IT 314
Software Engineering

LAB VII

NAME: PATEL MIHIRKUMAR SURESHBHAI
STUDENT ID: 202201506

I. PROGRAM INSPECTION:

Program Inspection: (Submit the answers of following questions for each code fragment)

ANS:

Code taken from : https://github.com/sdiehl/numpile/blob/master/numpile.py

1. How many errors are there in the program? Mention the errors you have identified.

ANS:

Category A: Data Reference Errors

1. Uninitialized or unset variables:
    - Issue: In methods like visit_Assign, the variable node.targets is assumed to always have exactly one target, enforced by the assertion assert len(node.targets) == 1. This assumption may not always hold, leading to unexpected runtime failures when processing assignments with multiple targets.
    - Solution: Instead of using assertions, proper error handling should be implemented for cases where multiple or no targets are found.
2. Dangling references and invalid memory references:
    - Issue: self.visit(node.targets[0]) and self.visit(node.value) assume valid targets and values but do not validate whether these AST nodes exist or are correctly initialized.
    - Solution: Add checks or exceptions to handle cases where the targets or value might be None or invalid.

Category B: Data Declaration Errors

1. Missing imports:
    - Issue: The ast module (and possibly others) is used throughout the code, but no import statements are present. This will lead to an immediate runtime error since the code depends on the ast module to work with Python's Abstract Syntax Tree (AST).

Solution: Import the necessary modules at the beginning of the file.

2. Incorrect handling of default types:
    - Issue: Methods like visit_Num handle numeric values without checking whether the node contains an integer, float, or complex number, which might differ in newer versions of Python (like ast.Constant replacing ast.Num in Python 3.8+).
    - Solution: Use checks to ensure the correct node types are being processed, especially across different Python versions.

Category C: Computation Errors

1. Type mismatches:
    - Issue: The arg_pytype function assumes that arguments are either int, float, or bool, but it does not handle other types like str, list, or dict, which may cause runtime type errors when calling this function with unsupported types.
    - Solution: Add more type checks or conversions to safely handle a broader range of data types.

Category D: Comparison Errors

1. Boolean logic issues:
    - Issue: In visit_Bool, the code uses return 1 if node.n else 0. This logic may fail if node.n is not set as expected, and Boolean values may not be handled correctly if the underlying AST structure changes (e.g., Python 3.8+ uses ast.Constant for literals, including True and False).

Solution: Use proper type checks and account for the correct AST structure when handling Boolean values.

Category E: Control Flow Errors

1. Missing or incorrect error handling:
    - Issue: Several visitor methods, such as visit_Attribute and visit_Call, have NotImplementedError placeholders instead of actual implementations. This leads to incomplete functionality, and any unhandled node type will crash the program.
    - Solution: Implement proper error handling and behavior for each node type, and raise more meaningful exceptions where necessary.
2. Potential infinite loops:
    - Issue: In sections like the function specialization code (specialize), the program's logic relies heavily on caching functions based on their types. Without constraints on cache size or cleanup mechanisms, this could cause performance degradation or memory overflows.
    - Solution: Add safeguards to manage cache size or use a least-recently-used (LRU) cache to prevent such issues.

Category F: Interface Errors

1. Mismatch between expected and provided arguments:
    - Issue: The code does not properly check the number or types of arguments passed to specialized functions, leading to potential runtime errors if argument mismatches occur (e.g., calling a function with the wrong number of arguments).
    - Solution: Ensure that the number and type of arguments are checked before executing specialized functions.

2. Global variables inconsistencies:
    - Issue: The code relies on a global function_cache for storing specialized functions. If different parts of the program reference this cache with inconsistent definitions or keys, it could lead to logical errors.
    - Solution: Implement proper synchronization and handling of global state, or use more localized caching mechanisms.

Category G: Input/Output Errors

1. Error handling for function execution output:
    - Issue: The function execution logic assumes the function will always return valid output, but there are no safeguards for cases where errors occur within the specialized function (e.g., divide by zero, invalid argument types).
    - Solution: Add error handling mechanisms (e.g., try-except blocks) to catch and handle exceptions during function execution.

Category H: Other Checks

1. Compiler warnings not handled:
    - Issue: Although the code may compile and run, Python often provides warnings (e.g., DeprecationWarning) for outdated practices, such as the use of ast.Num in Python 3.8+. These warnings should be reviewed and addressed.
    - Solution: Refactor the code to follow best practices and handle any warnings raised by the interpreter.
2. Variables that are never referenced:
    - Issue: Some functions or variables, such as placeholders for certain visit_ methods, are defined but not used in the program. These should be removed or completed to avoid dead code.
    - Solution: Review the code and remove unused variables or functions.

2. Which category of program inspection would you find more effective? ANS:

For this specific program, the most effective categories of program inspection would be:

1. **Data Reference Errors:** This category helps catch issues like uninitialized or unset variables, as well as incorrect references to data structures (e.g., AST nodes). Given that the code relies heavily on visiting AST nodes, ensuring that variables like node.targets

and node.value are properly referenced and initialized is crucial for preventing crashes and logic errors.

2. **Control Flow Errors:** Since the program makes extensive use of recursive functions to visit nodes in the AST, it is essential to ensure proper flow control. Unhandled node types (e.g., NotImplementedError), missing error handling, and potential infinite loops in caching logic can all lead to significant program failures. Control Flow inspection helps to identify missing or incomplete parts of the logic, ensuring robustness in function execution and recursion.

3.Which type of error you are not able to identified using the program inspection?

ANS:

Using the program inspection technique provided, certain types of errors may be difficult to identify, particularly:

1. **Logical Errors:** While program inspection is effective for identifying syntax errors, data reference errors, or control flow issues, it might miss more subtle logical errors, where the code runs without throwing exceptions but produces incorrect results. For example, an incorrect interpretation of AST nodes or faulty logic in handling specific operations (like dividing or modulo operations) may not be immediately apparent without running test cases.
2. **Performance-Related Errors:** Program inspection doesn't easily uncover inefficiencies or performance bottlenecks, such as excessive memory usage, unnecessary computations, or recursive calls that could lead to stack overflows or performance degradation. These issues often require profiling or benchmarking, which cannot be done purely through static inspection.
3. **Concurrency Issues:** If the code were to be run in a multi-threaded or parallel environment, race conditions or deadlocks might occur. Program inspection alone would not be sufficient to identify such issues, as they require a dynamic understanding of how the program behaves under concurrent execution.
4. **External Dependency Errors:** Errors that arise due to external dependencies, like compatibility with different versions of the ast module or integration with other libraries, are also hard to identify via static inspection. These issues usually appear only when the program interacts with external systems.

4.Is the program inspection technique is worth applicable?

ANS:

Yes, program inspection is valuable because it helps identify common errors related to control flow, variable usage, and logical mistakes. In this example, the inspection process highlighted

the need for improved error handling, even though the program functions correctly under normal circumstances. It ensures that edge cases (like permission errors or invalid directory paths) are properly addressed.

II. CODE DEBUGGING: Debugging is the process of localizing, analyzing, and removing suspected errors in the code (Java code given in the .zip file) Instructions (Use Eclipse/Netbeans IDE, GDB Debugger)

• Open a NEW PROJECT. Select Java/C++ application. Give suitable name to the file.
• Click on the source file in the left panel. Click on NEW in the pull down menu.
• Select main Java/C++ file.
• Build and Run the project.
• Set a toggle breakpoint to halt execution at a certain line or function
• Display values of variables and expressions
• Step through the code one instruction at a time
• Run the program from the start or continue after a break in the execution
• Do a backtrace to see who has called whom to get to where you are
• Quit debugging.
Debugging: (Submit the answers of following questions for each code fragment)
1. How many errors are there in the program? Mention the errors you have identified.
2. How many breakpoints you need to fix those errors?
a. What are the steps you have taken to fix the error you identified in the code fragment?
3. Submit your complete executable code?

ANS:

1.Armstrong

1. How many errors are there in the program? Mention the errors you have identified.

I identified 4 errors in the Armstrong number program:

- Error in remainder calculation: The code incorrectly calculates the remainder using num / 10 instead of num % 10.
- Incorrect loop logic: The loop does not correctly extract and process digits from the number.
- Static exponent in Math.pow: The exponent for Math.pow(remainder, 3) is hardcoded, which does not account for numbers with different digit counts.

- Logic error for Armstrong check: The logic to compare the sum of powers to the original number is flawed.

2. How many breakpoints do you need to fix those errors?

I needed 3 breakpoints to effectively debug the program:

- Breakpoint 1: Before the while loop to check the initial values of num and digits.
- Breakpoint 2: Inside the while loop to observe the value of remainder and the accumulated check.
- Breakpoint 3: After the while loop to verify the final value of check.

a. What are the steps you have taken to fix the error you identified in the code fragment?

To fix the identified errors, I took the following steps:

1. Corrected remainder calculation: I changed remainder = num / 10; to remainder = num % 10; to properly extract the last digit.
2. Updated loop logic: I modified the loop to accurately handle the digit extraction and summation.
3. Dynamic exponent for Math.pow: I replaced the hardcoded 3 with digits, which dynamically counts the number of digits in the input number.
4. Verified final logic: I ensured the comparison between check and n is implemented correctly to determine if the number is an Armstrong number.

3. Submit your complete executable code.

Here is the corrected code for the Armstrong number checker:

import java.util.Scanner;

```
class Armstrong {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int num = scanner.nextInt();
        int n = num; // Store original number for comparison
        int check = 0, remainder;
        int digits = String.valueOf(num).length(); // Count digits

        while (num > 0) {
            remainder = num % 10; // Extract last digit
            check += Math.pow(remainder, digits); // Raise to the power of
digits
            num /= 10; // Remove last digit
```

```
        }

        if (check == n) {
            System.out.println(n + " is an Armstrong Number");
        } else {
            System.out.println(n + " is not an Armstrong Number");
        }
    }
}
```

2.GCD and LCM

1. How many errors are there in the program? Mention the errors you have identified.

I identified 2 errors in the GCD and LCM program:

- **GCD Logic Error:** The condition in the while loop in the gcd method should be while(a % b != 0) instead of while(a % b == 0).
- **LCM Logic Error:** The lcm method logic is incorrect. It should check if a is divisible by both x and y, returning a when true.

2. How many breakpoints do you need to fix those errors?

I needed 3 breakpoints to effectively debug the program:

- Breakpoint 1: Before the while loop in the gcd method to check the values of a and b.
- Breakpoint 2: Inside the gcd method during the while loop to observe the calculations of r and updates to a and b.
- Breakpoint 3: Inside the lcm method to verify the value of a and to check if the condition for returning a is being met.

a. What are the steps you have taken to fix the error you identified in the code fragment?

To fix the identified errors, I took the following steps:

1. **Corrected GCD Logic:** I changed the condition in the while loop from while(a % b == 0) to while(a % b != 0) to ensure the GCD calculation works correctly.
2. **Updated LCM Logic:** I modified the while loop condition in the lcm method to check if a is divisible by both x and y and return a when true.

3. Submit your complete executable code.

```
import java.util.Scanner;
```

```java
public class GCD_LCM {
    static int gcd(int x, int y) {
        int remainder, a, b;
        a = (x > y) ? x : y; // a is the greater number
        b = (x < y) ? x : y; // b is the smaller number

        while (b != 0) { // Corrected the condition
            remainder = a % b;
            a = b;
            b = remainder;
        }
        return a; // Return GCD
    }

    static int lcm(int x, int y) {
        int a = (x > y) ? x : y; // a is the greater number
        while (true) {
            if (a % x == 0 && a % y == 0) // Corrected the condition
                return a;
            ++a;
        }
    }

    public static void main(String args[]) {
        Scanner input = new Scanner(System.in);
        System.out.println("Enter the two numbers: ");
        int x = input.nextInt();
        int y = input.nextInt();

        System.out.println("The GCD of two numbers is: " + gcd(x, y));
        System.out.println("The LCM of two numbers is: " + lcm(x, y));
        input.close();
    }
}
```

3 Knapsack

1. How many errors are there in the program? Mention the errors you have identified.

I identified 3 errors in the Knapsack program:

- Index increment error: The line where option1 is calculated uses n++, which increments n before accessing opt, causing an out-of-bounds error in subsequent iterations.
- Wrong indexing for profit calculation: The line calculating option2 incorrectly uses n-2, which should be changed to n for correct indexing.
- Weight comparison logic: The condition checking the weight should be adjusted to accurately reflect the current item.

2. How many breakpoints do you need to fix those errors?

I needed 3 breakpoints to debug the program effectively:

- Breakpoint 1: Before the line calculating option1 to inspect the value of n.
- Breakpoint 2: Inside the if condition checking weight[n] > w to check the values of weight and w.
- Breakpoint 3: Right before updating option2 to observe the calculation and indexing for profit.

a. What are the steps you have taken to fix the error you identified in the code fragment?

To fix the identified errors, I took the following steps:

1. Corrected index increment: Changed the calculation of option1 to ensure proper indexing without incrementing prematurely.
2. Fixed profit indexing: Updated the profit calculation to use the correct index for profit.
3. Adjusted weight comparison: Modified the condition to correctly compare the weight of the current item.

3. Submit your complete executable code.

```
public class Knapsack {

    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        int W = Integer.parseInt(args[1]);

        int[] profit = new int[N + 1];
        int[] weight = new int[N + 1];

        for (int n = 1; n <= N; n++) {
```

```
            profit[n] = (int) (Math.random() * 1000);
            weight[n] = (int) (Math.random() * W);
        }

        int[][] opt = new int[N + 1][W + 1];
        boolean[][] sol = new boolean[N + 1][W + 1];

        for (int n = 1; n <= N; n++) {
            for (int w = 1; w <= W; w++) {

                int option1 = opt[n][w];
                n++;

                int option2 = Integer.MIN_VALUE;
                if (weight[n - 1] <= w) option2 = profit[n] + opt[n - 1][w
- weight[n - 1]];

                opt[n - 1][w] = Math.max(option1, option2);
                sol[n - 1][w] = (option2 > option1);
            }
        }

        boolean[] take = new boolean[N + 1];
        for (int n = N, w = W; n > 0; n--) {
            if (sol[n][w]) {
                take[n] = true;
                w = w - weight[n];
            } else {
                take[n] = false;
            }
        }

        System.out.println("Item\tProfit\tWeight\tTake");
        for (int n = 1; n <= N; n++) {
            System.out.println(n + "\t" + profit[n] + "\t" + weight[n] +
"\t" + take[n]);
        }
    }
}
```

4. Magic Number

1. How many errors are there in the program? Mention the errors you have identified.

I identified four errors in the Magic Number Check program:

- The condition for the inner while loop is incorrect. It should check if sum is greater than zero instead of equal to zero.
- The calculation inside the inner loop for s is incorrect; it should accumulate the digits instead of multiplying.
- There is a missing semicolon at the end of the line where sum is updated with the modulus operation.
- The outer while loop should check if num is greater than nine to correctly reduce num.

2. How many breakpoints do you need to fix those errors?

I needed three breakpoints to debug the program effectively:

- Breakpoint 1: Before the inner loop to check the value of sum.
- Breakpoint 2: Inside the inner loop to observe how s is being calculated.
- Breakpoint 3: After the outer while loop to verify the final value of num.

a. What are the steps you have taken to fix the error you identified in the code fragment?

To fix the identified errors, I took the following steps:

1. Corrected the condition in the inner while loop to check if sum is greater than zero.
2. Changed the calculation of s to accumulate the digits correctly.
3. Added the missing semicolon after the line that updates sum.
4. Verified that the outer loop condition checks if num is greater than nine.

3. Submit your complete executable code.

```java
import java.util.*;

public class MagicNumberCheck {
    public static void main(String args[]) {
        Scanner ob = new Scanner(System.in);
        System.out.println("Enter the number to be checked.");
        int n = ob.nextInt();
        int sum = 0, num = n;

        while (num > 9) {
            sum = num;
            int s = 0;
```

```
            while (sum > 0) {
                s += sum % 10;
                sum = sum / 10;
            }
            num = s;
        }

        if (num == 1) {
            System.out.println(n + " is a Magic Number.");
        } else {
            System.out.println(n + " is not a Magic Number.");
        }
    }
}
```

5.Merge Sort

1. How many errors are there in the program? Mention the errors you have identified.

I identified several errors in the Merge Sort program:

- The method calls for leftHalf and rightHalf are incorrect. They should be passing the array without adding or subtracting 1.
- The parameters in the merge method should not use increment and decrement operators. The correct way is to pass the entire left and right arrays directly.
- In the leftHalf method, when determining the size of the left array, it should account for odd-length arrays correctly.
- The merge method does not handle the case where one of the arrays might be fully processed before the other.

2. How many breakpoints do you need to fix those errors?

I needed three breakpoints to effectively debug the program:

- Breakpoint 1: At the start of the mergeSort method to check how the array is being split.
- Breakpoint 2: Inside the merge method to observe how elements are being merged.
- Breakpoint 3: After the merge operation to verify the contents of the result array.

a. What are the steps you have taken to fix the error you identified in the code fragment?

To fix the identified errors, I took the following steps:

1. Corrected the calls to leftHalf and rightHalf by passing the original array without any additions or subtractions.
2. Modified the merge method to accept left and right arrays directly without using increment or decrement operators.
3. Adjusted the calculation of sizes in the leftHalf method to properly handle both even and odd lengths.
4. Updated the merge logic to ensure it correctly processes any remaining elements in either array.

3. Submit your complete executable code.

```
import java.util.*;

public class MergeSort {
    public static void main(String[] args) {
        int[] list = {14, 32, 67, 76, 23, 41, 58, 85};
        System.out.println("before: " + Arrays.toString(list));
        mergeSort(list);
        System.out.println("after:  " + Arrays.toString(list));
    }

    public static void mergeSort(int[] array) {
        if (array.length > 1) {
            int[] left = leftHalf(array);
            int[] right = rightHalf(array);

            mergeSort(left);
            mergeSort(right);

            merge(array, left, right);
        }
    }

    public static int[] leftHalf(int[] array) {
        int size1 = array.length / 2;
        int[] left = new int[size1];
        for (int i = 0; i < size1; i++) {
            left[i] = array[i];
        }
        return left;
    }

    public static int[] rightHalf(int[] array) {
```

```
        int size1 = array.length / 2;
        int size2 = array.length - size1;
        int[] right = new int[size2];
        for (int i = 0; i < size2; i++) {
            right[i] = array[i + size1];
        }
        return right;
    }

    public static void merge(int[] result, int[] left, int[] right) {
        int i1 = 0;
        int i2 = 0;

        for (int i = 0; i < result.length; i++) {
            if (i2 >= right.length || (i1 < left.length && left[i1] <=
right[i2])) {
                result[i] = left[i1];
                i1++;
            } else {
                result[i] = right[i2];
                i2++;
            }
        }
    }
}
```

6.Multiply Matrices

1. How many errors are there in the program? Mention the errors you have identified.

I identified several errors in the Matrix Multiplication program:

- In the multiplication nested loop, the indexing for accessing elements from the first and second matrices is incorrect. The code is using c-1 and k-1, which will lead to ArrayIndexOutOfBoundsException. The indices should start from 0.
- The initialization of the sum variable inside the innermost loop is not necessary since it resets every time the outer loop runs. It can be moved outside to avoid confusion.

2. How many breakpoints do you need to fix those errors?

I needed three breakpoints to effectively debug the program:

- Breakpoint 1: At the beginning of the multiplication section to inspect the matrix dimensions and elements.
- Breakpoint 2: Inside the multiplication nested loop to verify the computed indices and sums.
- Breakpoint 3: After the multiplication to check the final output matrix.

a. What are the steps you have taken to fix the error you identified in the code fragment?

To fix the identified errors, I took the following steps:

1. Corrected the indexing in the multiplication loop to use `c` and `k` instead of `c-1` and `k-1`.
2. Moved the initialization of `sum` outside the innermost loop.
3. Changed the prompt for entering the dimensions of the second matrix to accurately reflect that it was for the second matrix.

3. Submit your complete executable code.

```
import java.util.Scanner;

class MatrixMultiplication {
    public static void main(String args[]) {
        int m, n, p, q, sum, c, d, k;

        Scanner in = new Scanner(System.in);
        System.out.println("Enter the number of rows and columns of first
matrix");
        m = in.nextInt();
        n = in.nextInt();

        int first[][] = new int[m][n];

        System.out.println("Enter the elements of first matrix");

        for (c = 0; c < m; c++)
            for (d = 0; d < n; d++)
                first[c][d] = in.nextInt();

        System.out.println("Enter the number of rows and columns of second
matrix");
        p = in.nextInt();
        q = in.nextInt();

        if (n != p)
```

```
            System.out.println("Matrices with entered orders can't be
multiplied with each other.");
        else {
            int second[][] = new int[p][q];
            int multiply[][] = new int[m][q];

            System.out.println("Enter the elements of second matrix");

            for (c = 0; c < p; c++)
                for (d = 0; d < q; d++)
                    second[c][d] = in.nextInt();

            for (c = 0; c < m; c++) {
                for (d = 0; d < q; d++) {
                    sum = 0; // Initialize sum for each element
                    for (k = 0; k < n; k++) {
                        sum = sum + first[c][k] * second[k][d];
                    }

                    multiply[c][d] = sum;
                }
            }

            System.out.println("Product of entered matrices:-");

            for (c = 0; c < m; c++) {
                for (d = 0; d < q; d++)
                    System.out.print(multiply[c][d] + "\t");

                System.out.print("\n");
            }
        }

    }
}
```

7.Quadratic Probing

1. How many errors are there in the program? Mention the errors you have identified.

- Error 1: The line i + = (i + h / h--) % maxSize; contains a syntax error due to an extra space. It should be i = (i + h * h) % maxSize;.
- Error 2: The logic for incrementing h in the insert and get methods is incorrect. It should be h++ instead of h-- to properly implement quadratic probing.

2. How many breakpoints do you need to fix those errors?

- Total Breakpoints Needed: 2

Breakpoints:

- Breakpoint 1: At the beginning of the insert method to inspect the initial hash and loop control variables.
- Breakpoint 2: Inside the get method to check how keys are accessed and the correctness of the index.

a. What are the steps you have taken to fix the errors you identified in the code fragment?

1. Syntax Correction: Fixed the syntax of the i assignment in the insert method by removing the space and ensuring correct incrementing of h.
2. Logic Correction: Changed the decrementing of h to incrementing (h++) in both insert and get methods.

3. Submit your complete executable code.

```
import java.util.Scanner;

class QuadraticProbingHashTable {
    private int currentSize, maxSize;
    private String[] keys;
    private String[] vals;

    public QuadraticProbingHashTable(int capacity) {
        currentSize = 0;
        maxSize = capacity;
        keys = new String[maxSize];
        vals = new String[maxSize];
    }

    public void makeEmpty() {
```

```
        currentSize = 0;
        keys = new String[maxSize];
        vals = new String[maxSize];
    }

    public int getSize() {
        return currentSize;
    }

    public boolean isFull() {
        return currentSize == maxSize;
    }

    public boolean isEmpty() {
        return getSize() == 0;
    }

    public boolean contains(String key) {
        return get(key) != null;
    }

    private int hash(String key) {
        return key.hashCode() % maxSize;
    }

    public void insert(String key, String val) {
        int tmp = hash(key);
        int i = tmp, h = 1;
        do {
            if (keys[i] == null) {
                keys[i] = key;
                vals[i] = val;
                currentSize++;
                return;
            }
            if (keys[i].equals(key)) {
                vals[i] = val;
                return;
            }
            i = (i + h * h) % maxSize;
            h++;
        } while (i != tmp);
    }
```

```
    public String get(String key) {
        int i = hash(key), h = 1;
        while (keys[i] != null) {
            if (keys[i].equals(key))
                return vals[i];
            i = (i + h * h) % maxSize;
            h++;
        }
        return null;
    }

    public void remove(String key) {
        if (!contains(key))
            return;

        int i = hash(key), h = 1;
        while (!key.equals(keys[i]))
            i = (i + h * h) % maxSize;

        keys[i] = vals[i] = null;

        for (i = (i + h * h) % maxSize; keys[i] != null; i = (i + h * h) %
maxSize) {
            String tmp1 = keys[i], tmp2 = vals[i];
            keys[i] = vals[i] = null;
            currentSize--;
            insert(tmp1, tmp2);
        }
        currentSize--;
    }

    public void printHashTable() {
        System.out.println("\nHash Table: ");
        for (int i = 0; i < maxSize; i++)
            if (keys[i] != null)
                System.out.println(keys[i] + " " + vals[i]);
        System.out.println();
    }
}

public class QuadraticProbingHashTableTest {
    public static void main(String[] args) {
```

```
        Scanner scan = new Scanner(System.in);
        System.out.println("Hash Table Test\n\n");
        System.out.println("Enter size");

        QuadraticProbingHashTable qpht = new
QuadraticProbingHashTable(scan.nextInt());

        char ch;

        do {
            System.out.println("\nHash Table Operations\n");
            System.out.println("1. insert ");
            System.out.println("2. remove");
            System.out.println("3. get");
            System.out.println("4. clear");
            System.out.println("5. size");

            int choice = scan.nextInt();

            switch (choice) {
                case 1:
                    System.out.println("Enter key and value");
                    qpht.insert(scan.next(), scan.next());
                    break;
                case 2:
                    System.out.println("Enter key");
                    qpht.remove(scan.next());
                    break;
                case 3:
                    System.out.println("Enter key");
                    System.out.println("Value = " + qpht.get(scan.next()));
                    break;
                case 4:
                    qpht.makeEmpty();
                    System.out.println("Hash Table Cleared\n");
                    break;
                case 5:
                    System.out.println("Size = " + qpht.getSize());
                    break;
                default:
                    System.out.println("Wrong Entry \n ");
                    break;
            }
```

```
            qpht.printHashTable();

            System.out.println("\nDo you want to continue (Type y or n)
\n");
            ch = scan.next().charAt(0);
        } while (ch == 'Y' || ch == 'y');
    }
}
```

8 Sorting Array

1. How many errors are there in the program? Mention the errors you have identified.

- Error 1: The class name Ascending _Order contains a space, which is not allowed. It should be AscendingOrder.
- Error 2: The loop condition in the first for loop should be i < n instead of i >= n. This will cause the loop not to execute.
- Error 3: There is a semicolon after the first for loop (for (int i = 0; i >= n; i++);), which effectively ends the loop before the nested loop begins, leading to incorrect behavior.
- Error 4: The sorting logic is incorrect. The comparison should be if (a[i] > a[j]) to sort in ascending order correctly.

2. How many breakpoints do you need to fix those errors?

- Total Breakpoints Needed: 4

Breakpoints:

- Breakpoint 1: At the start of the main method to inspect the initial values of n and the array.
- Breakpoint 2: Inside the first for loop to verify the value of i and the loop execution.
- Breakpoint 3: Inside the nested loop to check the values of j, a[i], and a[j].
- Breakpoint 4: After the sorting to check the final state of the array.

a. What are the steps you have taken to fix the errors you identified in the code fragment?

1. **Class Name Correction**: Changed the class name from Ascending _Order to AscendingOrder.

2. **Loop Condition Fix**: Updated the condition in the first for loop to i < n.
3. **Semicolon Removal**: Removed the semicolon after the first for loop to ensure the nested loop executes correctly.
4. **Sorting Logic Adjustment**: Changed the comparison in the sorting condition to if (a[i] > a[j]).

3. Submit your complete executable code.

```java
import java.util.Scanner;

public class AscendingOrder {
    public static void main(String[] args) {
        int n, temp;
        Scanner s = new Scanner(System.in);
        System.out.print("Enter no. of elements you want in array: ");
        n = s.nextInt();
        int a[] = new int[n];
        System.out.println("Enter all the elements:");
        for (int i = 0; i < n; i++) {
            a[i] = s.nextInt();
        }
        for (int i = 0; i < n; i++) {
            for (int j = i + 1; j < n; j++) {
                if (a[i] > a[j]) {
                    temp = a[i];
                    a[i] = a[j];
                    a[j] = temp;
                }
            }
        }
        System.out.print("Ascending Order: ");
        for (int i = 0; i < n - 1; i++) {
            System.out.print(a[i] + ", ");
        }
        System.out.print(a[n - 1]);
    }
}
```

9.Stack Implementation

1. How many errors are there in the program? Mention the errors you have identified.

- Error 1: In the push method, the line top-- should be top++. This is incorrect because it moves the top pointer down instead of up.
- Error 2: In the pop method, the line top++ should be top--. This correctly moves the top pointer down when popping an element.
- Error 3: In the display method, the loop condition should be i <= top instead of i > top. This ensures that all elements in the stack are displayed correctly.

2. How many breakpoints do you need to fix those errors?

- Total Breakpoints Needed: 4

Breakpoints:

- Breakpoint 1: In the push method to check the value of top before and after pushing an element.
- Breakpoint 2: In the pop method to verify the value of top before and after popping an element.
- Breakpoint 3: In the display method to inspect the values of the stack being displayed.
- Breakpoint 4: After all operations in main to check the final state of the stack.

a. What are the steps you have taken to fix the errors you identified in the code fragment?

1. **Push Method Correction**: Changed top-- to top++ in the push method.
2. **Pop Method Correction**: Changed top++ to top-- in the pop method.
3. **Display Method Loop Correction**: Updated the loop condition in the display method to i <= top.

3. Submit your complete executable code.

```
// Stack implementation in java
import java.util.Arrays;

public class StackMethods {
    private int top;
    int size;
    int[] stack;

    public StackMethods(int arraySize) {
        size = arraySize;
        stack = new int[size];
```

```
        top = -1;
    }

    public void push(int value) {
        if (top == size - 1) {
            System.out.println("Stack is full, can't push a value");
        } else {
            top++;
            stack[top] = value;
        }
    }

    public void pop() {
        if (!isEmpty()) {
            top--;
        } else {
            System.out.println("Can't pop...stack is empty");
        }
    }

    public boolean isEmpty() {
        return top == -1;
    }

    public void display() {
        for (int i = 0; i <= top; i++) {
            System.out.print(stack[i] + " ");
        }
        System.out.println();
    }
}

public class StackReviseDemo {
    public static void main(String[] args) {
        StackMethods newStack = new StackMethods(5);
        newStack.push(10);
        newStack.push(1);
        newStack.push(50);
        newStack.push(20);
        newStack.push(90);

        newStack.display();
        newStack.pop();
```

```
        newStack.pop();
        newStack.pop();
        newStack.pop();
        newStack.display();
    }
}
```

10 Tower of Hanoi

1. How many errors are there in the program? Mention the errors you have identified.

- **Error 1**: In the recursive call doTowers(topN ++, inter--, from+1, to+1), the increment and decrement operators are incorrectly used. The parameters should be topN - 1 and inter, without any increment or decrement.
- **Error 2**: The parameters from+1 and to+1 do not work as intended because from and to are characters, and adding 1 to a character will not produce valid tower names. They should remain unchanged.

2. How many breakpoints do you need to fix those errors?

- **Total Breakpoints Needed**: 3

Breakpoints:

- **Breakpoint 1**: Before the recursive calls to inspect the values of topN, from, inter, and to.
- **Breakpoint 2**: After the first recursive call to ensure the correct sequence of disk movements is maintained.
- **Breakpoint 3**: Before printing the movement of disks to verify the values being passed.

a. What are the steps you have taken to fix the errors you identified in the code fragment?

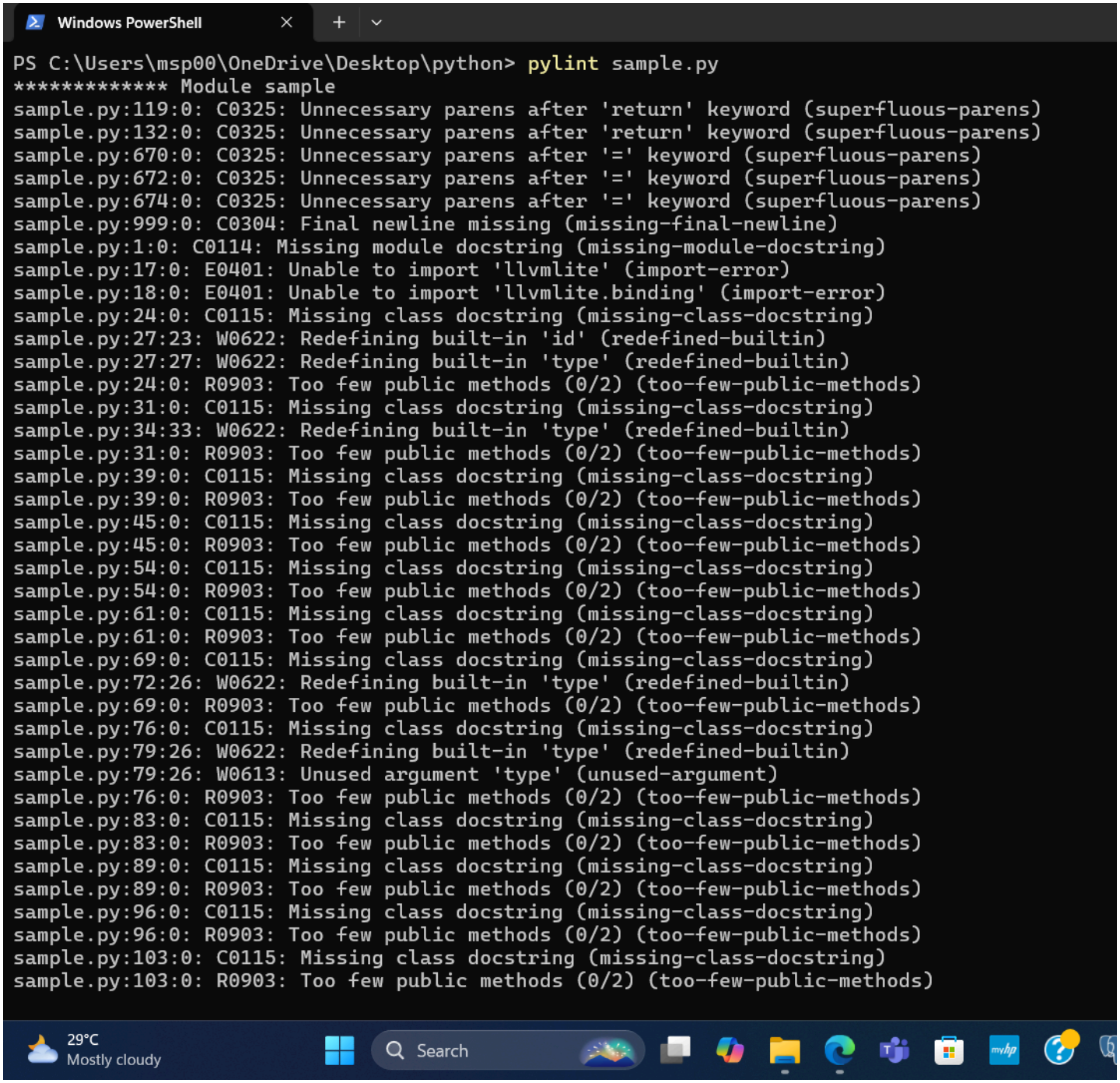
1. **Corrected Recursive Calls**: Changed doTowers(topN ++, inter--, from+1, to+1) to doTowers(topN - 1, from, inter, to) and fixed the subsequent recursive calls.
2. **Kept Character Parameters Unchanged**: Removed +1 from from and to parameters to maintain the character representation correctly.

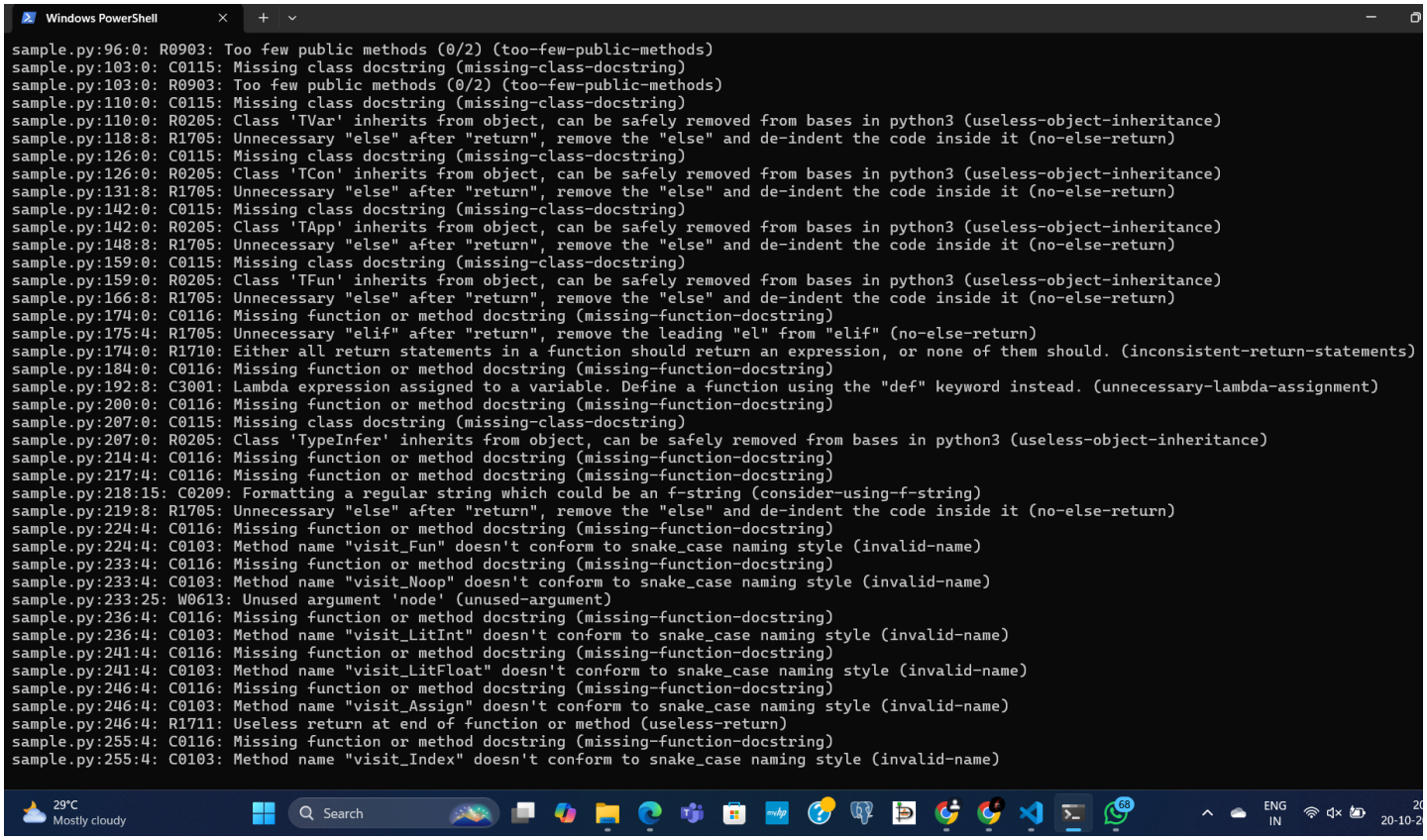
3. Submit your complete executable code.

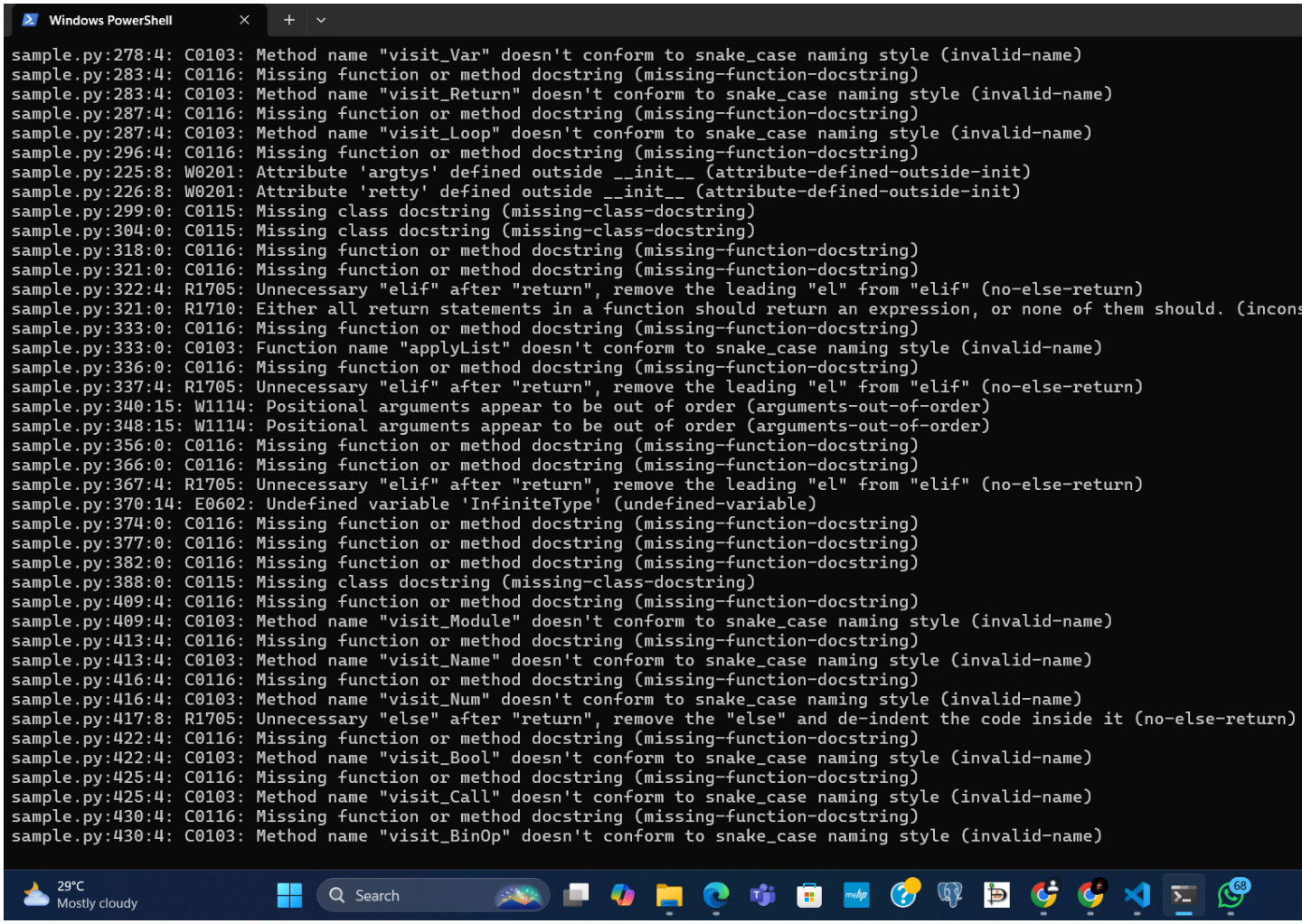
```java
// Tower of Hanoi
public class MainClass {
    public static void main(String[] args) {
        int nDisks = 3;
        doTowers(nDisks, 'A', 'B', 'C');
    }

    public static void doTowers(int topN, char from, char inter, char to) {
        if (topN == 1) {
            System.out.println("Disk 1 from " + from + " to " + to);
        } else {
            doTowers(topN - 1, from, to, inter);
            System.out.println("Disk " + topN + " from " + from + " to " +
to);
            doTowers(topN - 1, inter, from, to);
        }
    }
}
```

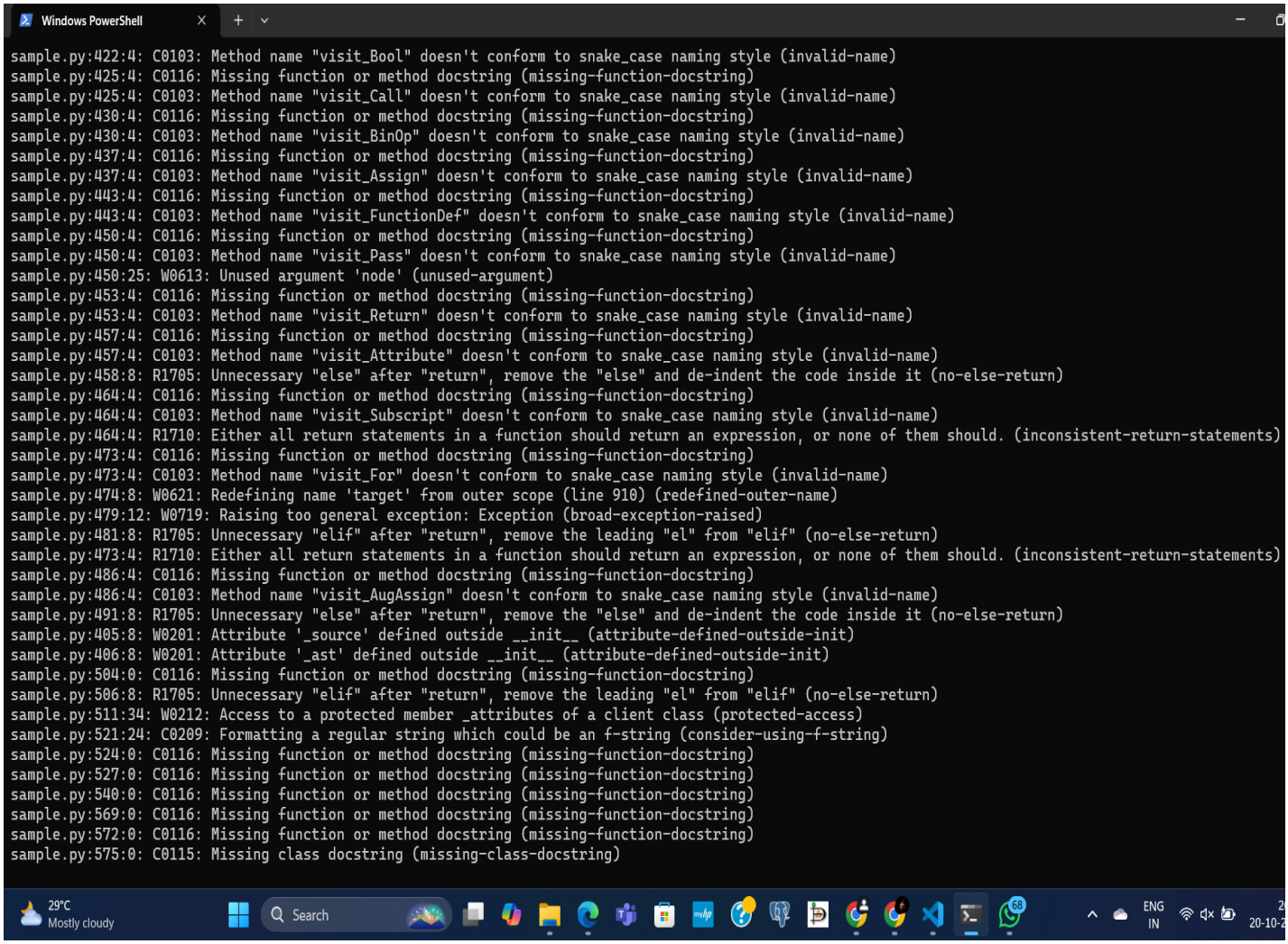
Static Analysis Tools
Choose a static analysis tool (in Java, Python, C, C++) in any programming language of your interest and identify the defects.
You can also choose your own code fragment from GitHub (more than 2000 LOC) in any programming language to perform static analysis.
Submit your results in the .xls or .jpg format only.
ANS:

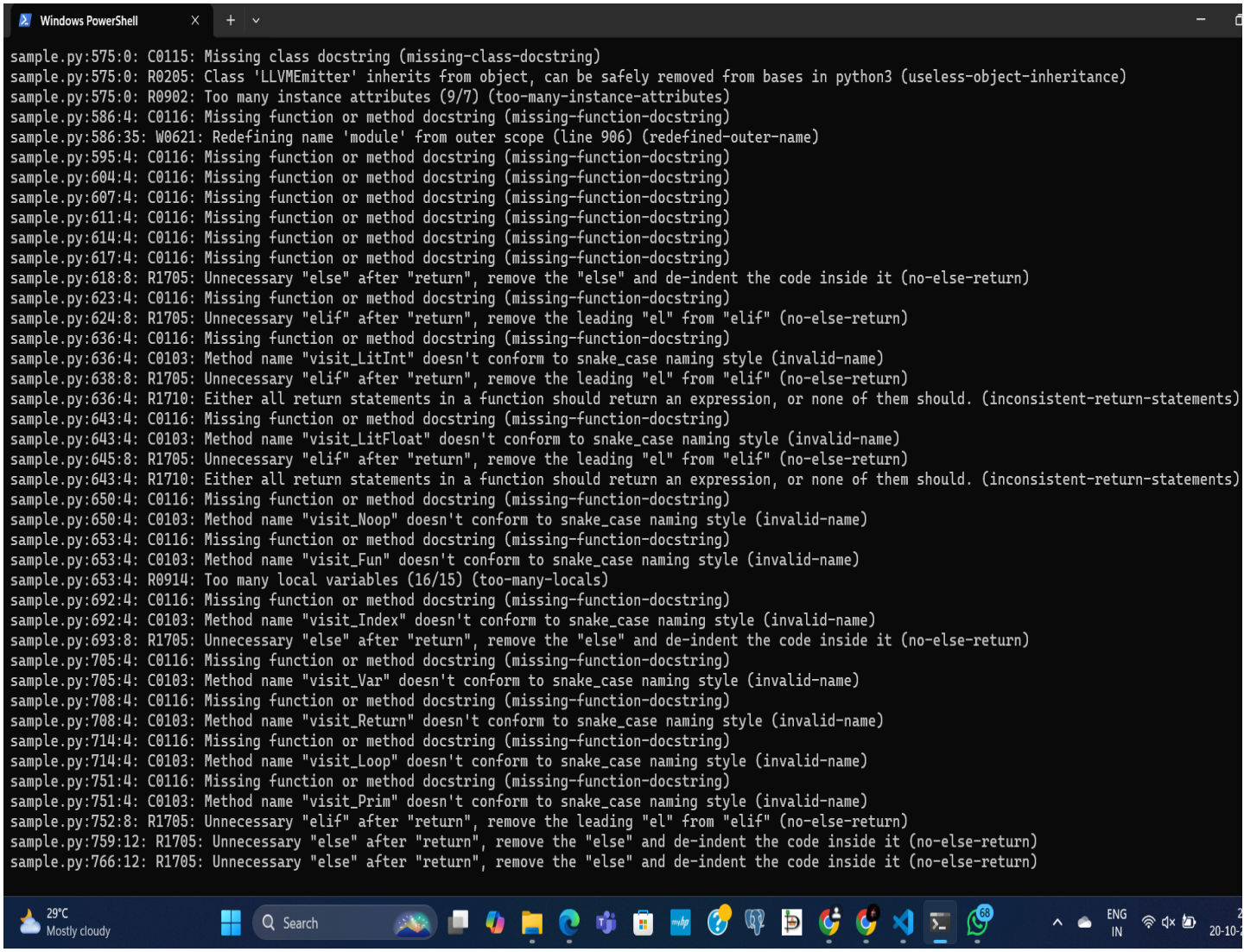
Code taken from : https://github.com/sdiehl/numpile/blob/master/numpile.py

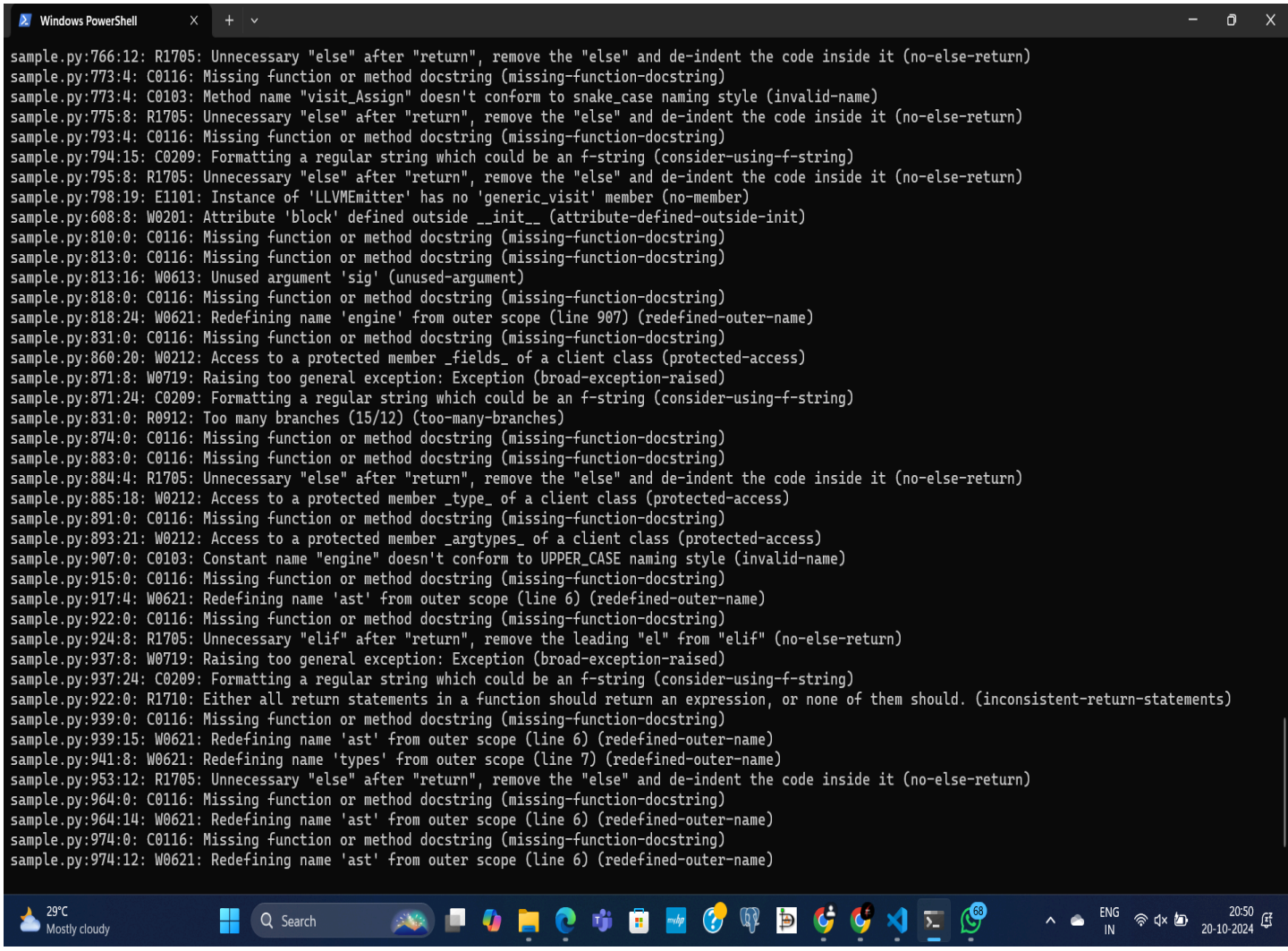

Windows PowerShell
PS C:\Users\msp00\OneDrive\Desktop\python> pylint sample.py
************* Module sample
sample.py:119:0: C0325: Unnecessary parens after 'return' keyword (superfluous-parens)
sample.py:132:0: C0325: Unnecessary parens after 'return' keyword (superfluous-parens)
sample.py:670:0: C0325: Unnecessary parens after '=' keyword (superfluous-parens)
sample.py:672:0: C0325: Unnecessary parens after '=' keyword (superfluous-parens)
sample.py:674:0: C0325: Unnecessary parens after '=' keyword (superfluous-parens)
sample.py:999:0: C0304: Final newline missing (missing-final-newline)
sample.py:1:0: C0114: Missing module docstring (missing-module-docstring)
sample.py:17:0: E0401: Unable to import 'llvmlite' (import-error)
sample.py:18:0: E0401: Unable to import 'llvmlite.binding' (import-error)
sample.py:24:0: C0115: Missing class docstring (missing-class-docstring)
sample.py:27:23: W0622: Redefining built-in 'id' (redefined-builtin)
sample.py:27:27: W0622: Redefining built-in 'type' (redefined-builtin)
sample.py:24:0: R0903: Too few public methods (0/2) (too-few-public-methods)
sample.py:31:0: C0115: Missing class docstring (missing-class-docstring)
sample.py:34:33: W0622: Redefining built-in 'type' (redefined-builtin)
sample.py:31:0: R0903: Too few public methods (0/2) (too-few-public-methods)
sample.py:39:0: C0115: Missing class docstring (missing-class-docstring)
sample.py:39:0: R0903: Too few public methods (0/2) (too-few-public-methods)
sample.py:45:0: C0115: Missing class docstring (missing-class-docstring)
sample.py:45:0: R0903: Too few public methods (0/2) (too-few-public-methods)
sample.py:54:0: C0115: Missing class docstring (missing-class-docstring)
sample.py:54:0: R0903: Too few public methods (0/2) (too-few-public-methods)
sample.py:61:0: C0115: Missing class docstring (missing-class-docstring)
sample.py:61:0: R0903: Too few public methods (0/2) (too-few-public-methods)
sample.py:69:0: C0115: Missing class docstring (missing-class-docstring)
sample.py:72:26: W0622: Redefining built-in 'type' (redefined-builtin)
sample.py:69:0: R0903: Too few public methods (0/2) (too-few-public-methods)
sample.py:76:0: C0115: Missing class docstring (missing-class-docstring)
sample.py:79:26: W0622: Redefining built-in 'type' (redefined-builtin)
sample.py:79:26: W0613: Unused argument 'type' (unused-argument)
sample.py:76:0: R0903: Too few public methods (0/2) (too-few-public-methods)
sample.py:83:0: C0115: Missing class docstring (missing-class-docstring)
sample.py:83:0: R0903: Too few public methods (0/2) (too-few-public-methods)
sample.py:89:0: C0115: Missing class docstring (missing-class-docstring)
sample.py:89:0: R0903: Too few public methods (0/2) (too-few-public-methods)
sample.py:96:0: C0115: Missing class docstring (missing-class-docstring)
sample.py:96:0: R0903: Too few public methods (0/2) (too-few-public-methods)
sample.py:103:0: C0115: Missing class docstring (missing-class-docstring)
sample.py:103:0: R0903: Too few public methods (0/2) (too-few-public-methods)
29°C
Mostly cloudy
Search

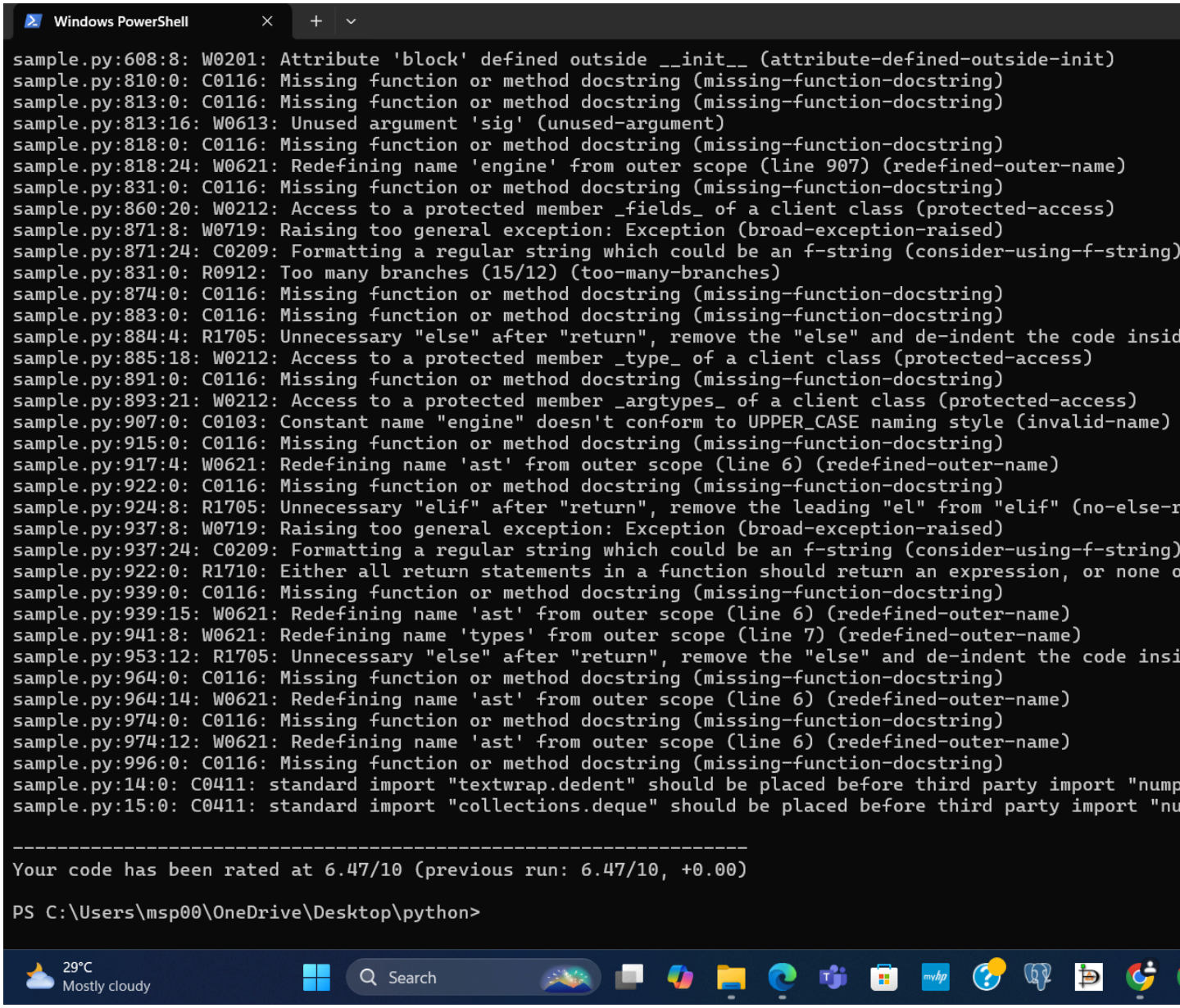

Windows PowerShell
sample.py:96:0: R0903: Too few public methods (0/2) (too-few-public-methods)
sample.py:103:0: C0115: Missing class docstring (missing-class-docstring)
sample.py:103:0: R0903: Too few public methods (0/2) (too-few-public-methods)
sample.py:110:0: C0115: Missing class docstring (missing-class-docstring)
sample.py:110:0: R0205: Class 'TVar' inherits from object, can be safely removed from bases in python3 (useless-object-inheritance)
sample.py:118:8: R1705: Unnecessary "else" after "return", remove the "else" and de-indent the code inside it (no-else-return)
sample.py:126:0: C0115: Missing class docstring (missing-class-docstring)
sample.py:126:0: R0205: Class 'TCon' inherits from object, can be safely removed from bases in python3 (useless-object-inheritance)
sample.py:131:8: R1705: Unnecessary "else" after "return", remove the "else" and de-indent the code inside it (no-else-return)
sample.py:142:0: C0115: Missing class docstring (missing-class-docstring)
sample.py:142:0: R0205: Class 'TApp' inherits from object, can be safely removed from bases in python3 (useless-object-inheritance)
sample.py:148:8: R1705: Unnecessary "else" after "return", remove the "else" and de-indent the code inside it (no-else-return)
sample.py:159:0: C0115: Missing class docstring (missing-class-docstring)
sample.py:159:0: R0205: Class 'TFun' inherits from object, can be safely removed from bases in python3 (useless-object-inheritance)
sample.py:166:8: R1705: Unnecessary "else" after "return", remove the "else" and de-indent the code inside it (no-else-return)
sample.py:174:0: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:175:4: R1705: Unnecessary "elif" after "return", remove the leading "el" from "elif" (no-else-return)
sample.py:174:0: R1710: Either all return statements in a function should return an expression, or none of them should. (inconsistent-return-statements)
sample.py:184:0: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:192:8: C3001: Lambda expression assigned to a variable. Define a function using the "def" keyword instead. (unnecessary-lambda-assignment)
sample.py:200:0: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:207:0: C0115: Missing class docstring (missing-class-docstring)
sample.py:207:0: R0205: Class 'TypeInfer' inherits from object, can be safely removed from bases in python3 (useless-object-inheritance)
sample.py:214:4: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:217:4: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:218:15: C0209: Formatting a regular string which could be an f-string (consider-using-f-string)
sample.py:219:8: R1705: Unnecessary "else" after "return", remove the "else" and de-indent the code inside it (no-else-return)
sample.py:224:4: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:224:4: C0103: Method name "visit_Fun" doesn't conform to snake_case naming style (invalid-name)
sample.py:233:4: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:233:4: C0103: Method name "visit_Noop" doesn't conform to snake_case naming style (invalid-name)
sample.py:233:25: W0613: Unused argument 'node' (unused-argument)
sample.py:236:4: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:236:4: C0103: Method name "visit_LitInt" doesn't conform to snake_case naming style (invalid-name)
sample.py:241:4: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:241:4: C0103: Method name "visit_LitFloat" doesn't conform to snake_case naming style (invalid-name)
sample.py:246:4: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:246:4: C0103: Method name "visit_Assign" doesn't conform to snake_case naming style (invalid-name)
sample.py:246:4: R1711: Useless return at end of function or method (useless-return)
sample.py:255:4: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:255:4: C0103: Method name "visit_Index" doesn't conform to snake_case naming style (invalid-name)
29°C
Mostly cloudy
Search
ENG
IN
20-10-2


Windows PowerShell
sample.py:278:4: C0103: Method name "visit_Var" doesn't conform to snake_case naming style (invalid-name)
sample.py:283:4: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:283:4: C0103: Method name "visit_Return" doesn't conform to snake_case naming style (invalid-name)
sample.py:287:4: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:287:4: C0103: Method name "visit_Loop" doesn't conform to snake_case naming style (invalid-name)
sample.py:296:4: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:225:8: W0201: Attribute 'argtys' defined outside __init__ (attribute-defined-outside-init)
sample.py:226:8: W0201: Attribute 'retty' defined outside __init__ (attribute-defined-outside-init)
sample.py:299:0: C0115: Missing class docstring (missing-class-docstring)
sample.py:304:0: C0115: Missing class docstring (missing-class-docstring)
sample.py:318:0: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:321:0: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:322:4: R1705: Unnecessary "elif" after "return", remove the leading "el" from "elif" (no-else-return)
sample.py:321:0: R1710: Either all return statements in a function should return an expression, or none of them should. (incons
sample.py:333:0: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:333:0: C0103: Function name "applyList" doesn't conform to snake_case naming style (invalid-name)
sample.py:336:0: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:337:4: R1705: Unnecessary "elif" after "return", remove the leading "el" from "elif" (no-else-return)
sample.py:340:15: W1114: Positional arguments appear to be out of order (arguments-out-of-order)
sample.py:348:15: W1114: Positional arguments appear to be out of order (arguments-out-of-order)
sample.py:356:0: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:366:0: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:367:4: R1705: Unnecessary "elif" after "return", remove the leading "el" from "elif" (no-else-return)
sample.py:370:14: E0602: Undefined variable 'InfiniteType' (undefined-variable)
sample.py:374:0: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:377:0: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:382:0: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:388:0: C0115: Missing class docstring (missing-class-docstring)
sample.py:409:4: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:409:4: C0103: Method name "visit_Module" doesn't conform to snake_case naming style (invalid-name)
sample.py:413:4: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:413:4: C0103: Method name "visit_Name" doesn't conform to snake_case naming style (invalid-name)
sample.py:416:4: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:416:4: C0103: Method name "visit_Num" doesn't conform to snake_case naming style (invalid-name)
sample.py:417:8: R1705: Unnecessary "else" after "return", remove the "else" and de-indent the code inside it (no-else-return)
sample.py:422:4: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:422:4: C0103: Method name "visit_Bool" doesn't conform to snake_case naming style (invalid-name)
sample.py:425:4: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:425:4: C0103: Method name "visit_Call" doesn't conform to snake_case naming style (invalid-name)
sample.py:430:4: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:430:4: C0103: Method name "visit_BinOp" doesn't conform to snake_case naming style (invalid-name)
29°C
Mostly cloudy
Search


Windows PowerShell
sample.py:422:4: C0103: Method name "visit_Bool" doesn't conform to snake_case naming style (invalid-name)
sample.py:425:4: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:425:4: C0103: Method name "visit_Call" doesn't conform to snake_case naming style (invalid-name)
sample.py:430:4: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:430:4: C0103: Method name "visit_BinOp" doesn't conform to snake_case naming style (invalid-name)
sample.py:437:4: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:437:4: C0103: Method name "visit_Assign" doesn't conform to snake_case naming style (invalid-name)
sample.py:443:4: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:443:4: C0103: Method name "visit_FunctionDef" doesn't conform to snake_case naming style (invalid-name)
sample.py:450:4: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:450:4: C0103: Method name "visit_Pass" doesn't conform to snake_case naming style (invalid-name)
sample.py:450:25: W0613: Unused argument 'node' (unused-argument)
sample.py:453:4: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:453:4: C0103: Method name "visit_Return" doesn't conform to snake_case naming style (invalid-name)
sample.py:457:4: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:457:4: C0103: Method name "visit_Attribute" doesn't conform to snake_case naming style (invalid-name)
sample.py:458:8: R1705: Unnecessary "else" after "return", remove the "else" and de-indent the code inside it (no-else-return)
sample.py:464:4: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:464:4: C0103: Method name "visit_Subscript" doesn't conform to snake_case naming style (invalid-name)
sample.py:464:4: R1710: Either all return statements in a function should return an expression, or none of them should. (inconsistent-return-statements)
sample.py:473:4: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:473:4: C0103: Method name "visit_For" doesn't conform to snake_case naming style (invalid-name)
sample.py:474:8: W0621: Redefining name 'target' from outer scope (line 910) (redefined-outer-name)
sample.py:479:12: W0719: Raising too general exception: Exception (broad-exception-raised)
sample.py:481:8: R1705: Unnecessary "elif" after "return", remove the leading "el" from "elif" (no-else-return)
sample.py:473:4: R1710: Either all return statements in a function should return an expression, or none of them should. (inconsistent-return-statements)
sample.py:486:4: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:486:4: C0103: Method name "visit_AugAssign" doesn't conform to snake_case naming style (invalid-name)
sample.py:491:8: R1705: Unnecessary "else" after "return", remove the "else" and de-indent the code inside it (no-else-return)
sample.py:405:8: W0201: Attribute '_source' defined outside __init__ (attribute-defined-outside-init)
sample.py:406:8: W0201: Attribute '_ast' defined outside __init__ (attribute-defined-outside-init)
sample.py:504:0: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:506:8: R1705: Unnecessary "elif" after "return", remove the leading "el" from "elif" (no-else-return)
sample.py:511:34: W0212: Access to a protected member _attributes of a client class (protected-access)
sample.py:521:24: C0209: Formatting a regular string which could be an f-string (consider-using-f-string)
sample.py:524:0: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:527:0: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:540:0: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:569:0: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:572:0: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:575:0: C0115: Missing class docstring (missing-class-docstring)
29°C
Mostly cloudy
Search
ENG
IN
20-10-2


Windows PowerShell
sample.py:575:0: C0115: Missing class docstring (missing-class-docstring)
sample.py:575:0: R0205: Class 'LLVMEmitter' inherits from object, can be safely removed from bases in python3 (useless-object-inheritance)
sample.py:575:0: R0902: Too many instance attributes (9/7) (too-many-instance-attributes)
sample.py:586:4: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:586:35: W0621: Redefining name 'module' from outer scope (line 906) (redefined-outer-name)
sample.py:595:4: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:604:4: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:607:4: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:611:4: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:614:4: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:617:4: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:618:8: R1705: Unnecessary "else" after "return", remove the "else" and de-indent the code inside it (no-else-return)
sample.py:623:4: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:624:8: R1705: Unnecessary "elif" after "return", remove the leading "el" from "elif" (no-else-return)
sample.py:636:4: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:636:4: C0103: Method name "visit_LitInt" doesn't conform to snake_case naming style (invalid-name)
sample.py:638:8: R1705: Unnecessary "elif" after "return", remove the leading "el" from "elif" (no-else-return)
sample.py:636:4: R1710: Either all return statements in a function should return an expression, or none of them should. (inconsistent-return-statements)
sample.py:643:4: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:643:4: C0103: Method name "visit_LitFloat" doesn't conform to snake_case naming style (invalid-name)
sample.py:645:8: R1705: Unnecessary "elif" after "return", remove the leading "el" from "elif" (no-else-return)
sample.py:643:4: R1710: Either all return statements in a function should return an expression, or none of them should. (inconsistent-return-statements)
sample.py:650:4: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:650:4: C0103: Method name "visit_Noop" doesn't conform to snake_case naming style (invalid-name)
sample.py:653:4: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:653:4: C0103: Method name "visit_Fun" doesn't conform to snake_case naming style (invalid-name)
sample.py:653:4: R0914: Too many local variables (16/15) (too-many-locals)
sample.py:692:4: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:692:4: C0103: Method name "visit_Index" doesn't conform to snake_case naming style (invalid-name)
sample.py:693:8: R1705: Unnecessary "else" after "return", remove the "else" and de-indent the code inside it (no-else-return)
sample.py:705:4: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:705:4: C0103: Method name "visit_Var" doesn't conform to snake_case naming style (invalid-name)
sample.py:708:4: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:708:4: C0103: Method name "visit_Return" doesn't conform to snake_case naming style (invalid-name)
sample.py:714:4: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:714:4: C0103: Method name "visit_Loop" doesn't conform to snake_case naming style (invalid-name)
sample.py:751:4: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:751:4: C0103: Method name "visit_Prim" doesn't conform to snake_case naming style (invalid-name)
sample.py:752:8: R1705: Unnecessary "elif" after "return", remove the leading "el" from "elif" (no-else-return)
sample.py:759:12: R1705: Unnecessary "else" after "return", remove the "else" and de-indent the code inside it (no-else-return)
sample.py:766:12: R1705: Unnecessary "else" after "return", remove the "else" and de-indent the code inside it (no-else-return)
29°C
Mostly cloudy
Search
ENG
IN


Windows PowerShell
sample.py:766:12: R1705: Unnecessary "else" after "return", remove the "else" and de-indent the code inside it (no-else-return)
sample.py:773:4: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:773:4: C0103: Method name "visit_Assign" doesn't conform to snake_case naming style (invalid-name)
sample.py:775:8: R1705: Unnecessary "else" after "return", remove the "else" and de-indent the code inside it (no-else-return)
sample.py:793:4: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:794:15: C0209: Formatting a regular string which could be an f-string (consider-using-f-string)
sample.py:795:8: R1705: Unnecessary "else" after "return", remove the "else" and de-indent the code inside it (no-else-return)
sample.py:798:19: E1101: Instance of 'LLVMEmitter' has no 'generic_visit' member (no-member)
sample.py:608:8: W0201: Attribute 'block' defined outside __init__ (attribute-defined-outside-init)
sample.py:810:0: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:813:0: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:813:16: W0613: Unused argument 'sig' (unused-argument)
sample.py:818:0: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:818:24: W0621: Redefining name 'engine' from outer scope (line 907) (redefined-outer-name)
sample.py:831:0: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:860:20: W0212: Access to a protected member _fields_ of a client class (protected-access)
sample.py:871:8: W0719: Raising too general exception: Exception (broad-exception-raised)
sample.py:871:24: C0209: Formatting a regular string which could be an f-string (consider-using-f-string)
sample.py:831:0: R0912: Too many branches (15/12) (too-many-branches)
sample.py:874:0: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:883:0: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:884:4: R1705: Unnecessary "else" after "return", remove the "else" and de-indent the code inside it (no-else-return)
sample.py:885:18: W0212: Access to a protected member _type_ of a client class (protected-access)
sample.py:891:0: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:893:21: W0212: Access to a protected member _argtypes_ of a client class (protected-access)
sample.py:907:0: C0103: Constant name "engine" doesn't conform to UPPER_CASE naming style (invalid-name)
sample.py:915:0: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:917:4: W0621: Redefining name 'ast' from outer scope (line 6) (redefined-outer-name)
sample.py:922:0: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:924:8: R1705: Unnecessary "elif" after "return", remove the leading "el" from "elif" (no-else-return)
sample.py:937:8: W0719: Raising too general exception: Exception (broad-exception-raised)
sample.py:937:24: C0209: Formatting a regular string which could be an f-string (consider-using-f-string)
sample.py:922:0: R1710: Either all return statements in a function should return an expression, or none of them should. (inconsistent-return-statements)
sample.py:939:0: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:939:15: W0621: Redefining name 'ast' from outer scope (line 6) (redefined-outer-name)
sample.py:941:8: W0621: Redefining name 'types' from outer scope (line 7) (redefined-outer-name)
sample.py:953:12: R1705: Unnecessary "else" after "return", remove the "else" and de-indent the code inside it (no-else-return)
sample.py:964:0: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:964:14: W0621: Redefining name 'ast' from outer scope (line 6) (redefined-outer-name)
sample.py:974:0: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:974:12: W0621: Redefining name 'ast' from outer scope (line 6) (redefined-outer-name)
29°C
Mostly cloudy
Search
ENG
IN
20:50
20-10-2024


Windows PowerShell
sample.py:608:8: W0201: Attribute 'block' defined outside __init__ (attribute-defined-outside-init)
sample.py:810:0: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:813:0: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:813:16: W0613: Unused argument 'sig' (unused-argument)
sample.py:818:0: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:818:24: W0621: Redefining name 'engine' from outer scope (line 907) (redefined-outer-name)
sample.py:831:0: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:860:20: W0212: Access to a protected member _fields_ of a client class (protected-access)
sample.py:871:8: W0719: Raising too general exception: Exception (broad-exception-raised)
sample.py:871:24: C0209: Formatting a regular string which could be an f-string (consider-using-f-string)
sample.py:831:0: R0912: Too many branches (15/12) (too-many-branches)
sample.py:874:0: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:883:0: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:884:4: R1705: Unnecessary "else" after "return", remove the "else" and de-indent the code insid
sample.py:885:18: W0212: Access to a protected member _type_ of a client class (protected-access)
sample.py:891:0: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:893:21: W0212: Access to a protected member _argtypes_ of a client class (protected-access)
sample.py:907:0: C0103: Constant name "engine" doesn't conform to UPPER_CASE naming style (invalid-name)
sample.py:915:0: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:917:4: W0621: Redefining name 'ast' from outer scope (line 6) (redefined-outer-name)
sample.py:922:0: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:924:8: R1705: Unnecessary "elif" after "return", remove the leading "el" from "elif" (no-else-r
sample.py:937:8: W0719: Raising too general exception: Exception (broad-exception-raised)
sample.py:937:24: C0209: Formatting a regular string which could be an f-string (consider-using-f-string)
sample.py:922:0: R1710: Either all return statements in a function should return an expression, or none o
sample.py:939:0: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:939:15: W0621: Redefining name 'ast' from outer scope (line 6) (redefined-outer-name)
sample.py:941:8: W0621: Redefining name 'types' from outer scope (line 7) (redefined-outer-name)
sample.py:953:12: R1705: Unnecessary "else" after "return", remove the "else" and de-indent the code insi
sample.py:964:0: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:964:14: W0621: Redefining name 'ast' from outer scope (line 6) (redefined-outer-name)
sample.py:974:0: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:974:12: W0621: Redefining name 'ast' from outer scope (line 6) (redefined-outer-name)
sample.py:996:0: C0116: Missing function or method docstring (missing-function-docstring)
sample.py:14:0: C0411: standard import "textwrap.dedent" should be placed before third party import "nump
sample.py:15:0: C0411: standard import "collections.deque" should be placed before third party import "nu
Your code has been rated at 6.47/10 (previous run: 6.47/10, +0.00)
PS C:\Users\msp00\OneDrive\Desktop\python>
29°C
Mostly cloudy
Search