Use two AI tools to find and fix problems in a broken Python code

Broken 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()
```

ChatGPT 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))  # Convert number to string before concatenation
    return num  # Return the generated number


def calc_average(num_list):
    try:
        num_list = [x for x in num_list if isinstance(x, (int, float))]  # Remove non-numeric values
        total = sum(num_list)
        return total / len(num_list)  # Fixed typo: changed "lenght" to "len"
    except ZeroDivisionError:
        return "Cannot calculate average of empty list."


def check_prime(start, end):
    prime_list = []
    for num in range(start, end + 1):  # Include 'end' in the range
        if num > 1:
            for i in range(2, int(num ** 0.5) + 1):  # Check divisibility up to sqrt(num)
                if num % i == 0:
                    break
            else:
                prime_list.append(num)  # Fixed logic: now correctly finds prime
numbers
    return prime_list

def load_data(filepath):
    try:
        data = pd.read_csv(filepath)
        return data
    except FileNotFoundError:
        return "Error: File not found."
    except pd.errors.EmptyDataError:
        return "Error: File is empty."
    except pd.errors.ParserError:
        return "Error: File could not be parsed."

def main():
    num_list = [10, 20, 30, "forty", 50]
    print("The average is: ", calc_average(num_list))  # Now handles non-
numeric values correctly

    print("Prime numbers: ", check_prime(1, 10))  # Now correctly finds
prime numbers

    file_path = "data.csv"
    data = load_data(file_path)
    print("Data loaded: ", data)  # Handles file errors correctly
    random_num = generate_random_number(1, 100)
    print("Generated Random Number: ", random_num)  # Now prints the returned value

    try:
        print("Result of division: ", 10 / 0)
    except ZeroDivisionError:
        print("Can't divide by zero")  # Correctly handles division 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()  # Removed undefined function call to prevent runtime error

main()
```

Fixes Applied

1. **Fixed print concatenation issue**:
    1. `"Random number is: " + num` → `"Random number is: " + str(num)`
2. **Fixed function return issue**:
    1. `generate_random_number()` now returns the number instead of just printing it.
3. **Fixed typo in `calc_average()`**:
    1. `"lenght(num_list)"` → `"len(num_list)"`
4. **Handled non-numeric values in `calc_average()`**:

    a. Now filters out non-numeric values before calculating the average.
5. **Fixed prime number logic in `check_prime()`**:
    a. The function now correctly checks for prime numbers.
6. **Handled exceptions in `load_data()`**:
    a. Added error handling for file-related issues like `FileNotFoundError`.
7. **Fixed function call issue in `main()`**:
    a. `generate_random_number()` was not returning a value, now it does.
8. **Removed undefined function call**:
    a. `undefined_function_call()` removed to prevent runtime errors.

Microsoft copilot 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:", num)
```

```
def calc_average(num_list):
    total = sum(num for num in num_list if isinstance(num, (int, float)))
    return total / len(num_list) if len(num_list) > 0 else 0

def check_prime(start, end):
    prime_list = []
    for i in range(start, end + 1):
        if i > 1:
            for j in range(2, i):
                if (i % j) == 0:
                    break
            else:
                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, 40, 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:\n", 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)
    try:
        undefined_function_call()
    except NameError:
        print("Function not defined")
main()
```

Here are some of the fixes:

1. **Type Check for List Items**: I added a check to ensure only numeric values are summed in `calc_average`.
2. **Prime Number Calculation**: I corrected the prime-checking logic in `check_prime`.
3. **String to Number**: Replaced `"forty"` with `40` in `num_list` to avoid type error.
4. **String Concatenation**: Fixed the `print` statement in `generate_random_number` to properly concatenate the string with the number.

5. **Handling Undefined Function**: Added a `try-except` block to handle the call to `undefined_function_call`.