



CHAPTER-5

CODE TESTING AND DEBUGGING

5.1 Introduction

In this chapter, students will learn the essential techniques of code testing and debugging to ensure their programs function correctly. They will understand the importance of testing for creating reliable applications and explore common types of programming errors and bugs. The chapter covers unit testing using Python's unittest and pytest modules, guiding students on how to write and execute test cases.

5.2 Testing and Debugging Techniques

Q. What is testing and debugging? Explain the importance of testing and debugging techniques.

Testing:

Testing is the process of checking a program to find errors and make sure it works correctly. It is essential for creating reliable and stable applications.

Debugging:

Debugging is the process of finding and fixing errors in a program. It helps the program to work correctly and efficiently.

Short Question

What is testing and debugging?

Advanced Testing and Debugging Techniques

Advanced testing and debugging techniques help to create reliable and efficient applications. These techniques go beyond basic error fixing and ensure better program quality. They help identify hidden problems that may affect the working of an application. Testing and debugging improve application performance and user satisfaction. They also make future maintenance easier.

Importance of Testing and Debugging

Testing and debugging are essential to ensure that applications work correctly under all conditions.

i. Detects and Removes Errors

Testing and debugging help to find and remove errors before the software is released.

ii. Prevents Unexpected Failures

Regular testing prevents unexpected failures during the operation of an application.

iii. Finds the Cause of Errors

Debugging helps locate the cause of errors and fixes them efficiently.

iv. Improves Reliability and User Confidence

Testing and debugging increase software reliability and improve user confidence.

Short Question

Why are testing and debugging important?

Ex.Q.1: Why is testing essential for reliable applications, and what role does it play in ensuring program stability?

Testing

Testing is the process of checking a program to find errors and make sure it works correctly. It is essential for creating reliable and stable applications. Testing is essential for creating reliable, stable, and high-quality applications.

Why Testing is Essential for Reliable Applications

Testing ensures that an application performs correctly and meets user requirements. It improves the quality of software and helps developers identify problems before the application is released.

i. Ensures Correct Results

Testing checks whether every function of the application produces the expected and correct results. It confirms that the program performs its tasks accurately.

ii. Reduces Crashes and Incorrect Outputs

Testing identifies bugs and programming errors that may cause the application to crash or produce incorrect results. Fixing these errors improves the application's performance.

iii. Verifies Software Changes

Whenever new features or updates are added, testing checks that they work correctly and do not affect the existing parts of the application. This helps maintain the overall quality of the software.

iv. Improves Stability

A thoroughly tested application performs consistently under different conditions. It is less likely to fail during normal use and provides a stable user experience.

v. Meets Application Requirements

Testing verifies that the application satisfies all functional and non-functional requirements. It ensures that the software behaves according to its design and user expectations.

vi. Increases Reliability

A well-tested application is more reliable because users can trust it to perform correctly without unexpected errors. Reliable software improves user confidence and satisfaction.

Role of Testing in Ensuring Program Stability

Testing plays an important role in ensuring program stability by checking how the application behaves in different situations. It detects errors before the software is released and helps developers fix them at an early stage. Testing also confirms that updates and modifications do not introduce new problems into the existing system. As a result, the application becomes more stable, reliable, efficient, and trustworthy for users.



CLASS ACTIVITY

Discuss scenarios where errors cause failure, and explain testing importance.

Sol: A banking application may calculate the wrong account balance because of a logic error. An online shopping website may crash during payment because of a runtime error. A student management system may produce incorrect results if invalid data is entered. Testing is important because it finds these errors before the software is released. It improves software quality, prevents failures, and increases user confidence.

Ex.Q.2. Discuss the common types of programming errors and bugs that occur during software development.

Errors

Programming errors and bugs are mistakes in a program that can cause incorrect results or prevent the program from working properly.

Debugging

Debugging is the process of finding and fixing errors in a program. It helps the program to work correctly and efficiently.

Common Types of Programming Errors and Bugs

Identifying and fixing these errors is important for building stable and reliable applications. Understanding different types of errors helps programmers develop better and more efficient software.

1. Syntax Errors

Syntax errors occur when the code does not follow the rules of the programming language. These errors prevent the program from running until they are corrected. They are usually detected by the Python interpreter before the program starts executing.

Example: Writing `if x = 5` instead of `if x == 5` in Python causes a syntax error.

Short Question

What are syntax errors? Give example.

2. Logic Errors

Logic errors occur when a program runs successfully but produces incorrect results. These errors happen due to mistakes in the program logic. They are often more difficult to identify because the program executes without showing an error message.

Example: Using addition instead of multiplication in a program that calculates the area of a rectangle gives an incorrect result.

Short Question

What are logic errors? Give example.

3. Runtime Errors

Runtime errors appear while the program is running. These errors may stop the program execution or cause unexpected behavior. They usually occur because of invalid operations or unexpected conditions during execution.

Example: Dividing a number by zero causes a runtime error because the operation is not possible.

Short Question

What are runtime errors? Give example.

4. Data Errors

Data errors occur when incorrect or invalid data is used in a program. These errors can affect the accuracy of program results. Proper input validation helps reduce data errors and improves program reliability.

Example: Entering text instead of a number in a program that calculates student marks can produce incorrect results.

Short Question

What are data errors? Give example.

5. Resource Leaks

Resource leaks occur when a program does not properly release resources such as memory or files. These problems can reduce program performance and stability. Managing resources correctly helps applications run efficiently and prevents unnecessary resource usage.

Example: A program opens a file but does not close it after use, which may waste system resources.

Short Question

What are resource leaks? Give example.

Types of Program Errors



Logic Error

```
length = 5
width = 3

area = length + width
# Incorrect logic

print(area)
# Output: 8 (Incorrect result)
```

The program runs, but the result is wrong because the formula is incorrect.



Correct logic:
`area = length * width`

Output: 15 (Correct result)



Runtime Error

```
a = 10
b = 0

result = a / b
# Division by zero
```

ZeroDivisionError:
division by zero

The program stops during execution because an invalid operation occurs.



Syntax Error

```
if x > 5
    print("x is greater")
# Missing colon (:)

# SyntaxError: invalid syntax
```

The program fails to run because the Python syntax is wrong.

Correct syntax:

```
if x > 5:
    print("x is greater")
```

Figure 5.1: Example showing logic error, runtime error and syntax error



Additional Multiple Choice Questions

- The process of checking a program to find errors and make sure it works correctly is called:
(a) Testing (b) Debugging (c) Maintenance (d) Documentation
- Debugging is the process of:
(a) Writing a program (b) Finding and fixing errors (c) Saving program files (d) Installing software
- Which of the following is an importance of testing and debugging?
(a) Increases file size (b) Prevents unexpected failures (c) Creates new programs (d) Deletes source code
- Why is testing essential for reliable applications?
(a) It checks whether an application meets its requirements. (b) It increases memory usage.
(c) It changes programming language. (d) It removes hardware.
- Which type of programming error occurs when the code does not follow the rules of the programming language?
(a) Logic error (b) Runtime error (c) Syntax error (d) Data error
- A program runs successfully but produces incorrect results. This is an example of:
(a) Syntax error (b) Resource leak (c) Runtime error (d) Logic error
- Which error appears while the program is running?
(a) Data error (b) Runtime error (c) Syntax error (d) Logic error
- Which type of error occurs when incorrect or invalid data is used in a program?
(a) Resource leak (b) Syntax error (c) Data error (d) Runtime error
- A problem that occurs when a program does not properly release resources such as memory or files is called:
(a) Logic error (b) Resource leak (c) Runtime error (d) Data error
- Which of the following best explains the role of testing in ensuring program stability?
(a) It helps applications work correctly under different conditions. (c) It removes all programming languages.
(b) It changes program logic automatically. (d) It increases hardware speed.

Answers:

1	A	2	B	3	B	4	A	5	C	6	D	7	B	8	C	9	B	10	A
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	----	---

Ex.Q.3. What is unit testing, and how do Python's unittest and pytest modules support effective testing?

Unit Testing

Unit testing is the process of testing individual parts or units of a program to ensure they work correctly. It helps to catch errors early and makes debugging easier. Unit testing also helps programmers verify that each part of the program works independently.

Importance of Unit Testing

- It tests individual parts of a program.
- It helps to catch errors early.
- It makes debugging easier.
- It checks whether each component produces the expected output.
- Clear and simple test cases improve code reliability.
- It helps programmers find and fix problems before the complete program is tested.

Python's unittest and pytest Modules

Python provides modules such as unittest and pytest for writing and executing tests. These modules allow automated testing of functions and methods. Unit tests check whether each component produces the expected output. Clear and simple test cases improve code reliability. They help developers create stable, reliable, and error-free applications.

1. unittest Module

The unittest module is a built-in Python module used for creating and running test cases. It provides a framework for testing functions, classes, and modules. It helps programmers check whether the code works according to expected results. Since it is built into Python, no separate installation is required.

Short Question

What is unittest module?

Example:

```
import unittest
def add(a, b):
    return a + b
class TestAdd(unittest.TestCase):
    def test_add(self):
        self.assertEqual(add(2, 3), 5)
unittest.main()
```

Output:

```
.....
Ran 1 test in 0.000s
OK
```

Explanation:

- unittest is imported to use the testing framework.
- add() adds two numbers.
- test_add() checks whether the result is correct.
- assertEquals() checks whether the actual result is equal to the expected result.
- unittest.main() runs the test.

2. pytest Module

The pytest module is an external Python module that makes writing and running tests easier. It simplifies the testing process and can be used to test individual functions, classes, or complete modules. It provides a simple syntax that makes writing test cases quick and easy.

Short Question

What is pytest module?

Example:

```
def add(a, b):
    return a + b
def test_add():
    assert add(2, 3) == 5
```

Output:

```
1 passed
```

Explanation:

add() adds two numbers.

test_add() tests the add() function.

assert checks whether the result is correct.

The test passes because $2 + 3 = 5$.

Both unittest and pytest improve program stability and reduce errors in large applications. They help programmers produce reliable software by identifying errors before the program is released.

Ex.Q.4. Explain the process of writing and executing test cases using Python's unittest or pytest modules.**Test Cases**

A test case is a set of inputs, expected outputs, and conditions used to check whether a specific part of a program works correctly. It helps to verify that the program produces the expected result.

Test cases define specific inputs and the expected output for a function. Python executes test cases and reports any failures or errors. Repeated testing ensures that new changes in code do not affect existing features. Automated test execution saves time and improves accuracy during software development.

Short Question

What are test cases in software testing?

Process of Writing and Executing Test Cases

i. Write Test Cases

Define test cases by providing specific inputs and the expected output for each function or program component. Well-written test cases help identify errors easily.

ii. Use unittest or pytest

Python provides the unittest and pytest modules to create and run test cases. These modules help automate the testing process and make testing more efficient.

iii. Execute the Tests

Run the test cases using the selected testing module. The module executes each test and reports whether it passes or fails.

iv. Verify Code Changes

After modifying the program, run the test cases again to ensure that the new changes have not affected the existing features.

v. Save Time and Improve Accuracy

Automated test execution saves time and improves accuracy during software development.

Using pytest

i. Simple Test Functions

pytest simplifies testing. It does not require creating test classes or writing extra setup code. Tests can be written as simple Python functions.

ii. Use of assert Statement

The assert statement checks whether the actual output matches the expected output. If the condition is true, the test passes. If it is false, the test fails and Python reports an error.

iii. Run the Tests

Save the program in a file. Open the command prompt or terminal, go to the folder where the file is saved, and type `pytest`.

iv. View the Results

pytest automatically finds the test functions, runs them, and displays whether the tests passed or failed.

Example:

```
def add(a, b):  
    return a + b  
def multiply(a, b):  
    return a * b  
def test_add():  
    assert add(2, 3) == 5  
def test_multiply():  
    assert multiply(2, 3) == 6
```

Explanation:

- **pytest simplifies testing.** It allows programmers to write test cases as simple Python functions. It does not require creating test classes for basic tests.
- **Tests use the assert statement.** The assert statement checks whether the actual result matches the expected result. If the condition is true, the test passes. If it is false, the test fails.

Running the tests. Save the program in a file, for example, `test_math.py`. Open the command prompt or terminal, go to the folder where the file is saved, and type: `pytest`

pytest automatically finds the test functions, runs them, and displays the test results.

Ex.0.5. How do breakpoints and watch expressions help in debugging? Discuss their role in identifying issues in the code.

Breakpoints

A breakpoint is a point in a Python program where the execution is temporarily stopped during debugging. It allows the programmer to check the values of variables and find errors in the code step by step.

A breakpoint is a marker that pauses program execution at a selected line of code.

IDEs such as PyCharm and VS Code allow programmers to set breakpoints with a simple click. When the program reaches a breakpoint, execution stops and the current state of the program can be examined. Breakpoints help programmers focus on a specific part of the code without running the entire program.

Short Question

What is a breakpoint and why are they used?

Watch Expressions

A watch is a debugging feature that allows the programmer to monitor the value of a specific variable or expression while the program is running. It helps to see how values change during program execution.

Monitoring Variable Values with Watch Expressions

A watch expression is used to monitor the value of a variable while the program is running.

Watch expressions automatically update when the value of a variable changes. They help identify incorrect or unexpected values during program execution. Watches allow programmers to observe how data changes step by step in real time.

Short Question

What are watch expressions and why are they used?

Role of Breakpoints and Watch Expressions

i. Analyze Program Behavior

Breakpoints and watches are debugging tools used to analyze the behavior of a program.

ii. Pause Program Execution

They allow programmers to pause program execution and examine data at specific points.

iii. Identify Errors

These tools help to identify errors and understand how the program flows.

iv. Make Debugging Faster

Using breakpoints and watches makes debugging faster and more accurate.

Short Question

Describe the role of breakpoints in debugging?

Role of Watch Expressions

i. Monitor Variable Values

A watch expression is used to monitor the value of a variable while the program is running.

ii. Detect Incorrect Values

Watch expressions automatically update when the value of a variable changes. They help identify incorrect or unexpected values during program execution.

iii. Observe Data Changes

Watches allow programmers to observe how data changes step by step in real time.

Short Question

Describe role of watch expressions in debugging?



Additional Multiple Choice Questions

- Unit testing is the process of testing of:
(a) Complete application (b) Individual parts (c) Network devices (d) Database tables
- Which of the following is an importance of unit testing?
(a) It installs software. (b) It creates new functions.
(c) It helps to catch errors early. (d) It increases memory.
- The built-in Python module used for creating and running test cases is called:
(a) pytest (b) unittest (c) tkinter (d) pandas
- Which Python module is an external module that makes writing and running tests easier?
(a) unittest (b) os (c) math (d) pytest
- A test case defines specific inputs and the expected output for a:
(a) Variable (b) Function (c) Loop (d) Comment
- Which statement checks whether the actual output matches the expected output in pytest?
(a) print (b) input (c) assert (d) return
- A point where program execution is temporarily stopped during debugging is called:
(a) Breakpoint (b) Watch expression (c) Test case (d) Assertion
- A debugging feature used to monitor the value of a variable while the program is running is called:
(a) Loop (b) Breakpoint (c) Function (d) Watch expression
- Why are breakpoints mainly used during debugging?
(a) To pause program execution and examine data. (b) To create test classes.
(c) To encrypt program files. (d) To install Python modules.
- Why are watch expressions useful during debugging?
(a) They automatically update when the value of a variable changes.
(b) They increase program speed.
(c) They compile the program automatically.
(d) They remove syntax errors automatically.

Answers:

1	B	2	C	3	B	4	D	5	B	6	C	7	A	8	D	9	A	10	A
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	----	---

Ex.Q.6. Describe how to set breakpoints in IDEs like PyCharm or VS Code, and explain their significance in the debugging process.

Setting Breakpoints in PyCharm:

A breakpoint is a point in the program where the execution stops temporarily during debugging. It allows the programmer to examine the code and check the values of variables at a specific point.

The steps to set a breakpoint in PyCharm are:

- Open the Python program in PyCharm.
- Find the line of code where you want the program to stop.
- Click on the left side of the line number in the editor window.
- A red dot appears, which indicates that a breakpoint has been set.
- Run the program in Debug mode instead of normal Run mode.
- When the program reaches the breakpoint, execution pauses, and the programmer can check variables, expressions, and program flow.

Short Question

How is a breakpoint set in PyCharm?

Significance of Breakpoints in Debugging:

Breakpoints are important because they help programmers find and fix errors in a program. They allow the programmer to:

- Stop program execution at a specific point.
- Check the current values of variables.
- Understand how the program is executing step by step.
- Identify logical errors that do not produce error messages.
- Find the exact location where the program behaves incorrectly.
- Save time by examining only the required parts of the program instead of checking the whole code.

Short Question

Why breakpoints are used? / Describe importance of breakpoints

Q. Describe how to set breakpoints in VS Code

Setting Breakpoints in VS Code:

The steps to set a breakpoint in VS Code are:

1. Open the Python file in VS Code.
2. Select the line where you want to pause the program.
3. Click on the left side of the line number.
4. A red dot appears, showing that the breakpoint is active.
5. Start the program using the Run and Debug option.
6. When execution reaches the breakpoint, the program stops, allowing the programmer to inspect the code and variable values.

Short Question

Write steps to set a breakpoint in VS Code?

Ex.Q.7. What is step-by-step debugging, and how does it help in identifying and fixing bugs in Python programs?

Step-by-Step Debugging Process

Step-by-step debugging executes code one line or function at a time. It helps programmers to follow the program logic and detect where errors occur. Developers can step into functions, step over lines, or continue execution after a breakpoint. This process makes it easier to locate and fix bugs accurately.

Example:

```
def calculate_area(radius):
    area = 3.14 * radius ** 2
    return area

def main():
    radius = 5
    print("Area of circle:", calculate_area(radius))
    radius = 10
    print("Area of circle:", calculate_area(radius))

if __name__ == "__main__":
    main()
```

Explanation:

- The function `calculate_area(radius)` calculates the area of a circle using the formula $3.14 * radius ** 2$ and returns the calculated value.
- The `main()` function tests the area calculation twice by using two different radius values: 5 and 10.
- Breakpoints can be placed inside `calculate_area()` and near the `print()` statements to check the value of radius, observe the calculated area, and understand step-by-step debugging.

How Step-by-Step Debugging Helps

I. Follows Program Logic

It helps programmers follow the program logic.

II. Detects Errors

It helps detect where errors occur.

iii. Locates and Fixes Bugs

It makes it easier to locate and fix bugs accurately.

iv. Checks Variable Values

It allows programmers to check variable values while the program is running.

Step-by-Step Debugging Options in PyCharm

Step-by-step debugging provides different options to control program execution.

Step Into

Step Into moves inside a function to see how it works.

Example:

- Use Step Into to enter the `calculate_area()` function and examine how the area is calculated.

Step Over

Step Over executes the current line and moves to the next line without entering any function.

Example:

- Use Step Over on the line `area = 3.14 * radius ** 2` to skip the function details and move to the next instruction.

Continue: Continue resumes program execution until the next breakpoint.

Example:

- After checking the value of `radius`, use Continue to run the program until it reaches the next breakpoint.

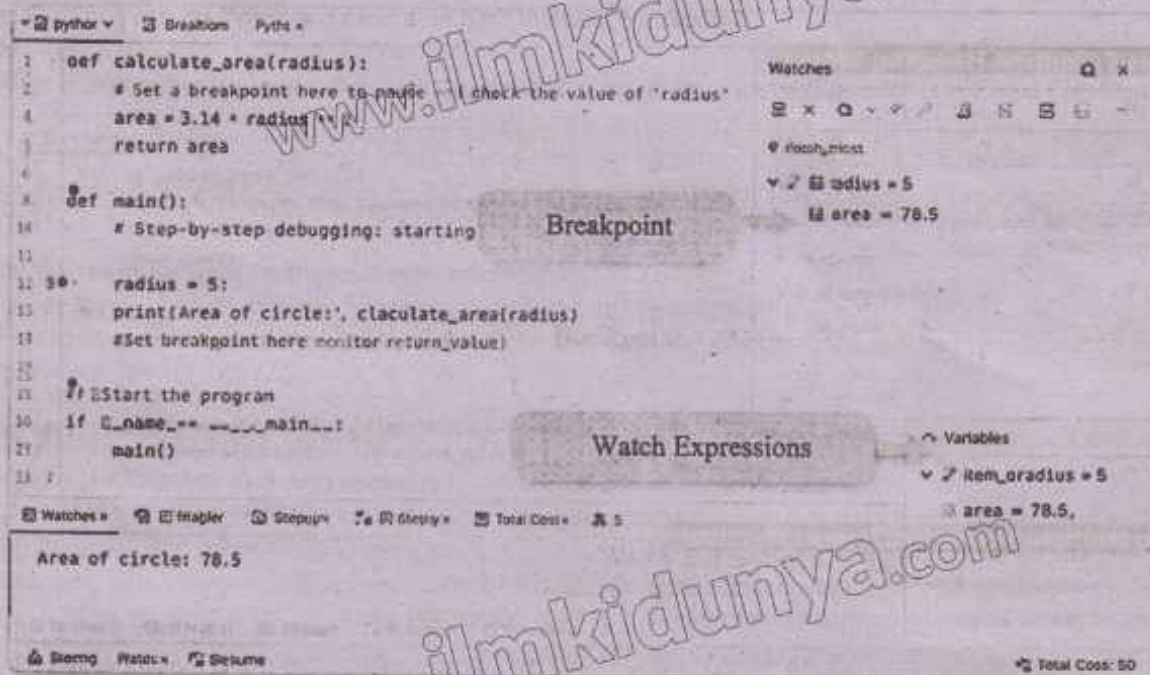


Figure 5.2: Debugging in PyCharm

Ex.Q.8. How does exception handling work in Python? Explain how to handle multiple exception types effectively to prevent program crashes.

Exception Handling

Exception handling is a technique used to manage errors that occur during program execution.

It prevents a program from crashing and allows it to respond properly to unexpected situations. Proper exception handling improves program reliability and user experience. Python provides built-in tools to catch and handle exceptions efficiently.

Short Question

What is exception handling?

Example

```
import sqlite3
try:
    #connect to database
    connection = sqlite3.connect("school.db")
    cursor = connection.cursor()
    #insert data into table
    cursor.execute("INSERT INTO students (ID, Name, Age) VALUES (?, ?, ?)", (1,
    "Ali", 20))
    #Save changes
    connection.commit()
    print("Data inserted successfully")
except sqlite3.Error as e:
    #Handle database errors
    print("Error occurred:", e)
    connection.rollback()
finally:
    #Close connection
    if connection:
        connection.close()
    print("Connection closed")
```

How Does Exception Handling Works?

try-except

The try-except block is used to handle errors and prevent the program from crashing.

commit()

The commit() method saves correct data permanently in the database. It ensures data integrity by saving the changes made during the operation.

rollback()

The rollback() method cancels changes if an error occurs. It prevents incorrect or incomplete data from being stored in the database.

finally

The finally block executes after the try-except block. It is used to perform final tasks such as safely closing the database connection.

Handling Multiple Exception Types Effectively

- A program may face different types of exceptions during execution.
- Python allows multiple exceptions to be handled in a single try-except block.
- Each type of exception can be handled with a specific response.
- This prevents unexpected program crashes and provides proper error recovery.
- Handling multiple exceptions improves program stability and gives better control over errors.



TIDBIT

- Exception handling helps programs manage unexpected problems without stopping suddenly.



CLASS ACTIVITY

Write a Python program that takes two numbers from the user and uses exception handling to display an error message if the user enters invalid input or tries to divide by zero.

Program:

```
try:
    num1 = float(input("Enter first number: "))
    num2 = float(input("Enter second number: "))
    result = num1 / num2
    print("Result:", result)
except ValueError:
    print("Error: Please enter valid numbers.")
except ZeroDivisionError:
    print("Error: Division by zero is not allowed.")
```

Output (Valid Input):

Enter first number: 10
Enter second number: 2
Result: 5.0

Output (Invalid Input):

Enter first number: abc
Error: Please enter valid numbers.

Output (Division by Zero):

Enter first number: 10
Enter second number: 0
Error: Division by zero is not allowed.



Additional Multiple Choice Questions

- Step-by-step debugging is the process of executing code:
 - At random
 - One line at a time
 - After compilation
 - Without debugging
- Which debugging option moves inside a function to see how it works?
 - Continue
 - Step Over
 - Step Into
 - Watch Expression
- Which debugging option executes the current line without entering a function?
 - Step Over
 - Step Into
 - Breakpoint
 - Continue
- Which debugging option resumes program execution until the next breakpoint?
 - Step Into
 - Watch
 - Step Over
 - Continue
- Exception handling is a technique used to:
 - Manage errors during program execution
 - Increase program size
 - Create Python modules
 - Design user interfaces
- Which block is used to handle errors and prevent the program from crashing?
 - finally
 - rollback()
 - try-except
 - commit()
- Which method saves correct data permanently in the database?
 - rollback()
 - close()
 - execute()
 - commit()
- Which method cancels changes if an error occurs?
 - commit()
 - rollback()
 - connect()
 - cursor()
- Which block is used to perform final tasks such as safely closing the database connection?
 - except
 - try
 - finally
 - commit()
- Why does Python allow multiple exceptions to be handled in a single try-except block?
 - To handle different errors correctly.
 - To make programs run faster.
 - To create database tables automatically.
 - To remove syntax errors.

Answers:

1	B	2	C	3	A	4	D	5	A	6	C	7	D	8	B	9	C	10	A
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	----	---

5.3 Profiling and Optimization

Q. Explain profiling and optimization. How do profiling tools help to identify performance bottlenecks?

Profiling and Optimization

Profiling and optimization are techniques used to improve the speed and efficiency of a program. Profiling measures the time and resources used by different parts of a program. Optimization focuses on improving slow or resource-consuming parts of code.

These techniques help in developing high-performance applications. Regular profiling ensures that programs remain efficient as they become larger.

Example:

Use of nested loop:

```
for i in range(10000):  
    for j in range(10000):  
        pass
```

In this code, the outer loop runs 10,000 times. For each iteration of the outer loop, the inner loop also runs 10,000 times.

Total executions:

$10000 \times 10000 = 100,000,000$

The inner statement may execute one hundred million times, which can make the program very slow. Such a slow part of code is called a performance bottleneck. Profiling tools help programmers find these slow sections so they can optimize the code.

Measuring Performance Using Profiling Tools

Profiling tools measure the execution time and memory usage of different functions in a program. Python provides modules such as cProfile and timeit for measuring program performance. These tools generate reports that show which parts of the code are slow or use more resources. Performance measurement helps programmers decide which areas need optimization.

Call Count	Time	Cumulative Time	Description
process_data	10	2.500 2.500	Bottleneck
read_file	1	2.400 2.400	Bottleneck
analyze_results	10	0.100 0.100	



Figure 5.3: Diagram showing profiling output highlighting bottlenecks

Identifying and Fixing Performance Bottlenecks

A performance bottleneck is a part of a program that slows down the overall performance. Profiling reports help locate these slow sections. Optimization techniques include improving algorithms, reducing unnecessary operations, and using efficient data structures. Fixing bottlenecks improves program speed and responsiveness.

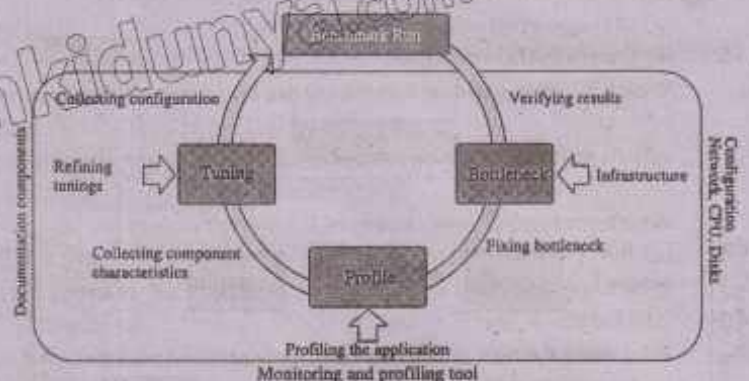


Figure 5.4: Performance Test Cycle

Example:

```

import cProfile
import timeit
# Slow code example
def slow_sum(numbers):
    total = 0
    # Add numbers one by one using a loop
    for num in numbers:
        total = total + num
    return total
# Fast code example
def fast_sum(numbers):
    return sum(numbers)
# Main function
def main():
    numbers = list(range(1, 1000000))
    print("Slow code result:", slow_sum(numbers))
    print("Fast code result:", fast_sum(numbers))
# Run the main program
main()
# Profiling the program using cProfile
print("\nProfiling Report:")
cProfile.run("main()")
# Comparing execution time using timeit
numbers = list(range(1, 1000000))
slow_time = timeit.timeit(lambda: slow_sum(numbers), number=10)
fast_time = timeit.timeit(lambda: fast_sum(numbers), number=10)
print("\nTime Comparison Report:")
print("Slow code time:", slow_time)
print("Fast code time:", fast_time)

```

In this example, `slow_sum()` adds numbers one by one using a loop, while `fast_sum()` uses Python's built-in `sum()` function.

Explanation:

- Both functions produce the same result: 499999500000.
- `slow_sum()` takes more time because it uses a manual loop.
- `fast_sum()` takes less time because it uses Python's optimized built-in `sum()` function.
- Therefore, `slow_sum()` is the performance bottleneck in this program.

Example: (The exact time values may be different on each computer, but the output will look like this)

Profiling Report:

Slow code result: 499999500000

Fast code result: 499999500000

6 function calls in 0.080 seconds

Ordered by: standard name

ncalls	tottime	percall	cumtime	percall	function
1	0.060	0.060	0.080	0.080	<module>
1	0.050	0.050	0.050	0.050	slow_sum
1	0.010	0.010	0.010	0.010	{built-in sum}
1	0.010	0.010	0.010	0.010	fast_sum
1	0.000	0.000	0.080	0.080	main

Time Comparison Report:

Slow code time: 0.42

Fast code time: 0.09

The output shows that the optimized function completes the task faster than the slow function.

? DO YOU KNOW?

- Profiling helps identify slow parts of a program so they can be improved for better performance.



Additional Multiple Choice Questions

- Profiling and optimization are techniques used to improve the:
(a) Program speed (b) Screen resolution (c) File storage (d) Network cable
- Profiling is the process of measuring the:
(a) Program design (b) Time and resources used (c) Database size (d) User interface
- Optimization mainly focuses on:
(a) Writing comments (b) Improving slow code (c) Creating variables (d) Installing software
- A part of a program that slows down the overall performance is called:
(a) Performance bottleneck (b) Syntax error (c) Test case (d) Watch expression
- Which Python module is commonly used to measure program performance?
(a) sqlite3 (b) unittest (c) cProfile (d) tkinter
- Which Python module is used to measure the execution time of code?
(a) timeit (b) pytest (c) random (d) os
- Why do programmers use profiling reports?
(a) To locate slow sections of code. (b) To create new functions.
(c) To install Python modules. (d) To design web pages.
- Which of the following is an optimization technique?
(a) Increasing unnecessary operations (c) Adding more nested loops
(b) Reducing unnecessary operations (d) Increasing memory usage
- Why is fast_sum() faster than slow_sum()?
(a) It uses Python's optimized built-in sum() function. (b) It uses more loops.
(c) It stores more variables. (d) It creates extra functions.
- Why is regular profiling important?
(a) It ensures that programs remain efficient as they become larger. (c) It increases file size automatically.
(b) It removes all programming errors. (d) It changes program logic.

Answers:

1	A	2	B	3	B	4	A	5	C	6	A	7	A	8	B	9	A	10	A
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	----	---

TEXTBOOK EXERCISE (SOLVED)



MULTIPLE CHOICE QUESTIONS

- Testing is essential for reliable applications because it helps to:
 - Optimise code performance
 - Identify and fix bugs
 - Reduce the program size
 - Make code run faster
- Common types of programming errors include:
 - Syntax error
 - Logic error
 - Runtime error
 - All of the above
- Unit testing focuses on:
 - Testing the entire application
 - Testing individual components or functions
 - Testing the user interface
 - Testing database connections
- The Python module used for unit testing is:
 - numpy
 - matplotlib
 - unittest
 - os
- The main purpose of using testing tools like unittest and pytest is:
 - To increase program size
 - To verify code works correctly
 - To design layouts
 - To compile programs
- Breakpoints in debugging are used to:
 - Run the program faster
 - Pause the program at specific points and inspect its state
 - Skip certain code sections
 - Optimise the code
- Watch expressions in debugging are used to:
 - Change variable values
 - Monitor variable values during program execution
 - Pause the program
 - Display output messages
- Step-by-step debugging is used to:
 - Run the program faster
 - Execute code one line at a time for easier error identification
 - Increase the program's performance
 - Compile the code
- The Python keyword pair used for exception handling is:
 - try-except
 - try-finally
 - try-catch
 - handle-throw
- Profiling in programming is used to:
 - Improve the user interface design
 - Measure performance and identify bottlenecks
 - Optimise memory usage only
 - Test for security vulnerabilities

Answers:

1	B	2	D	3	B	4	C	5	B	6	B	7	B	8	B	9	A	10	B
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	----	---



SHORT QUESTIONS

- Why is testing essential for ensuring reliable applications?

Ans: Testing checks whether an application meets its requirements. It ensures correct results, reduces crashes and incorrect outputs, and improves application stability.

- What are some common types of programming errors and bugs?

Ans: **Syntax Errors:** Syntax errors occur when the code does not follow the rules of the programming language.

Logic Errors: Logic errors occur when a program runs successfully but produces incorrect results.

Runtime Errors: Runtime errors appear while the program is running.

Data Errors: Data errors occur when incorrect or invalid data is used in a program.

Resource Leaks: Resource leaks occur when a program does not properly release resources.

3. **What is unit testing, and why is it important in programming?**
Ans: Unit testing is the process of testing individual parts or units of a program to ensure they work correctly. It helps to catch errors early and makes debugging easier. It improves code reliability.
4. **How do Python's unittest and pytest modules help in unit testing?**
Ans: Python provides unittest and pytest for writing and executing tests. They allow automated testing of functions and methods. Both modules improve program stability and reduce errors in large applications.
5. **What is the purpose of writing and executing test cases?**
Ans: Test cases define specific inputs and the expected output for a function. They help to find failures and ensure that new code changes do not affect existing features.
6. **How do you set breakpoints in IDEs like PyCharm or VS Code?**
Ans: Open the program and click on the left side of the line number where you want to pause execution. Run the program in Debug mode. When the program reaches the breakpoint, execution stops for checking variables and program flow.
7. **What is the role of monitoring variable values with watch expressions during debugging?**
Ans: Watch expressions monitor the value of a variable while the program is running. They help to identify incorrect or unexpected values and show how data changes step by step.
8. **What is the step-by-step debugging process in IDEs?**
Ans: Step-by-step debugging executes code one line or function at a time. It helps programmers follow program logic, detect errors, and fix bugs accurately.
9. **How does exception handling help in managing multiple exception types effectively?**
Ans: Exception handling prevents a program from crashing and allows it to respond properly to errors. Python can handle multiple exception types with separate responses, improving program stability and error recovery.
10. **What tools can be used for profiling and measuring performance in Python?**
Ans: Python provides cProfile and timeit for measuring program performance. These tools measure execution time, memory usage, and help identify slow parts of the code.



LONG QUESTIONS

All Qs are Solved in Chapter

1. Why is testing essential for reliable applications, and what role does it play in ensuring program stability?
2. Discuss the common types of programming errors and bugs that occur during software development.
3. What is unit testing, and how do Python's unittest and pytest modules support effective testing?
4. Explain the process of writing and executing test cases using Python's unittest or pytest modules.
5. How do breakpoints and watch expressions help in debugging? Discuss their role in identifying issues in the code.
6. Describe how to set breakpoints in IDEs like PyCharm or VS Code, and explain their significance in the debugging process.
7. What is step-by-step debugging, and how does it help in identifying and fixing bugs in Python programs?
8. How does exception handling work in Python? Explain how to handle multiple exception types effectively to prevent program crashes.

ADDITIONAL SHORT QUESTIONS

Testing and Debugging

1. Why are testing and debugging important?

Ans: Testing and debugging help ensure that applications work correctly under all conditions. They find and remove errors before software is released. They also improve software reliability and increase user confidence.

2. What are programming errors and bugs?

Ans: Programming errors and bugs are mistakes in a program. They can produce incorrect results or stop the program from working properly. These errors should be identified and fixed during debugging.

3. Differentiate between Testing and Debugging

Testing	Debugging
Testing is the process of checking a program to find errors and make sure it works correctly.	Debugging is the process of finding and fixing errors in a program.
Testing improves software quality and stability.	Debugging improves program correctness and efficiency.

4. Differentiate between syntax errors and logical errors

Syntax Errors	Logical Errors
Syntax errors occur when the code does not follow the rules of the programming language.	Logical errors occur when the program runs but produces incorrect results.
Syntax errors prevent the program from running until they are corrected.	Logical errors do not stop the program from running, but the output is incorrect.

Unit Testing

5. Why is unit testing important?

Ans: Unit testing helps find errors before the complete program is tested. It checks whether each component produces the expected output. It makes debugging easier and improves code reliability. This helps create stable and reliable software.

6. What are Python's unittest and pytest modules?

Ans: Python provides **unittest** and **pytest** modules for writing and running test cases. They automate the testing process and check whether the program works correctly. These modules help developers create stable and error-free applications.

7. Differentiate between unittest and pytest.

unittest	pytest
unittest is a built-in Python module.	pytest is an external Python module.
It provides a framework for creating test cases.	It provides a simpler way to write and run tests.
It requires no separate installation.	It must be installed before use.
It usually requires more code to write tests.	It uses simple test functions and less code.

Test Cases

8. Why are test cases important?

Ans: Test cases help check whether the program gives correct results. They help identify errors during testing. Repeated testing ensures that new changes do not affect existing features. This improves software quality.

9. Explain the process of writing and executing test cases.

Ans: First, write test cases with specific inputs and expected outputs. Then use unittest or pytest to run the tests. The module reports whether each test passes or fails. After code changes, run the tests again to ensure existing features still work.

10. **What is the purpose of the assert statement in pytest?**

Ans: The assert statement checks whether the actual output matches the expected output. If the condition is true, the test passes. If it is false, the test fails and Python reports an error.

11. **In pytest, what happens if the assert condition is false?**

Ans: If the assert condition is false, the test fails. Python reports an error message. This helps programmers identify and fix the problem quickly.

Breakpoints and Watch Expressions

12. **What is the role of breakpoints in debugging?**

Ans: Breakpoints pause program execution at a selected line. They allow programmers to examine variables and program flow. This helps identify errors more easily. They also make debugging faster and more accurate.

13. **What is the role of watch expressions in debugging?**

Ans: Watch expressions monitor variable values while the program is running. They automatically update whenever a value changes. This helps programmers detect incorrect values and understand how data changes.

Step-by-Step Debugging Process

14. **Why is step-by-step debugging important?**

Ans: Step-by-step debugging helps programmers understand how the program works. It shows where errors occur during execution. This makes it easier to locate and fix bugs accurately.

Step-by-Step Debugging Options in PyCharm

15. **What is Step Into?**

Ans: Step Into moves inside a function during debugging. It allows programmers to see how the function works. This helps understand the execution of each statement inside the function.

16. **What is Step Over?**

Ans: Step Over executes the current line and moves to the next line. It does not enter any function. This allows programmers to skip function details while debugging.

17. **What is Continue in debugging?**

Ans: Continue resumes program execution after a breakpoint. The program keeps running until it reaches the next breakpoint. This saves time during debugging.

18. **Differentiate between Step Into and Step Over.**

Ans:

Step Into	Step Over
Step Into enters a function during debugging.	Step Over does not enter the function.
It shows how the function works line by line.	It executes the function and moves to the next line.
It is used to examine function details.	It is used to skip function details.

19. **How do you set a breakpoint in PyCharm?**

Ans: Open the Python program in PyCharm and select the required line. Click on the left side of the line number to place a breakpoint. Run the program in Debug mode. The program stops when it reaches the breakpoint.

20. **How do you set a breakpoint in VS Code?**

Ans: Open the Python file in VS Code. Click on the left side of the required line number. A red dot appears to show the breakpoint. Run the program using the Run and Debug option.

Exception Handling

21. **Why is exception handling important?**

Ans: Exception handling prevents unexpected program crashes. It allows errors to be handled properly. This improves program reliability and user experience.

22. **What is the purpose of the try-except block?**

Ans: The try-except block is used to handle errors in a program. It catches exceptions and prevents the program from crashing. This allows the program to continue running safely.

23. **What is the purpose of commit()?**

Ans: The **commit()** method saves correct data permanently in the database. It stores all successful changes. This helps maintain data integrity.

24. **What is the purpose of rollback()?**

Ans: The **rollback()** method cancels changes if an error occurs. It prevents incorrect or incomplete data from being saved. This helps keep the database accurate.

25. **What is the purpose of the finally block?**

Ans: The **finally** block executes after the **try-except** block. It performs final tasks such as closing the database connection. It runs whether an error occurs or not.

26. **How does exception handling work in Python?**

Ans: Python uses the **try-except** block to catch errors during program execution. If an error occurs, the **except** block handles it without crashing the program. The **finally** block performs final tasks such as closing resources safely.

27. **How does Python handle multiple exception types?**

Ans: Python allows multiple exception types to be handled in a single **try-except** block. Each exception can have its own response. This prevents program crashes and improves program stability.

Profiling and Optimization

28. **Why are profiling and optimization important?**

Ans: Profiling and optimization improve the speed and efficiency of a program. They help identify slow parts of the code. This makes applications faster and more reliable.

29. **Which Python modules are used for profiling?**

Ans: Python provides **cProfile** and **timeit** modules for measuring program performance. These tools measure execution time and resource usage. They help identify parts of the program that need optimization.

Performance Bottlenecks

30. **What is a performance bottleneck?**

Ans: A performance bottleneck is a part of a program that slows down its overall performance. It uses more time or resources than other parts. Removing bottlenecks makes the program faster.

31. **What happens when a program contains a performance bottleneck?**

Ans: A performance bottleneck slows down the entire program. It increases execution time and reduces efficiency. Fixing the bottleneck improves the overall performance of the application.

32. **Differentiate Between Profiling and Optimization**

Ans:

Profiling	Optimization
Profiling measures the time and resources used by a program.	Optimization improves slow or resource-consuming parts of a program.
It helps identify performance problems.	It helps solve performance problems.
It finds slow sections of code.	It makes the code faster and more efficient.

CONCEPTUAL SHORT QUESTIONS

Programming Errors

1. Look at this code:

```
if x = 5:
```

```
    print("Hello")
```

Which type of programming error will occur? Explain.

Ans: This is a **syntax error**. The code does not follow Python's syntax rules. The program will not run until the error is corrected. Python detects this error before execution starts.

2. Look at this code:

```
result = 10 / user_input
```

If the user enters 0, what specific type of built-in Python exception will be raised? Explain.

Ans: A **ZeroDivisionError** exception will be raised. It occurs because division by zero is not allowed in Python. The program stops unless the exception is handled. Exception handling can prevent the program from crashing.

3. Look at this code:

```
area = length + width
```

The program runs successfully, but the calculated area is always incorrect. Which type of error is this? Explain.

Ans: This is a **logic error**. The program executes without showing an error message. However, the program logic is incorrect, so it produces the wrong result. Logic errors are usually difficult to identify.

4. A student enters "Ali" instead of a number in a program that calculates marks. Which type of error can occur? Explain.

Ans: This is a **data error**. It occurs because invalid data is entered into the program. Incorrect input affects the accuracy of the results. Proper input validation helps prevent this error.

5. A program opens a file but does not close it after use. Which type of bug is this? Explain.

Ans: This is a **resource leak**. It occurs when a program does not properly release resources such as files or memory. It reduces program performance and stability. Proper resource management prevents this problem.

Unit Testing

6. A programmer wants to test only the `login()` function before testing the complete application. Which testing method should be used? Explain.

Ans: The programmer should use **unit testing**. Unit testing checks one function or unit at a time. It helps find errors early. This makes debugging easier and improves software quality.

7. A programmer writes test cases using only the `assert` statement without creating test classes. Which Python testing module is being used? Explain.

Ans: The programmer is using the **pytest** module. `pytest` allows test cases to be written as simple Python functions. It uses the `assert` statement to check results. This makes testing easy and quick.

8. A programmer wants to create test cases using a built-in Python module without installing any extra package. Which module should be used? Explain.

Ans: The **unittest** module should be used. It is built into Python. It allows programmers to create and run test cases. No separate installation is required.

Debugging

9. A programmer wants the program to stop at a specific line to check variable values. Which debugging tool should be used? Explain.

Ans: The programmer should use a **breakpoint**. A breakpoint pauses program execution at the selected line. It allows the programmer to inspect variables and program flow. This helps locate errors accurately.

10. While debugging, a programmer wants to see how the value of total marks changes during program execution. Which debugging feature should be used? Explain.

Ans: The programmer should use a watch expression. It monitors the value of a variable while the program is running. The value updates automatically whenever it changes. This helps identify incorrect values.

11. During debugging, a programmer wants to examine every statement inside the calculate_area() function. Which debugging option should be used? Explain.

Ans: The programmer should use Step Into. It enters the function and executes it line by line. This helps understand how the function works. It is useful for finding errors inside functions.

12. A programmer wants to execute a function without viewing its internal statements. Which debugging option should be used? Explain.

Ans: The programmer should use Step Over. It executes the current line without entering the function. The debugger then moves to the next statement. This saves time during debugging.

Exception Handling

13. A database program successfully inserts new records. Which method should be used to save the changes permanently? Explain.

Ans: The commit() method should be used. It saves all changes permanently in the database. This ensures that the inserted data is stored correctly. It also maintains data integrity.

14. A database error occurs while inserting records. Which method should be used to cancel the incomplete changes? Explain.

Ans: The rollback() method should be used. It cancels all changes made during the failed operation. This prevents incorrect or incomplete data from being stored. It keeps the database accurate.

15. A programmer wants to close the database connection whether an error occurs or not. Which block should be used? Explain.

Ans: The finally block should be used. It always executes after the try-except block. It performs final tasks such as closing the database connection safely. This helps release system resources properly.

Profiling and Optimization

16. A Python program becomes very slow because two nested loops execute millions of times. What is this slow part of the program called? Explain.

Ans: This slow part is called a performance bottleneck. It reduces the speed of the program. Profiling tools help identify bottlenecks. After finding them, programmers can optimize the code.

17. A programmer wants to find which function takes the most execution time before improving the program. Which technique should be used? Explain.

Ans: The programmer should use profiling. Profiling measures the execution time of different functions. It identifies slow parts of the program. This helps programmers optimize the correct section of code.

TEST YOUR SKILLS

Total Marks: 40

Time: 1:30 Hour

8 x 1 = 8

Q.1 MULTIPLE CHOICE QUESTIONS

1. A program runs successfully but produces incorrect results. This is an example of:
(a) Syntax error (b) Resource leak (c) Runtime error (d) Logic error
2. A test case defines specific inputs and the expected output for a:
(a) Variable (b) Function (c) Loop (d) Comment
3. The Python module used for unit testing is:
(a) numpy (b) matplotlib (c) unittest (d) os
4. A point where program execution is temporarily stopped during debugging is called:
(a) Breakpoint (b) Watch expression (c) Test case (d) Assertion
5. A debugging feature used to monitor the value of a variable while the program is running is called:
(a) Loop (b) Breakpoint (c) Function (d) Watch expression
6. The Python keyword pair used for exception handling is:
(a) try-except (b) try-finally (c) try-catch (d) handle-throw
7. Which method saves correct data permanently in the database?
(a) rollback() (b) close() (c) execute() (d) commit()
8. Profiling in programming is used to:
(a) Improve the user interface design (b) Measure performance and identify bottlenecks
(c) Optimise memory usage only (d) Test for security vulnerabilities

Q.2 SHORT QUESTIONS:

Note: Attempt any 8 short questions

8 x 2 = 16

- i. What are resource leaks? Give example.
- ii. Why is testing essential for ensuring reliable applications?
- iii. Describe the role of breakpoints in debugging?
- iv. What are Profiling and optimization techniques?
- v. What is unit testing, and why is it important in programming?
- vi. How do Python's unittest and pytest modules help in unit testing?
- vii. How do you set breakpoints in IDEs like PyCharm or VS Code?
- viii. What are watch expressions and why are they used?
- ix. What is exception handling?
- x. What tools can be used for profiling and measuring performance in Python?

LONG QUESTIONS:

Note: Attempt any 2 long questions

2 x 8 = 16

3. Explain the process of writing and executing test cases using Python's unittest or pytest modules.
4. What is step-by-step debugging, and how does it help in identifying and fixing bugs in Python programs?
5. How does exception handling work in Python? Explain how to handle multiple exception types effectively to prevent program crashes.