Untitled1

February 8, 2025

[3]:
```python
#unfixed  code
import numpy as np
import pandas as pd
import random

def generate_random_number(min_num, max_num):
    num = random.randint(min_num, max_num)
    print("Random number is: " + num)

def calc_average(num_list):
    total = sum(num_list)
    return total / lenght(num_list)

def check_prime(start, end):
    prime_list = []
    for i in range(start, end):
        if i % 2 == 0:
            prime_list.append(i)
    return prime_list

def load_data(filepath):
    data = pd.read_csv(filepath)
    return data

def main():
    num_list = [10, 20, 30, "forty", 50]
    print("The average is: ", calc_average(num_list))
    print("Prime numbers: ", check_prime(1, 10))

    file_path = "data.csv"
    data = load_data(file_path)
    print("Data loaded: ", data)

    random_num = generate_random_number(1, 100)
    print("Generated Random Number: ", random_num)

    try:
```

```
        print("Result of division: ", 10 / 0)
    except ZeroDivisionError:
        print("Can't divide by zero")

    numbers = [x for x in range(100) if x % 3 == 0 and x % 5 == 0]
    print("Numbers divisible by 3 and 5 are: ", numbers)

    undefined_function_call()

main()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[3], line 47
     43     print("Numbers divisible by 3 and 5 are: ", numbers)
     45     undefined_function_call()
---> 47 main()

Cell In[3], line 27, in main()
     25 def main():
     26     num_list = [10, 20, 30, "forty", 50]
---> 27     print("The average is: ", calc_average(num_list))
     28     print("Prime numbers: ", check_prime(1, 10))
     30     file_path = "data.csv"

Cell In[3], line 11, in calc_average(num_list)
     10 def calc_average(num_list):
---> 11     total = sum(num_list)
     12     return total / lenght(num_list)

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

[5]:
```
#chatgpt fixed code
import numpy as np
import pandas as pd
import random

# Function to generate a random number
def generate_random_number(min_num, max_num):
    num = random.randint(min_num, max_num)
    print("Random number is: " + str(num))   # Fixed concatenation error
    return num # Added return statement

# Function to calculate the average of a list of numbers
def calc_average(num_list):
    try:
```

```
        num_list = [x for x in num_list if isinstance(x, (int, float))]    # Filter out non-numeric values
        total = sum(num_list)
        return total / len(num_list)  # Fixed "lenght" typo
    except ZeroDivisionError:
        return "List is empty, can't compute average"

# Function to find prime numbers in a given range
def check_prime(start, end):
    prime_list = []
    for i in range(start, end):
        if i > 1:  # 1 is not prime
            for j in range(2, int(i ** 0.5) + 1):  # Check divisibility up to sqrt(i)
                if i % j == 0:
                    break
            else:
                prime_list.append(i)  # Fixed logic: Only add primes
    return prime_list

# Function to load CSV data
def load_data(filepath):
    try:
        data = pd.read_csv(filepath)
        return data
    except FileNotFoundError:
        print(f"Error: File '{filepath}' not found.")
        return None

# Main function
def main():
    num_list = [10, 20, 30, "forty", 50]
    print("The average is:", calc_average(num_list))  # Fixed function call

    print("Prime numbers:", check_prime(1, 10))   # Fixed logic in prime checking

    file_path = "data.csv"
    data = load_data(file_path)
    if data is not None:
        print("Data loaded:\n", data)

    random_num = generate_random_number(1, 100)
    print("Generated Random Number:", random_num)# Now prints correctly

    try:
        print("Result of division:", 10 / 0)  # Will trigger ZeroDivisionError
    except ZeroDivisionError:
```

```
        print("Can't divide by zero.")

    # Generate numbers divisible by 3 and 5 up to 100
    numbers = [x for x in range(100) if x % 3 == 0 and x % 5 == 0]
    print("Numbers divisible by 3 and 5:", numbers)

    # Removed undefined_function_call() since it's not defined

# Run the main function
if __name__ == "__main__":
    main()
```

```
The average is: 27.5
Prime numbers: [2, 3, 5, 7]
Error: File 'data.csv' not found.
Random number is: 30
Generated Random Number: 30
Can't divide by zero.
Numbers divisible by 3 and 5: [0, 15, 30, 45, 60, 75, 90]
```

[7]:
```
#perplexity ai fixed code
import numpy as np
import pandas as pd
import random

def generate_random_number(min_num, max_num):
    num = random.randint(min_num, max_num)
    print("Random number is: " + str(num)) #Corrected: Convert num to string

def calc_average(num_list):
    total = sum(num_list)
    return total / len(num_list) #Corrected: Use len() instead of lenght()

def check_prime(start, end):
    prime_list = []
    for i in range(start, end + 1): # Corrected: Iterate through the correct
 range
        if i > 1: # Corrected: Check only for numbers greater than 1
            is_prime = True
            for j in range(2, int(i**0.5) + 1):
                if (i % j) == 0:
                    is_prime = False
                    break
            if is_prime:
                prime_list.append(i)
    return prime_list
```

```
def load_data(filepath):
    try:
        data = pd.read_csv(filepath)
        return data
    except FileNotFoundError:
        print(f"Error: File not found at {filepath}")
        return None # Handle the case where the file is not found

def main():
    num_list = [10, 20, 30, 40, 50] #Corrected: "forty" to 40
    print("The average is: ", calc_average(num_list))
    print("Prime numbers: ", check_prime(1, 10))

    file_path = "data.csv"
    data = load_data(file_path)
    if data is not None:
        print("Data loaded successfully.") # Improved: Confirmation message
        print(data)  # Print the data

    generate_random_number(1, 100) # Corrected: remove the assignment
    #print("Generated Random Number: ", random_num) #Removed: random_num not defined

    try:
        print("Result of division: ", 10 / 0)
    except ZeroDivisionError as e:
        print(f"Can't divide by zero: {e}") #Improved: include the error message

    numbers = [x for x in range(100) if x % 3 == 0 and x % 5 == 0]
    print("Numbers divisible by 3 and 5 are: ", numbers)

    #undefined_function_call() #Corrected: remove the call, because it's undefined

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

```
The average is:  30.0
Prime numbers: [2, 3, 5, 7] Error:
File not found at data.csv
Random number is: 14
Can't divide by zero: division by zero
Numbers divisible by 3 and 5 are:  [0, 15, 30, 45, 60, 75, 90]
```

[ ]:

Chat gpt ai:

1)Improve `generate_random_number()`

Issue: The function prints but does not return anything useful.
Improvement: Allow flexibility by returning the number instead of printing inside the function.

```
python
CopyEdit
def generate_random_number(min_num, max_num):
    return random.randint(min_num, max_num)
```

2) Improve `calc_average()`

Issues:

- It filters non-numeric values but doesn't alert the user.
- Returns `"List is empty, can't compute average"` instead of `None` or `float('nan')`, which can be more useful for handling.

Improvement:

```
python
CopyEdit
def calc_average(num_list):
    num_list = [x for x in num_list if isinstance(x, (int, float))]

    if not num_list:
        print("Warning: No valid numbers to calculate average.")
        return None  # Return None instead of a string

    return sum(num_list) / len(num_list)
```

3)Optimize `check_prime()`

Issue: Checking divisibility for all numbers up to `sqrt(n)` for **every number** is inefficient.
Improvement: Use a **list comprehension** for performance and readability.

```
python
CopyEdit
def check_prime(start, end):
    return [n for n in range(max(2, start), end) if all(n % d != 0 for d in
range(2, int(n **
```

4)Improve `load_data()`

Issue: Doesn't handle other **errors like empty files** or **corrupt CSVs**.
Improvement:

```
python
CopyEdit
def load_data(filepath):
    try:
        data = pd.read_csv(filepath)
        if data.empty:
            print(f"Warning: '{filepath}' is empty.")
            return None
        return data
    except FileNotFoundError:
        print(f"Error: File '{filepath}' not found.")
    except pd.errors.EmptyDataError:
        print(f"Error: '{filepath}' contains no data.")
    except pd.errors.ParserError:
        print(f"Error: Could not parse '{filepath}'. Check file format.")
    return None
```

Perplexity ai:

Improvements and Suggestions

- **`if __name__ == "__main__":`**: I wrapped the `main()` call in this standard Python construct. This ensures that `main()` is only called when the script is executed directly, not when it's imported as a module into another script.
- **Error Handling**: Added `try...except` block in `load_data` to handle the `FileNotFoundError` gracefully. This prevents the program from crashing if the CSV file doesn't exist.
- **Function Comments/Docstrings**: Consider adding docstrings to each function to explain its purpose, arguments, and return value. This makes the code more readable and maintainable.
- **More Robust Prime Number Check**: The prime number check could be further optimized, but the current version is a significant improvement in terms of correctness and efficiency.
- **Data Validation**: In the `load_data` function, consider adding validation steps to ensure that the loaded data has the expected format and content. This can prevent errors later in the program.
- **Clearer Output**: Improved some of the print statements to provide more informative output to the user.
- **Return values**: Consider returning a value from `generate_random_number` instead of just printing it, for greater flexibility.