Experiment 15: Using a Python Debugger

1. Aim

Demonstrate the use of a Python debugger (e.g., pdb or an IDE with debugging capabilities) on a sample program with intentional errors. Guide students on setting breakpoints, stepping through code, and examining variable values.

---

2. Theory

What is Debugging?

Debugging is the process of identifying and fixing errors (bugs) in a program. It allows developers to analyse how their code executes and track down logical, runtime, and syntax errors.

Types of Errors in Python

1. **Syntax Errors** – Code structure issues (e.g., missing colons or parentheses).
2. **Runtime Errors** – Occur during execution (e.g., division by zero, file not found).
3. **Logical Errors** – The program runs but produces incorrect results.

Debugging Tools in Python

1. **pdb (Python Debugger)** – A built-in command-line debugger.
2. **IDE Debuggers (e.g., PyCharm, VS Code)** – Provide a graphical interface with breakpoints and variable inspection.

---

3. Algorithm to Debug a Python Program

Using `pdb` (Python Debugger)

1. **Write a Python program with intentional errors.**
2. **Import the `pdb` module** to enable debugging.
3. **Set breakpoints** using `pdb.set_trace()` or run the script with `python -m pdb script.py`.
4. **Step through the code** using debugger commands:
   - `n` → Next line
   - `s` → Step into a function
   - `c` → Continue execution
   - `p variable` → Print the value of a variable
5. **Identify and fix the error** based on debugging information.

---

4. Python Code with Intentional Errors (debug_example.py)

```
import pdb

def divide_numbers(a, b):
    pdb.set_trace()  # Set a breakpoint here
    result = a / b  # Possible division by zero error
    return result

def main():
    num1 = 10
    num2 = 0  # Intentional error (division by zero)
    print("Starting Debugging...")
    output = divide_numbers(num1, num2)  # This will cause an error
    print(f"Result: {output}")

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

---

5. Explanation of Code

- **Intentional Error**: The program attempts to divide by zero (`num2 = 0`), which causes a runtime error.
- **Using `pdb.set_trace()`**:
  - It **pauses execution** at the breakpoint.
  - The user can **inspect variables** and **step through the code** interactively.
- **Debugging Steps**:
  1. Run the script in debug mode:

     ```
     python -m pdb debug_example.py
     ```

  2. Use debugger commands:
     - `p num1` → Prints the value of `num1`
     - `p num2` → Prints the value of `num2`
     - `n` → Moves to the next line
     - `q` → Quit debugging

---

6. Debugging with an IDE (PyCharm / VS Code)

1. **Open the script in an IDE** like PyCharm or VS Code.
2. **Set a Breakpoint** by clicking next to a line number.
3. **Run the script in Debug Mode** (`Shift + F9` in PyCharm, `F5` in VS Code).
4. **Step through execution** and inspect variables.
5. **Fix the bug** (e.g., change `num2 = 0` to `num2 = 5`).

---

7. Expected Output (Before Fixing the Bug)

```
ZeroDivisionError: division by zero
```

Debugger stops execution at:

```
result = a / b
```

By printing `p b`, we see that `b = 0`, causing the error.

---

8. Conclusion

In this experiment, we used **pdb and an IDE debugger** to step through a Python program, inspect variables, and identify a division-by-zero error. Debugging tools help efficiently detect and fix logical and runtime errors, improving code reliability.