

AI ASSISTED CODING LAB

ASSIGNMENT 13.2

ENROLLMENT NO :2503A51L39

BATCH NO: 20

NAME: Shaik faheem

TASK1

TASK1 DESCRIPTION:- Provide AI with the following redundant code and ask it to refactor

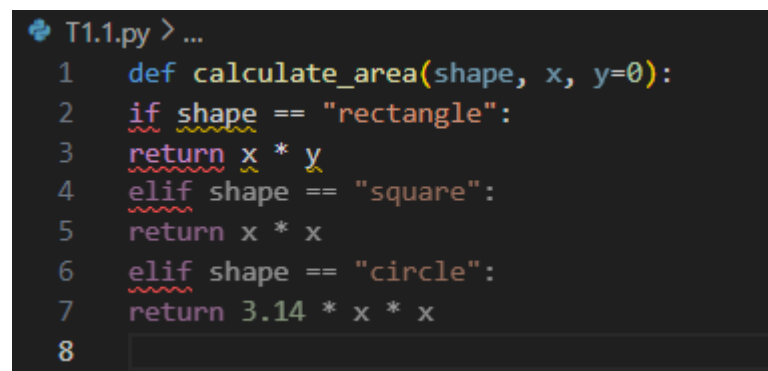
Python Code:-

```
def calculate_area(shape, x, y=0):  
    if shape == "rectangle":  
        return x * y  
    elif shape == "square":  
        return x * x  
    elif shape == "circle":  
        return 3.14 * x * x
```

PROMPT:- Refactor the redundant code for calculating areas of shapes.

CODE:-

ERROR CODE:-

A screenshot of a code editor window titled 'T1.1.py > ...'. The editor contains the following Python code:

```
1  def calculate_area(shape, x, y=0):  
2  if shape == "rectangle":  
3  return x * y  
4  elif shape == "square":  
5  return x * x  
6  elif shape == "circle":  
7  return 3.14 * x * x  
8
```

The code is displayed with syntax highlighting: keywords like 'def', 'if', 'elif', 'return' are in blue, strings are in red, and variables/numbers are in green. The line numbers 1 through 8 are visible on the left side of the editor.

OUTPUT:-

```
if shape == "rectangle":  
^^
```

AI GENERATED CODE:-

```
t1.py > ...  
1  import math  
2  
3  def calculate_area(shape, x, y=None):  
4      try:  
5          shape = shape.lower()  
6          if shape == "rectangle":  
7              if y is None:  
8                  raise ValueError("Rectangle requires two dimensions (x and y).")  
9              return x * y  
10             elif shape == "square":  
11                 return math.pow(x, 2)  
12             elif shape == "circle":  
13                 return math.pi * math.pow(x, 2)  
14             else:  
15                 raise ValueError(f"Unknown shape: {shape}")  
16         except TypeError:  
17             print("Error: Invalid type for dimensions. Please provide numbers.")  
18             return None  
19         except ValueError as ve:  
20             print(f"Error: {ve}")  
21             return None  
22  
23     # Example usage:  
24     if __name__ == "__main__":  
25         print("Rectangle area (5, 3):", calculate_area("rectangle", 5, 3))  
26         print("Square area (4):", calculate_area("square", 4))  
27         print("Circle area (radius 2):", calculate_area("circle", 2))  
28         print("Invalid shape:", calculate_area("triangle", 5, 6))  
29         print("Missing y for rectangle:", calculate_area("rectangle", 5))  
30         print("Invalid type:", calculate_area("square", "four"))
```

OUTPUT:-

```
Error: Rectangle requires two dimensions (x and y).  
Missing y for rectangle: None  
Error: Invalid type for dimensions. Please provide numbers.  
Invalid type: None
```

OBSERVATION:-

The AI refactored the function, improved readability, and added error handling, making the

code more reliable.

TASK2

TASK2 DESCRIPTION:- Error Handling in Legacy Code

Task: Legacy function without proper error handling

Python Code

```
def read_file(filename):
```

```
    f = open(filename, "r")
```

```
    data = f.read()
```

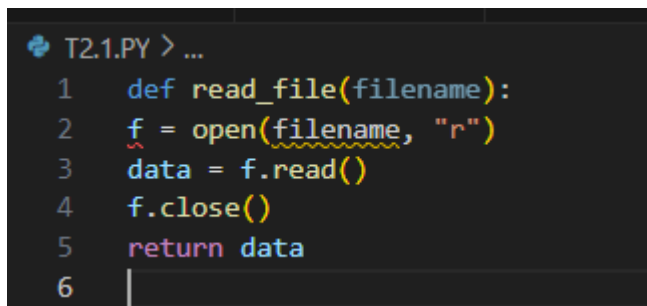
```
    f.close()
```

```
    return data
```

PROMPT:- Improve the legacy file reading function with error handling.

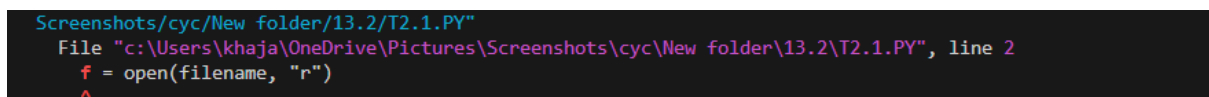
CODE:-

ERROR CODE:-



```
T2.1.PY > ...
1  def read_file(filename):
2  f = open(filename, "r")
3  data = f.read()
4  f.close()
5  return data
6  |
```

OUTPUT:-



```
Screenshots/cyc/New folder/13.2/T2.1.PY"
File "c:\Users\khaja\OneDrive\Pictures\Screenshots\cyc\New folder\13.2\T2.1.PY", line 2
f = open(filename, "r")
^
```

AI GENERATED CODE:-

```

t2.py > ...
1  def read_file(filename):
2      try:
3          with open(filename, "r") as f:
4              data = f.read()
5              return data
6      except FileNotFoundError:
7          print(f"Error: The file '{filename}' was not found.")
8          return None
9      except IOError as e:
10         print(f"IO error occurred while reading '{filename}': {e}")
11         return None
12 print(read_file("examples.txt"))

```

OUTPUT:-

```

Error: The file 'examples.txt' was not found.
None

```

OBSERVATION:- The AI added exception handling for missing files and ensured safe file operations, preventing runtime errors.

TASK3

TASK3 DESCRIPTION:- Complex Refactoring

Task: Provide this legacy class to AI for readability and modularity improvements:

Python Code

class Student:

def __init__(self, n, a, m1, m2, m3):

self.n = n

self.a = a

self.m1 = m1

self.m2 = m2

self.m3 = m3

def details(self):

print("Name:", self.n, "Age:", self.a)

def total(self):

return self.m1+self.m2+self.m3

PROMPT:- Refactor the legacy Student class for readability and modularity.

CODE:-

ERROR CODE:-

```
T3.1.PY > ...
1  class Student:
2  def __init__(self, n, a, m1, m2, m3):
3  self.n = n
4  self.a = a
5  self.m1 = m1
6  self.m2 = m2
7  self.m3 = m3
8  def details(self):
9  print("Name:", self.n, "Age:", self.a)
10 def total(self):
11 return self.m1+self.m2+self.m3
12
```

OUTPUT:-

```
def __init__(self, n, a, m1, m2, m3):
    """
```

AI GENERATED CODE:-

```

t3.py > Student > __init__
1  class Student:
2      """
3      Represents a student with name, age, and marks in three subjects.
4      """
5
6      def __init__(self, name, age, mark1, mark2, mark3):
7          """
8          Initialize a Student object.
9
10         Args:
11             name (str): The student's name.
12             age (int): The student's age.
13             mark1 (int/float): Marks in subject 1.
14             mark2 (int/float): Marks in subject 2.
15             mark3 (int/float): Marks in subject 3.
16         """
17         self.name = name
18         self.age = age
19         self.marks = [mark1, mark2, mark3]
20
21     def details(self):
22         """Prints the student's name and age."""
23         print(f"Name: {self.name}, Age: {self.age}")
24
25     def total(self):
26         """Returns the total marks obtained by the student."""
27         return sum(self.marks)
28
29 # Example usage:
30 if __name__ == "__main__":
31     student = Student("Alice", 20, 85, 90, 88)
32     student.details()
33     print("Total Marks:", student.total())

```

OUTPUT:-

```

Name: Alice, Age: 20
Total Marks: 263

```

OBSERVATION:- The AI improved the class structure, added clear methods, and made the code more organized and maintainable.

TASK4

TASK4 DESCRIPTION:- Inefficient Loop Refactoring

Task: Refactor this inefficient loop with AI help

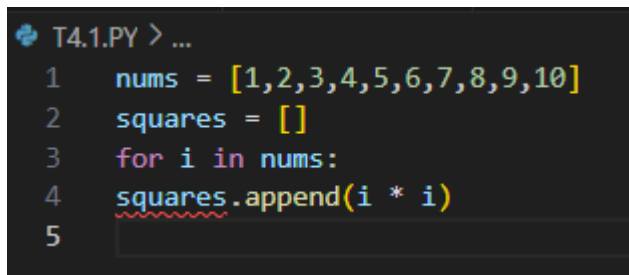
Python Code

```
nums = [1,2,3,4,5,6,7,8,9,10]
squares = []
for i in nums:
    squares.append(i * i)
```

PROMPT:- Refactor the inefficient loop for better performance.

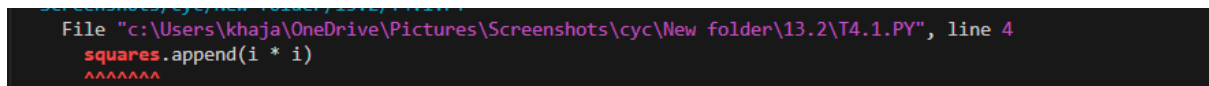
CODE:-

ERROR CODE:-



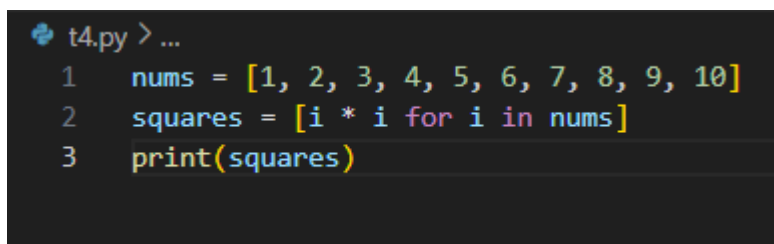
```
T4.1.PY > ...
1  nums = [1,2,3,4,5,6,7,8,9,10]
2  squares = []
3  for i in nums:
4  squares.append(i * i)
5
```

OUTPUT:-



```
File "c:\Users\khaja\OneDrive\Pictures\Screenshots\cyc\New folder\13.2\T4.1.PY", line 4
    squares.append(i * i)
    ^^^^^^^
```

AI GENERATED CODE:-



```
t4.py > ...
1  nums = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
2  squares = [i * i for i in nums]
3  print(squares)
```

OUTPUT:-



```
Screenshots/cyc/New folder/13.2/t4.py"
• [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

OBSERVATION:- The AI replaced the loop with a list comprehension, simplifying the code and improving efficiency.