



10th COMPUTER SCIENCE SOLVED EXERCISE

UNIT #04 — Control Structures In Python

Prepared by: Mahmood Khan | MS Cyber Security

[Codymentor Software House Sahiwal](#) | 03186879182

Multiple Choice Questions (UNIT #04)

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. What is the purpose of the range() function?

- (a) To create a list of numbers
- (b) To generate a sequence of numbers
- (c) To calculate the range of values
- (d) To sort numbers

10. Which debugging technique involves adding output statements?

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

MCQ'S Answer Key (UNIT #04):

Q. No	1	2	3	4	5	6	7	8	9	10
Answer	B	C	B	B	C	A	C	B	B	B

Short Questions (UNIT #04)

1. Explain the use of the range () function in a for loop?

Answer: We can use the range () function to generate a sequence of numbers, which 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)

It prints numbers from **0 to 4**.

2. What is the difference between the append () and remove () methods in Python lists?

Answer:

Method

Purpose

append()

Adds an item to the **end** of a list.

remove()

Removes the first occurrence of an item from the list

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

```
students = ["Ahmed", "Sara", "Ali"]
```

```
students.append("Hina")
```

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

```
students = ["Ahmed", "Sara", "Ali", "Hina", "Sara"]
```

```
students.remove("Sara")
```

3. Define debugging.

Answer: 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 statement
- Debugging tools
- Error messages

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

Answer: The three parameters of range () are:

1. **Start** – starting value
2. **Stop** – ending limit (not included)
3. **Step** – amount by which the value increases or decreases

Syntax:

```
range (start, stop, step)
```

5. Name three list methods and their purposes

Answer:

List Method	Syntax	Purpose
append()	list.append(item)	Adds an item at the end of the list.
remove()	list.remove(item)	Removes a specific item from the list.
sort()	list.sort()	Arranges the items in ascending order.

6. How do you import a library in Python?

Answer: We can import a library in Python using the import keyword.

Syntax:

```
import library_name
```

Example:

```
import math
```

This imports the **math** library so we can use its functions.

7. What is unit testing and why is it important?

Answer: Tests individual parts of the code (like functions or classes) in isolation. Python's unit test module is commonly used for this.

Importance:

- Helps find errors early.
- Ensures that each part of the program works correctly.
- Makes debugging easier.

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

Answer: We can access the third element using index 2, because Python list indexing starts from 0.

Example:

```
students = ["Ahmed", "Sara", "Ali"]
print(students[2])
```

Output: Ali

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

Answer: 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.

10. How can we assess an element from the list?

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

Example:

```
students = ["Ahmed", "Sara", "Ali"]
print(students[1])
```

Output:

Sara

11. How do you modify an existing item in a list.

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

Example:

```
students = ["Ahmed", "Sara", "Ali"]
students[1] = "Hina"
print(students)
```

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

Long Questions (UNIT #04)

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

Answer: Decision making in programming allows the program to choose different actions based on conditions. Python provides a variety of conditional statements to implement decision making.

1. if Statement

The if statement lets us make decisions based on conditions. If the condition is true, it runs a block of code.

Example:

```
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.

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**.

Example:

```
weather = "rainy"
```

```
temperature = 10
```

```
if weather == "rainy":
```

```
    if temperature < 15:
```

```
        print("Wear a rain coat")
```

```
    else:
```

```
        print("Take an umbrella")
```

```
else:
```

```
    print("Enjoy your day!")
```

Output:

Wear a rain coat

Note: In Python, proper indentation is mandatory because indentation defines the structure or scope of the code. Incorrect indentation may cause an **Indentation Error**.

2. Compare while loops and for loops.

Answer:

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

while Loop

A while loop runs as long as a condition is true.

It checks the condition before each iteration.

It stops when the condition is no longer true.

Example of while loop:

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

This loop continues until the number reaches 10.

Example of for loop:

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

This loop goes through each friend in the list and prints a greeting message.

for Loop

A for loop repeats a block of code a specific number of times.

It is commonly used to iterate over a sequence.

It can be used with a list, tuple, or string.

3. Describe how to work with Python lists, including creation and modification. Provide code examples for adding items, removing items and accessing elements.

Answer: 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 a List:

A list is created by placing items inside square brackets [], separated by commas. Lists can contain items of different types.

Example:

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

Output:

```
['Mango', 'Apple', 'Banana']
```

Accessing List Items:

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

Example:

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

Output:

```
Apple
```

Modifying a List:

We can modify list items by accessing them through their index and assigning a new value.

Example:

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

```
print(fruits)
```

Output:

```
['Orange', 'Apple', 'Banana', 'Pineapple']
```

Adding an Item:

The `append()` method adds an item to the end of the list.

```
students = ["Ahmed", "Sara", "Ali"]
```

```
students.append("Hina")
```

```
print(students)
```

Removing an Item:

The `remove()` method removes the first occurrence of an item from the list.

```
students = ["Ahmed", "Sara", "Ali", "Hina", "Sara"]
```

```
students.remove("Sara")
```

```
print(students)
```

Output:

```
['Ahmed', 'Ali', 'Hina', 'Sara']
```

Thus, Python lists can be created, accessed, modified, added to, and have items removed using built-in list operations.

4. Explain the importance of testing and debugging in programming.

Answer: Testing and debugging in Python 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:

1. Unit Testing: Tests individual parts of the code, such as functions or classes, in isolation.
2. Integration Testing: Checks how different parts of the program work together.
3. Functional Testing: Validates that the software behaves as expected from the user's perspective.
4. Regression Testing: Ensures that new changes do not affect 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:

1. Print Statements: Adding print statements to check the values of variables at different stages of the code.
2. Debugging Tools: Using tools like pdb (Python Debugger) to step through the code, inspect variables, and understand the flow of execution.
3. Error Messages: Reading and interpreting error messages to identify the source of the problem.

Conclusion:

Testing helps find issues in a program, while debugging helps find and fix the errors. Both are important for developing reliable and efficient Python programs.

5. Discuss different debugging techniques.

Answer: 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 the source of the problem

Testing: Testing different inputs helps identify incorrect results.

Code review: Checking the code carefully can help find mistakes.

6. How do Python libraries enhance programming efficiency? Provide an example.

Answer: 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 prebuilt 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.

Example1: 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
```

Example2: 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
```

6. Write a Python program using a loop that prints all the odd numbers between 1 and 20.

Answer:

Using for loop

```
for i in range(1, 21):
```

```
    if i % 2 != 0:
```

```
        print(i)
```

Using while loop

```
number = 1
```

```
while number <= 20:
```

```
    print(number)
```

```
    number += 2
```

7. Write a Python program that performs the following tasks:

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

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

Answer:

```
import random
```

```
import statistics
```

```
numbers = [random.randint(1, 20) for i in range(5)]
```

```
print("Numbers:", numbers)
```

```
print("Mean:", statistics.mean(numbers))
```

The program generates 5 random numbers between 1 and 20 and calculates their mean.

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.

Answer:

```
dishes = ["Biryani", "Nihari", "Karahi", "Seekh Kebabs"]
```

```
# Add a new dish
```

```
dishes.append("Quorma")
```

```
# Change Karahi to Chicken Karahi
```

```
dishes[2] = "Chicken Karahi"
```

```
# Remove Nihari
```

```
dishes.remove("Nihari")
```

```
print(dishes)
```

Output:

```
['Biryani', 'Chicken Karahi', 'Seekh Kebabs', 'Quorma']
```

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

Answer:

```
for i in range(3):
```

```
    for j in range(5):
```

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

```
    print()
```

Output:

```
* * * * *
```

* * * * *

* * * * *

10. Write the output of the following code.

```
print("=== Multiplication Table (1-5) ===")
```

```
for i in range(2, 4):
```

```
    for j in range(1, 11):
```

```
        print(f"{i} x {j} = {i*j:2d}", end=" ")
```

```
    print()
```

Output:

```
=== Multiplication Table (1-5) ===
```

```
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
```

Note: Although the heading says (1-5), the actual code uses range(2, 4), so it prints multiplication tables for 2 and 3 only.