



Python Syntax Checker Tool

Content

- Problem Statement
- Description & Functional Components
- Why this ?
- Applying the try() method
- Using the compile() function
- Applying the except() method

Problem Statement

No. 1

To create a Python utility that reads the contents of .py file and checks for syntax errors using the built-in compile() function.

Input:

A python file (.py file)

Description

- Accept the file path of a Python script from the user
- Read the entire file content
- Use the `compile()` function to validate the syntax
- If a syntax error is detected, display the error message with the line number
- If no errors are found, display a success message indicating the file is valid

Functional Components

- File input through user prompt
- Read and store script content using file handling
- Validate syntax using `compile()`
- Catch syntax errors using `try-except`
- Display result to the user with appropriate message

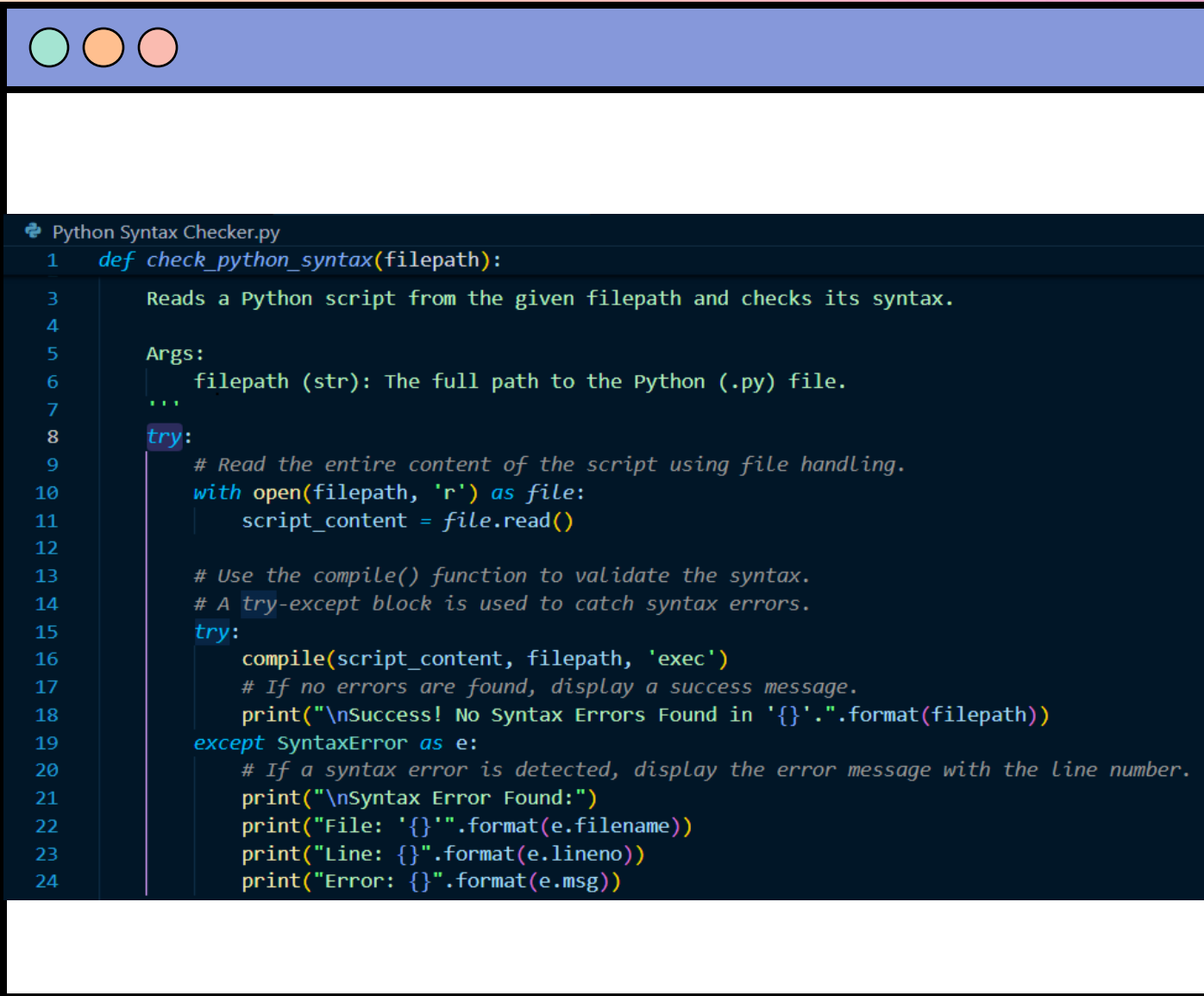


I chose this problem to address the following question asked :

" How can we develop a Python tool that reads another Python script and identifies any syntax errors without running the program ? "

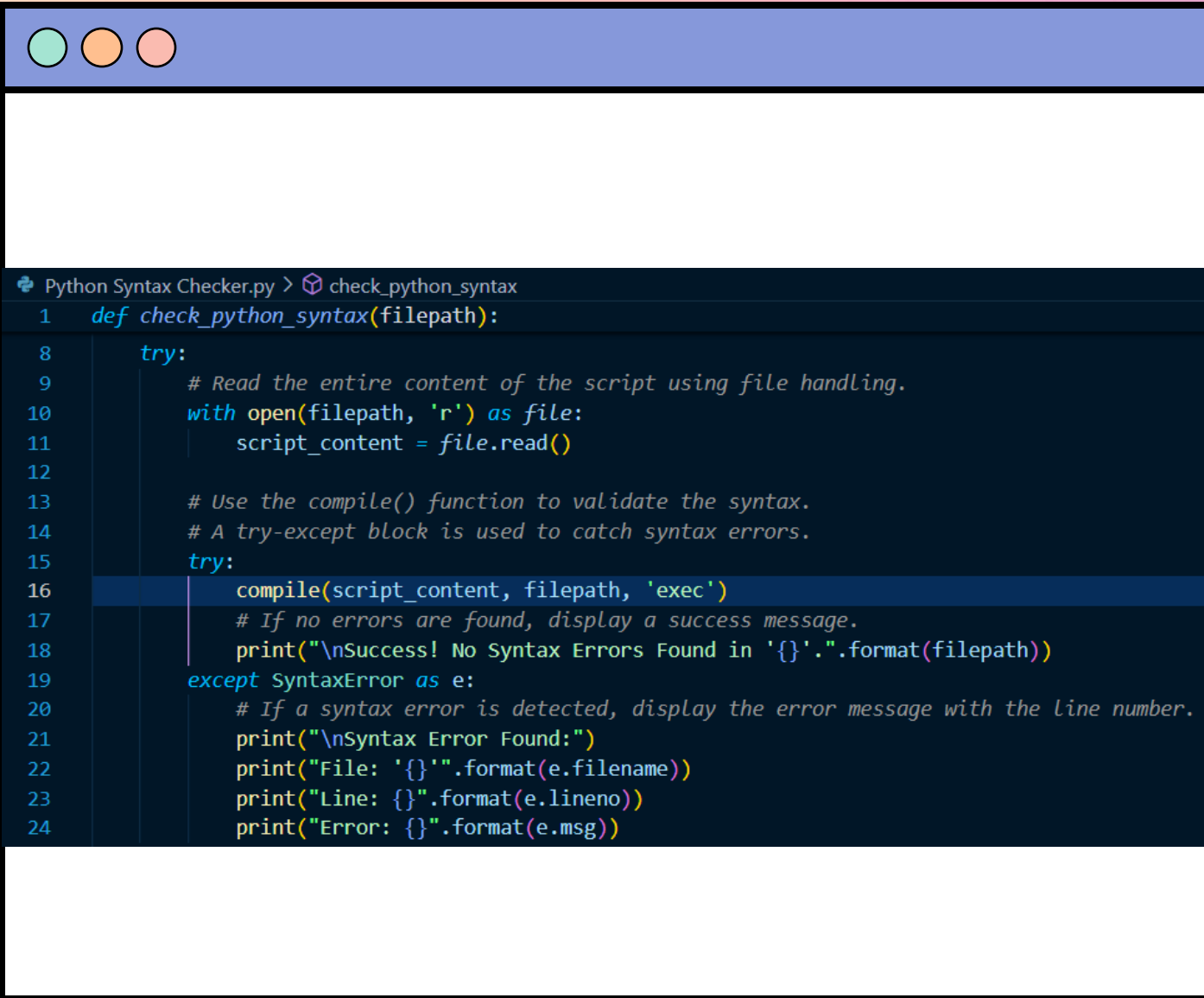
I relate this problem to myself and feels like a tool to pre-identify errors before executing the script will help a lot of beginners.

Why this ?



```
Python Syntax Checker.py
1 def check_python_syntax(filepath):
2
3     Reads a Python script from the given filepath and checks its syntax.
4
5     Args:
6         filepath (str): The full path to the Python (.py) file.
7     ...
8     try:
9         # Read the entire content of the script using file handling.
10        with open(filepath, 'r') as file:
11            script_content = file.read()
12
13        # Use the compile() function to validate the syntax.
14        # A try-except block is used to catch syntax errors.
15        try:
16            compile(script_content, filepath, 'exec')
17            # If no errors are found, display a success message.
18            print("\nSuccess! No Syntax Errors Found in '{}'.format(filepath))
19        except SyntaxError as e:
20            # If a syntax error is detected, display the error message with the line number.
21            print("\nSyntax Error Found:")
22            print("File: {}".format(e.filename))
23            print("Line: {}".format(e.lineno))
24            print("Error: {}".format(e.msg))
```

Applying the *try()* method



```
Python Syntax Checker.py > check_python_syntax
1 def check_python_syntax(filepath):
8     try:
9         # Read the entire content of the script using file handling.
10        with open(filepath, 'r') as file:
11            script_content = file.read()
12
13        # Use the compile() function to validate the syntax.
14        # A try-except block is used to catch syntax errors.
15        try:
16            compile(script_content, filepath, 'exec')
17            # If no errors are found, display a success message.
18            print("\nSuccess! No Syntax Errors Found in '{}'.format(filepath))
19        except SyntaxError as e:
20            # If a syntax error is detected, display the error message with the line number.
21            print("\nSyntax Error Found:")
22            print("File: {}".format(e.filename))
23            print("Line: {}".format(e.lineno))
24            print("Error: {}".format(e.msg))
```

Using the *compile()* function

```
12
13     # Use the compile() function to validate the syntax.
14     # A try-except block is used to catch syntax errors.
15     try:
16         compile(script_content, filepath, 'exec')
17         # If no errors are found, display a success message.
18         print("\nSuccess! No Syntax Errors Found in '{}'.format(filepath))
19     except SyntaxError as e:
20         # If a syntax error is detected, display the error message with the line number.
21         print("\nSyntax Error Found:")
22         print("File: {}".format(e.filename))
23         print("Line: {}".format(e.lineno))
24         print("Error: {}".format(e.msg))
25
26     except FileNotFoundError:
27         print("\nError: The file '{}' was not found.".format(filepath))
28     except Exception as e:
29         print("\nAn unexpected error occurred: {}".format(e))
```

Applying the *except()* method

For Error Handling

Thank you

From : Nisarg Kumar Gharde

