if' and 'else'	x=4 if x<5: print("Less") else: print("More")
Q.54 (10504064)	x=5 if x>2: if x<10 print("In Range")	Syntax Error: missing colon after inner 'if'	x=5 if x>2: if x<10: print("In Range")
Q.55 (10504065)	x=7 if x>5 if x%2==0: print("Even Big") else: print("Odd Big")	Syntax Error: missing colon after outer 'if'	x=7 if x>5: if x%2==0: print("Even Big") else: print("Odd Big")
Q.56 (10504066)	x=0 if x print("True") else: print("False")	Syntax Error: missing colon after 'if'	x=0 if x ==0: print("True") else: print("False")
Q.57 (10504067)	x=-1 if x>0: print("Positive") else print("Not Positive")	Syntax Error: missing colon after 'else'	x=-1 if x>0: print("Positive") else: print("Not Positive")
Q.58 (10504068)	x=5 if x==5 print("Equal")	Syntax Error: missing colon and indentation error	x=5 if x==5: print("Equal")

TOPIC WISE MULTIPLE CHOICE QUESTIONS (ADDITIONAL)

Decision Making

1. Which Python statement is used to make a decision based on a condition? *10504069*

(a) loop
(b) if statement
(c) function
(d) variable

2. What happens if indentation is not correct in an if statement? *10504070*

(a) The code runs normally
(b) SyntaxError
(c) IndentationError
(d) NameError

3. Which of the following allows executing one block if a condition is true and another if it is false? *10504071*

(a) if statement
(b) if-else statement
(c) nested loops
(d) while loop

4. What is nesting in Python decision-making structures? *10504072*

(a) A condition inside another condition
(b) A loop inside a function
(c) A variable inside a function
(d) A list inside another list

5. What does the following code print? *10504073*

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

(a) It's a hot day
(b) It's not a hot day
(c) 35
(d) Error

6. What is the output of this nested condition? *10504074*

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

(a) Enjoy your day!
(b) Take an umbrella
(c) Wear a raincoat
(d) No output

7. Which of the following is true about if-else statements? *10504075*

(a) They allow two possible blocks of code based on condition
(b) They execute code only if the condition is true
(c) They execute code only if the condition is false
(d) They are used for loops

Looping Constructs

8. What is a looping structure in programming? *10504076*

(a) A way to execute a block only once
(b) A structure to repeat instructions until a condition is met
(c) A function to print numbers
(d) A type of variable

9. Which of the following is a type of loop in Python? *10504077*

(a) while loop
(b) for loop
(c) both a and b
(d) if loop

10. What happens when the condition in a while loop becomes false? *10504078*

(a) The loop stops
(b) The loop continues
(c) Syntax Error occurs
(d) The loop repeats infinitely

11. What is the purpose of the range() function in Python? *10504079*

(a) To generate random numbers
(b) To print output
(c) To stop a loop
(d) To generate a sequence of numbers for iteration

12. Which of the following correctly uses the end statement in Python? *10504080*

(a) print("Hello", end="***")
(b) end("Hello")
(c) print("Hello") end("***")
(d) print(end("***"))

13. What is a nested loop? *10504081*

(a) A function inside a loop
(b) A loop inside another loop
(c) A loop with no condition
(d) A loop with only one iteration

14. Which loop is best for iterating over a known sequence like a list? *10504082*

(a) while loop
(b) nested loop
(c) if loop
(d) for loop

15. What is the output of the following while loop? *10504083*

```
number=1
while number<4:
    print(number)
    number+=1
```

(a) 1 2 3 4
(b) 1 2 3
(c) 2 3 4
(d) 1 2

16. Output of the following code: *10504084*

```
num=1
odd_count=0
while num<=5:
    if num%2==0:
        print(f"Even:{num}")
    else:
        odd_count+=1
    num+=1
print("Total odd numbers:",odd_count)
```

(a) Even: 2 Even: 4 Total odd numbers: 3
(b) Even: 2 Even: 4 Total odd numbers: 2
(c) Even: 1 Even: 3 Total odd numbers: 2
(d) Even: 1 Even: 2 Total odd numbers: 3

17. Output of the following code: *10504085*

```
friends=["Arshad","Ayaan","Areeda"]
for friend in friends:
    print("Hello",friend)
```

(a) Hello Arshad Hello Ayaan Hello Areeda
(b) Arshad Hello Ayaan Hello Areeda
(c) Hello Arshad / Hello Ayaan / Hello Areeda
(d) Error

18. Output of the following code: *10504086*

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

(a) 1 2 3 4 5

19. Output of the following code: *10504087*

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

(a) 12345

(b) 0 1 2 3 4
(c) 0 1 2 3 4 5
(d) 1 2 3 4

(b) 1 2 3 4 5
(c) 1 2 4 6
(d) 54321

20. Output of the following code: *10504088*

```
for i in range(2):
    for j in range(3):
        print(f"i={i}, j={j}")
```

(a) i=0,j=0 i=0,j=1 i=0,j=2 i=1,j=0 i=1,j=1 i=1,j=2
(b) i=0,j=0 i=1,j=0 i=0,j=1 i=1,j=1 i=0,j=2 i=1,j=2
(c) i=0,j=0 i=0,j=1 i=0,j=2
(d) i=0,j=0 i=1,j=1

21. Output of the following code: *10504089*

```
n=3
for i in range(1,n+1):
    for j in range(i):
        print("*",end="")
    print()
```

(a) * / *** / **
(b) *** / * / **
(c) * / ** / ***
(d) ** / **

Libraries in Python

22. What are Python libraries? *10504090*

(a) Variables used to store data
(b) Toolboxes full of useful tools
(c) Functions only for loops
(d) A type of loop

23. Why should you import libraries in Python? *10504091*

(a) To repeat code multiple times
(b) To print strings
(c) To solve problems without writing all the code yourself
(d) To create variables

24. Which library is used to generate random numbers in Python? *10504092*

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

25. What does the following code do? *10504093*

```
import random
number=random.randint(1,10)
print(number)
```

(a) Prints numbers from 1 to 10 sequentially
(b) Generates a random number between 1 and 10 and prints it
(c) Calculates mean of numbers
(d) Prints current date and time

26. Which library is useful for handling dates and times in Python? *10504094*

(a) random
(b) statistics
(c) datetime
(d) os

27. What is the purpose of the statistics library in Python? *10504095*

(a) To perform statistical calculations
(b) To generate random numbers
(c) To handle dates and times
(d) To create loops

28. Why should you only import what you need in Python? *10504096*

(a) To make the code run slower
(b) To avoid unnecessary memory usage
(c) To create more variables
(d) To avoid syntax errors

29. Where can you find more information about Python libraries? *10504097*

(a) https://www.python.org
(b) https://www.stackoverflow.com
(c) https://www.github.com
(d) https://docs.python.org/3/

Lists in Python

30. What is a list in Python? *10504098*

(a) A type of loop
(b) A versatile data structure that can hold a collection of items
(c) A way to print strings
(d) A function for calculations

31. How do you create a list in Python? *10504099*

(a) Using curly braces { }
(b) Using parentheses ( )
(c) Using square brackets [ ]
(d) Using angle brackets < >

32. Which of the following is a valid list? *10504100*

(a) [1, 2, 3]
(b) (1, 2, 3)
(c) {1, 2, 3}
(d) <1, 2, 3>

33. How can you access the second item in a list fruits = ["Mango", "Apple", "Banana"]? *10504101*

(a) fruits[2]
(b) fruits[1]
(c) fruits[0]
(d) fruits[-1]

34. How do you change the first element of a list to "Orange"? *10504102*

(a) fruits[1] = "Orange"
(b) fruits.append("Orange")
(c) fruits[0] = "Orange"
(d) fruits.add("Orange")

35. Which method adds a new item to the end of a list? *10504103*

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

36. What does the remove(item) method do in Python lists? *10504104*

(a) Deletes all items in the list
(b) Deletes the first occurrence of the specified item
(c) Sorts the list
(d) Adds a new item to the list

37. What is the result of sorting a list [34, 12, 89, 5, 23] using sort()? *10504105*

(a) [5, 12, 23, 34, 89]
(b) [89, 34, 23, 12, 5]
(c) [34, 12, 89, 5, 23]
(d) [12, 23, 34, 5, 89]

38. What does the reverse() method do in Python lists? *10504106*

(a) Sorts the list in ascending order
(b) Deletes the last item
(c) Flips the order of elements so the last item comes first
(d) Adds an item to the start of the list

39. What is the output of the following code? *10504107*

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

(a) ['Mango','Apple','Banana','Pineapple']
(b) ['Orange','Apple','Banana']
(c) ['Orange','Apple','Banana','Pineapple']
(d) ['Pineapple','Orange','Apple','Banana']

40. What is the output of the following code? *10504108*

```
students=["Ahmed","Sara","Ali","Hina","Sara"]
students.remove("Sara")
print(students)
```

(a) ['Ahmed','Ali','Hina','Sara']
(b) ['Sara','Ahmed','Ali','Hina']
(c) ['Ahmed','Ali','Hina']
(d) ['Ahmed','Ali','Sara','Hina','Sara']

41. What is the output of the following code? *10504109*

```
fruits=["Apple","Banana","Orange","Mango"]
fruits.reverse()
print(fruits)
```

(a) ['Apple','Banana','Orange','Mango']
(b) ['Mango','Orange','Banana','Apple']
(c) ['Banana','Apple','Mango','Orange']
(d) ['Orange','Mango','Apple','Banana']

Testing and Debugging in Python

42. What is the main purpose of testing in Python programming? *10504110*

(a) To write longer programs
(b) To find and fix issues
(c) To delete code
(d) To speed up code execution

43. Which type of testing checks individual parts of the code, like functions or classes, in isolation? *10504111*

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

44. Which type of testing ensures that new changes do not break existing functionality? *10504112*

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

45. Which testing validates that the software behaves as expected from the user's perspective? *10504113*

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

46. What is debugging in Python? *10504114*

(a) Running the program repeatedly
(b) Finding and fixing errors
(c) Printing all variables
(d) Sorting a list

47. Which of the following is a common technique for debugging? *10504115*

(a) Print Statements to check variable values
(b) Using loops
(c) Using range() function
(d) Creating lists

48. Which Python tool allows you to step through code and inspect variables during execution? *10504116*

(a) sort()
(b) pdb (Python Debugger)
(c) append()
(d) input()

49. How can error messages help in debugging? *10504117*

(a) By creating new variables
(b) By locating the source of the problem and understanding what went wrong
(c) By generating random numbers
(d) By sorting the code

50. What are the output of the following code? *10504118*

```
x=5
y=0
print(x/y)
```

(a) Finding logical errors
(b) Generating a syntax error
(c) Locating runtime errors
(d) Running the program faster

51. Which of the following is NOT a type of testing? *10504119*

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

Answers

1	2	3	4	5	6	7	8
b	c	b	a	a	c	a	b
9	10	11	12	13	14	15	16
c	a	d	a	b	d	b	a
17	18	19	20	21	22	23	24
c	b	a	a	a	b	c	b
25	26	27	28	29	30	31	32
b	c	a	b	d	b	c	a

33	34	35	36	37	38	39	40
b	c	c	b	a	c	c	a
41	42	43	44	45	46	47	48
b	b	c	a	c	b	a	b
49	50	51					
b	c	d					

SOLVED EXERCISE

Multiple Choice Questions

1. What does the range() function do in Python? *10504120*

(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? *10504121*

```
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? *10504122*

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

4. What is the purpose of control structures in programming? *10504123*

(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? *10504124*

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

6. How do you prevent print() from adding a newline? *10504125*

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

7. Which library is used to generate random numbers? *10504126*

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

8. What does list.append() do? *10504127*

(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? *10504128*

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

10. Which debugging technique involves adding output statements? *10504129*

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

Answers

1	2	3	4	5	6	7	8
b	c	b	b	c	a	c	b
9	10						
b	b						

Short Questions

Q.1 Explain the use of the range() function in for loop? *10504130*

Ans. The range() function is used to create a series of numbers. It helps decide how many times the loop should repeat. It is often used with for loops to repeat actions.

Q.2 What is the difference between the append and remove methods in Python lists? *10504131*

Ans. The append() method adds a new item to the end of the list. The remove() method takes out or deletes a specific item from the list. Append makes the list longer, while remove makes it shorter.

Q.3 Define debugging. *10504132*

Ans. Debugging is the process of finding and removing mistakes in a program. It also means fixing those mistakes so the program works right. Debugging helps make the code better and more reliable.

Q.4 What are the three parameters of the range() function? *10504133*

Ans. The three parameters are start, stop, and step. Start is where the sequence begins. Stop is where it ends. Step is the difference between numbers.

Q.5 Name three list methods and their purposes. *10504134*

Ans. Append is used to add an item to the list. Insert adds an item at a certain position. Pop removes the last item from the list.

Q.6 How do you import a library in Python? *10504135*

Ans. We use the import keyword to bring a library into the program. Libraries give us extra functions and features. For example, import math lets us use math functions.

Q.7 What is unit testing and why is it important? *10504136*

Ans. Unit testing is a technique in which programs or units are tested independently. Unit testing checks small parts of a program. It ensures each function works as it should. It is important because it helps find problems early on.

Q.8 How do you access the third element in a list? *10504137*

Ans. In Python, list indexes start from 0. The third element is at index 2. Example: mylist[2] gives the third element.

Q.9 What is the purpose of the end parameter in print()? *10504138*

Ans. The end parameter controls what comes after the printed output. By default, print() ends with a newline character, but you can change it to a space or any symbol using end=" ".

Q.10 How can we access an element from the list? *10504139*

Ans. We use index numbers to get elements from a list. The first element has an index of 0. Example: mylist[0] gives the first item.

Q.11 How do you modify an existing item in a list? *10504140*

Ans. We can change a list item by using its index. We assign a new value to that position. Example: mylist[1] = 50 changes the second item.

Long Questions

Q.1 Explain the different types of decision-making statements in Python with examples for each. *10504141*

Ans. See Long Question No. 1.

Q.2 Compare while loops and for loops. *10504142*

Ans. See Long Question No. 3.

Q.3 Describe how to work with Python lists, including creation and modification. *10504143*

Ans. See Long Question No. 8.

Q.4 Explain the importance of testing and debugging in programming. Discuss different debugging techniques. *10504144*

Ans. See Long Question No. 10.

Q.5 How do Python libraries enhance programming efficiency? Provide an example. *10504145*

Ans. See Long Question No. 7.

Q.6 Write a Python program using a loop that prints all the odd numbers between 1 to 20. *10504145*

Ans.

```
#Loop through numbers from 1 to 20
for num in range(1,21):
    if num %2!=0:              #Check if the number is odd
        print(num)
```

Output: *1 3 5 7 9 11 13 15 17 19*

Q.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.
10504146

Ans. (a) Generates a list of 5 random numbers between 1 and 20.

```
# Generates a list of 5 random numbers between 1 and 20.
#Import the random library
import random
#Generate a list of 5 random numbers between 1 and 20
random_numbers=[random.randint(1,20) for _ in range(5)]
#Print the generated list
print("Random numbers:",random_numbers)
```

Output: *Random numbers: [12, 5, 19, 8, 16]*

Note: The actual numbers will be different each time the program is executed because random.randint(1, 20) generates different random values.

(b) Uses the statistics library to calculate and print the mean of the generated numbers.

```
#Import necessary libraries
import random
import statistics
mean_value=statistics.mean(random_numbers)
print("Mean of the numbers:",mean_value)
```

Output: *Random numbers: [12, 5, 19, 8, 16] Mean of the numbers: 12.0*

Q.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". (c) Change "Karahi" to "Chicken Karahi". (d) Remove "Nihari" from the list. *10504147*

Ans.

```
(a) Create a list of popular Pakistani dishes
dishes=["Biryani","Nihari","Karahi","Seekh Kebabs"]
print("Original list:",dishes)

(b) Add a new dish "Quorma"
dishes.append("Quorma")

(c) Change "Karahi" to "Chicken Karahi"
index=dishes.index("Karahi")
dishes[index]="Chicken Karahi"

(d) Remove "Nihari"
dishes.remove("Nihari")
#Print the final updated list
print("Updated list:",dishes)
```

Output: *Original list: ['Biryani','Nihari','Karahi','Seekh Kebabs']*

Output: *Updated list: ['Biryani','Chicken Karahi','Seekh Kebabs','Quorma']*

Q.9 Write a code to print stars (asterisk) in rectangular pattern using nested loops with three rows and five columns. *10504148*

Ans.

```
#Define the number of rows and columns
rows=3
columns=5
#Outer loop for rows
for i in range(rows):
    #Inner loop for columns
    for j in range(columns):
        print("*",end="") #Print '*' without newline
    print()          #Move to the next line after each row
```

Output: `*****` / `*****` / `*****` *(three rows)*

Q.10 Write the output of the following code. *10504149*

```
print("= = = Multiplication Table (2-3) = = =")
for i in range(2, 4):              # Outer: 1 to 5
    for j in range(1, 11):          # Inner: 1 to 10
        print(f"{i} x {j} = {i*j:2d}", end="   ")
    print()                         # New line after each table
```

Ans. Output

```
= = = Multiplication Table (2-3) = = =
2 x 1 = 2   2 x 2 = 4   2 x 3 = 6   2 x 4 = 8   2 x 5 = 10   2 x 6 = 12   2 x 7 =
14   2 x 8 = 16   2 x 9 = 18   2 x 10 = 20
3 x 1 = 3   3 x 2 = 6   3 x 3 = 9   3 x 4 = 12   3 x 5 = 15   3 x 6 = 18   3 x 7 =
21   3 x 8 = 24   3 x 9 = 27   3 x 10 = 30
```

SOLVED ACTIVITIES

Activity - 1

1. Write a for loop using range() to print the even numbers from 2 to 10 on a single line. *10504150*

Solution:

```
for i in range(2, 11, 2):   # start=2, stop=11, step=2
    print(i, end=" ")
```

Output: *2 4 6 8 10*

2. Write a Python program that prints the first 10 multiples of 3 using a for loop and the range() function. *10504151*

Solution:

```
for i in range(1,11):   # 1 to 10
    print(3*i,end="")
```

Output: *3 6 9 12 15 18 21 24 27 30*

Activity - 2

Q.1 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. *10504152*

Solution:

```
import random
import statistics
# Generate a list of 10 random numbers between 1 and 50
numbers = [random.randint(1, 50) for _ in range(10)]
print("Random numbers:", numbers)
# Calculate the mean (average) using statistics module
mean_value = statistics.mean(numbers)
print("Mean (average):", mean_value)
```

Output: *Random numbers: [12, 45, 7, 30, 23, 18, 50, 4, 36, 29]*

Output: *Mean (average): 25.4*