

Name:T.RamyaSri

Hall.no:2303A52136

TASK-1:

The screenshot shows a code editor with a dark theme. A Python file named `add.py` is open, containing the following code:

```
def add(a,b) > add
1     def add(a,b)
2         return a + b
3
```

The cursor is positioned at the end of the first line of the function definition. The status bar at the bottom indicates there is 1 problem.

Below the code editor, the terminal window shows the following output:

```
SyntaxError: expected ':'
PS C:\Users\ramya\OneDrive\Desktop\coding> & C:/Users/ramya/AppData/Local,.11.exe "c:/Users/ramya/OneDrive/Desktop/coding/def add(a,b).py"
File "c:/Users/ramya/OneDrive/Desktop/coding/def add(a,b).py", line 1
    def add(a,b)
               ^
SyntaxError: expected ':'
PS C:\Users\ramya\OneDrive\Desktop\coding>
```

A modal dialog titled "Modify selected code" is displayed over the code editor. It contains the following text:

```
Modify selected code
✓ X
Add Context...
Auto ▾
1 def add(a,b):
2     return a + b
3
```

At the bottom right of the dialog are "Keep", "Undo", and a "Discard" button.

```
def add(a, b):
    return a + b
```

Tip:
Always end function definitions with a colon. This tells Python that the following indented lines belong to the function body.

TASK-2:

```
def add(a,b) > count_down
1 def count_down(n):
2     while n>=0:
3         print(n)
4         n+=1
```

Corrected code:

```
ef add(a,b).py > ↵ count_down
  def count_down(n):
    Modify selected code ➤
    ↵ Add Context... Auto
      while n>=0:
        print(n)
        n+=1
        ↴ -
```

Justification:

- Condition: `while n >= 0` → always true once `n` starts positive.
- Update: `n += 1` → makes `n` bigger, never less than 0.
- Result: infinite loop, output keeps going up: `5, 6, 7, 8...`.

TASK-3:

```
1 def divide(a,b):  
2     return a/b  
3  
4 print(divide(10,0))
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

POTS

```
PS C:\Users\ramya\OneDrive\Desktop\coding> & C:/Users/ramya/AppData/Local/WindowsApps/python3.11.exe "c:/Users/ramya/OneDrive/Desktop/coding\def add(a,b).py"  
le>  
    print(divide(10,0))  
          ^^^^^^^^^^  
File "c:\Users\ramya\OneDrive\Desktop\coding\def add(a,b).py", line 4  
    return a/b  
          ~~  
ZeroDivisionError: division by zero  
PS C:\Users\ramya\OneDrive\Desktop\coding> 
```

Corrected code:

```

❸ def add(a,b) > ...
1  def divide(a, b):
2      """
3          Safely divide a by b.
4          Returns the quotient when b != 0, otherwise returns None.
5      """
6      try:
7          return a / b
8      except ZeroDivisionError:
9          # Handle division by zero gracefully
10         print("Error: division by zero")
11         return None
12
13     # Demo runs
14     if __name__ == "__main__":
15         print("divide(10, 2) ->", divide(10, 2))
16         print("divide(10, 0) ->", divide(10, 0))

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

- ▶ PS C:\Users\ramya\OneDrive\Desktop\coding> & C:/Users/ramya/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/ramya/OneDrive/Desktop/coding/def add(a).py"

File "c:/Users/ramya/OneDrive/Desktop/coding/def add(a,b).py", line 7

 return new_func()
 ^^^^^^^^^^^^^^
- ▶ SyntaxError: 'return' outside function

PS C:\Users\ramya\OneDrive\Desktop\coding> & C:/Users/ramya/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/ramya/OneDrive/Desktop/coding/def add(a).py"

divide(10, 2) -> 5.0

Error: division by zero

divide(10, 0) -> None

PS C:\Users\ramya\OneDrive\Desktop\coding> []

Justification:

: The function tries to divide by zero, which is not allowed, so Python throws an error instead of producing output.

TASK-4:

```
def add(a,b).py > ...
1   class Rectangle:
2       def __init__(length,width):
3           self.length=length
4           self.width=width
5
```

PROBLEMS 7 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
''
File "c:\Users\ramya\OneDrive\Desktop\coding\def add(a,b)
$ C:\Users\ramya\OneDrive\Desktop\coding> & C:/Users/ramya
t/WindowsApps/python3.11.exe "c:/Users/ramya/OneDrive/Desktop
.py"
File "c:\Users\ramya\OneDrive\Desktop\coding\def add(a,b)
    def __init__(length,width):
        ^
SyntaxError: invalid syntax
$ C:\Users\ramya\OneDrive\Desktop\coding> []
```

Corrected code:

```
class Rectangle:
    def __init__(self, length, width):
        self.length = length
        self.width = width
```

Justification:

@workspace /explain "self" is not defined, "self" is not defined

def add(a,b).py:1-5

Workspace

> Used 1 reference

✓ Crafting a clear explanation

You're missing the instance parameter; add `self` as the first parameter.

```
# ...existing code...
class Rectangle:
    def __init__(self, length, width):
        self.length = length
        self.width = width
# ...existing code...
```

Gotcha: `self` must be the first parameter of instance methods (conventionally named `self`).

GPT-5 mini • 1x

TASK-5:

```
def add(a,b) > ...
1   numbers = [1,2,3]
2   print(numbers[5])
```

PROBLEMS OUTPUT TERMINAL ... + v ... |

```
^C
PS C:\Users\ramya\OneDrive\Desktop\coding>&
  C:/Users/ramya/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/ramya/One
  Drive/Desktop/coding/def add(a,b).py"
Traceback (most recent call last):
  File "c:/Users/ramya/OneDrive/Desktop/cod
  ing/def add(a,b).py", line 2, in <module>
    print(numbers[5])
    ~~~~~^A
IndexError: list index out of range
PS C:\Users\ramya\OneDrive\Desktop\coding>[]
```

Corrected code:

```
def add(a,b) > ...
1     numbers = [1,2,3]
2     print(numbers[0] + numbers[1])
```

Justification: The crash happens since the index is outside the list range. Bounds checking or try/except prevents the error and ensures safe access.