

AI ASSISTED CODING

HTNO: 2403A51282

TASK1:

Welcome to Colab

le Edit View Insert Runtime Tools Help

inds + Code + Text Run all Copy to Drive

of contents

come to Colab!

Getting started

Data science

Machine learning

More resources

Featured examples

Section

- Learn how to [analyse images and detect items in your pictures](#) using Gemini (bonus, there's a [3D version](#) as well!).
- Unlock the power of the [Gemini thinking model](#), capable of solving complex tasks with its inner thoughts.

Explore complex use cases

- Use [Gemini grounding capabilities](#) to create a report on a company based on what the model can find on the Internet.
- Extract [invoices and form data from PDFs](#) in a structured way.
- Create [illustrations based on a whole book](#) using Gemini large context window and Imagen.

To learn more, take a look at the [Gemini cookbook](#) or visit the [Gemini API documentation](#).

```
def factr(n):  
    if n==0:  
        return 0  
    elif n==1:  
        return 1  
    else:  
        return n* factr(n-2)  
    print(factr("5"))
```

Colab now has AI features powered by [Gemini](#). The video below provides information on how to use these features, whether you're new to Python or a seasoned veteran.

Gemini X

the code i typed is not perfect please detect and fix the logical or syntax errors.

Working...

What can I help you build?

Gemini can make mistakes, so double-check it and use code with caution. [Learn more](#)

Use [Gemini grounding capabilities](#) to create a report on a company based on what the model can find on the Internet.

- Extract [invoices and form data from PDFs](#) in a structured way.
- Create [illustrations based on a whole book](#) using Gemini large context window and Imagen.

To learn more, take a look at the [Gemini cookbook](#) or visit the [Gemini API documentation](#).

```
def factr(n):  
    if n == 0:  
        return 1 # Factorial of 0 is 1  
    elif n == 1:  
        return 1 # Factorial of 1 is 1  
    else:  
        return n * factr(n - 1) # Correct recursive step  
  
    # Call the function with a valid integer input and print the result.  
    print(factr(5))
```

Colab now has AI features powered by [Gemini](#). The video below provides information on how to use these features, whether you're new to Python or a seasoned veteran.

Gemini

the log

TASK2:

```
#CODE2 - Fixed for consistent sorting  
def sort_list(data):  
    return sorted(data)  
items = ["3", "apple", "1", "banana", "2"] # Changed to all strings for consistent sorting  
print(sort_list(items))
```

```
['1', '2', '3', 'apple', 'banana']
```

Colab now has AI features powered by [Gemini](#). The video below provides information on how to use these features, whether you're new to Python or a seasoned veteran.

here CODE2 is not perfect detect the issue and fix the code for consistent sorting

You're right, the `sort_list` function will cause an error because the list `items` contains mixed data types (integers and strings). Python cannot directly compare integers and strings for sorting.

To fix this for consistent sorting, I'll change the list to contain only strings.

I've fixed the code in CODE2 by making all the elements in the list strings, and the code executed successfully.

TASK3:

```
# Example of improper file handling: opening a file but not closing it
try:
    file = open("my_file.txt", "w")
    file.write("This is some text.")
    # File is not explicitly closed here
except IOError as e:
    print(f"An error occurred: {e}")

# Best practice for file handling using a 'with' statement
try:
    with open("my_file.txt", "w") as file:
        file.write("This is some text using with.")
    # The 'with' statement automatically closes the file
except IOError as e:
    print(f"An error occurred: {e}")

# You can verify the file is closed by trying to access it
# print(file.closed) # This would cause an error because 'file' is not accessible
```

ib now has AI features powered by [Gemini](#). The video below provides information on

Gemini

K Generate Python snippet for file handling that opens a file but forgets to close it and improve it using the best practice using above image example

Screenshot 2025-09-19 094742.png

Okay, I can generate those Python snippets for file handling. The first will show a common mistake of not closing a file, and the second will demonstrate the recommended way to handle files using a `with` statement, which ensures the file is automatically closed.

I've generated and executed the code snippets demonstrating improper and proper file handling in Python. You can see the two code cells above.

Would you like to read the content from the file you just created, or perhaps try appending more text to it?

Task4:

```
# Example demonstrating ZeroDivisionError in a loop
for i in range(5):
    try:
        result = 10 / i
        print(f"Result of 10 / {i}: {result}")
    except ZeroDivisionError:
        print(f"Error: Cannot divide by {i}. Skipping this iteration.")

Error: Cannot divide by 0. Skipping this iteration.
Result of 10 / 1: 10.0
Result of 10 / 2: 5.0
Result of 10 / 3: 3.3333333333333335
Result of 10 / 4: 2.5
```

K in the above image a piece of code with a ZeroDivisionError inside a loop and dd error handling using try-except and continue execution safely

Screenshot 2025-09-19 095227.png

Okay, I can generate code snippets to demonstrate handling a ZeroDivisionError within a loop using a `try-except` block. The first example will show the error occurring, and the second will show how to handle it gracefully so the loop continues.

TASK5:

```
class BuggyStudent:
    def __init__(self, name, student_id, course_list=[]): # Incorrect parameter n
        self.studentName = name # Incorrect attribute reference
        self.id = student_id
        self.courses = course_list # Incorrect attribute name

    def add_course(self, course):
        self.course_list.append(course) # Incorrect attribute reference

    def get_summary(self):
        return f"Student: {self.studentName}, ID: {self.id}, Courses: {'', '.join(

# Example usage (will likely cause errors)
# s1 = BuggyStudent("Alice", 101, ["Math", "Science"])
# print(s1.get_summary())
```

K Include a buggy class definition with incorrect `__init__` parameters or attribute references. Ask AI to analyze and correct the constructor and attribute usage.

Okay, I can demonstrate a buggy class definition and then explain how to correct it. I'll modify cell `omXLA8060Uw` to introduce some common errors in the `__init__` method and attribute usage, and then I'll explain how to analyze and fix them.

I've introduced some bugs in the `BuggyStudent` class in cell `omXLA8060Uw`.

Here's an analysis of the errors in the code:

1 Incorrect parameter name in `__init__`: The