Instructions

Write a program to:

- Check if a given number is prime or not.
- Find the factorial of a given number.
- Create a simple graphical interface where the user can enter the number in a textbox.
- Ensure to include comments in your code to explain how it works.
- Explore ways to optimize your code for faster execution time.

Running the File

1. Open `Outputs` folder
2. Run `PrimeFact.exe`

Documentation

The implementation uses Python to get an input and (1) check if it's a prime number and (2) find its factorial result.

Functions

Two primary functions are made, which work for both checking a number's primality and its factorial result.

`primalCheck(n)`

This simply checks if the number `n` is prime or not. It is a prime number if there are no factors except one or itself. We do this by getting the modulo of the square root of `n` by the iterator `i`. If the result is 0 (meaning it does not have a remainder), it is not a prime number. This is much more efficient in finding a prime number instead of iterating from 2 to `n`.

Take note that the values 0 and 1 aren't considered prime (nor is 1 considered composite), hence returning a `False`.

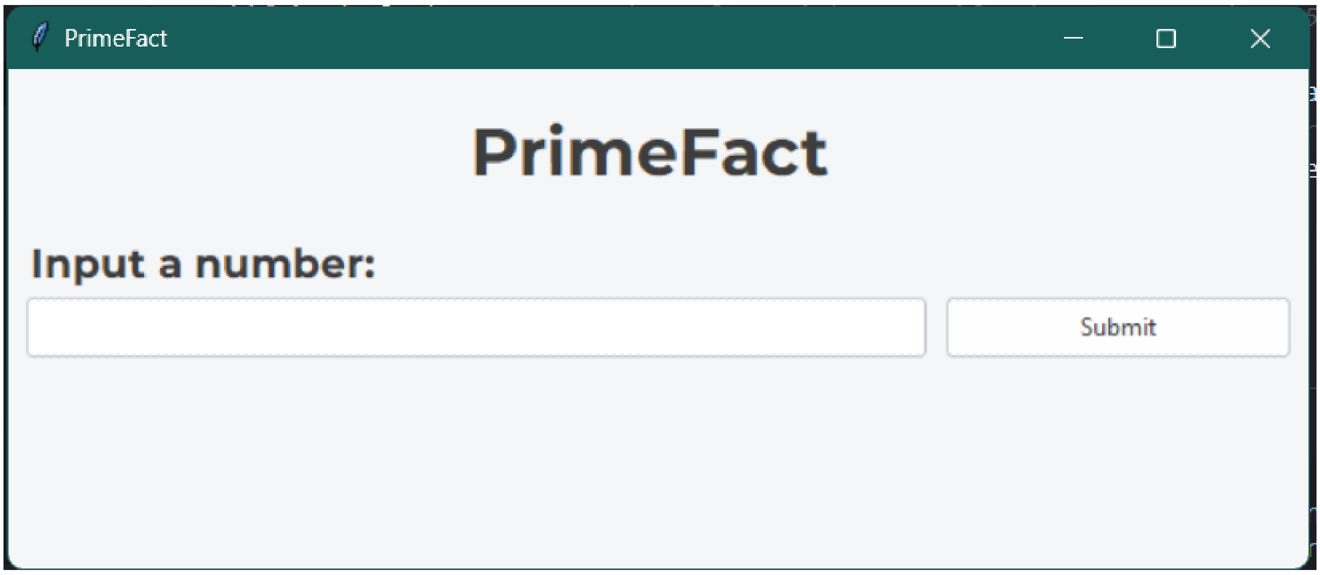
```python
from math import sqrt # Module needed for sqrt() function

# Function to check if number is prime
def primalCheck(n):
    if n < 2:
        return False # 0 and 1 are not prime numbers, return false

    for i in range(2, int(sqrt(n)) + 1): # Divide from 2 to sqrt(n)+1 of a given number
        if n % i == 0:
            return False # If number has no remainder, return false
    return True # Else return true
```

fact(n)

Gets the factorial of the number `n`. For readability, and to prevent large numbers from filling up the screen, the results are formatted using commas (for less than 15 digits) or scientific notation (for larger values). The `Decimal` function is there to allow for larger values (up to 4300 digits), which will be explained later.

```
# Function to get factorial
from decimal import Decimal # Decimal needed for larger numbers

def fact(n):
    result = 1 # Start result with 1 (0! = 1)
    for i in range(1,n+1):
        result *= i # Iterate by multiplying iterator from the result
    if len(str(result)) < 15: # Display comma-separated results for less than 15 digits
        return f"{result:,}"
    else: return f"{Decimal(result):.10e}" # Otherwise, display scientific notation of result
(for larger numbers)
```

Graphical User Interface (GUI)



GUI

`Tkinter` is primarily used for creating a GUI. It's a simple implementation containing labels, and input, and a button for events.

```
...
# GUI that will be displayed in the exeutable

window = ThemedTk(theme="yaru") # Display a window
window.title("PrimeFact") # Title of program

window.geometry("650x250") # Fixed sized of 650x250 pixels

# Ensured window is not resizable
window.minsize(650,250)
window.maxsize(650,250)
```

```
# Changed BGM color
window.config(background="#F5F6F7")

# Header of program
greeting = ttk.Label(window, text="PrimeFact", font=('Montserrat', '25', 'bold'))
greeting.grid(row=0,columnspan=2,padx=10,pady=15)

# Label to input number
inputLabel = ttk.Label(window, text="Input a number:", font=('Montserrat', '15','bold'))
inputLabel.grid(row=1,column=0,sticky="W",padx=10,pady=3)

# Textbox for input
entry = ttk.Entry(window,width=40,font=('Montserrat', '12'))
entry.grid(row=2,column=0,sticky="W",padx=8)

# Submit button
submit = ttk.Button(window, text="Submit", width=25, command=handle_click)
submit.grid(row=2,column=1)

# Label that appears when input is invalid.
warningLabel = ttk.Label(window, text="",font=('Montserrat', '10','bold'), foreground='red')
warningLabel.grid(row=3,column=0,sticky="W",padx=10,pady=3)

# Displays primaility check of input
primeLabel = ttk.Label(window, text="",font=('Montserrat', '12', 'bold'))
primeLabel.grid(row=4,column=0,columnspan=2,sticky="W",padx=10,pady=2)

# Displays factorial of input
factLabel = ttk.Label(window, text="", font=('Montserrat', '12', 'bold'))
factLabel.grid(row=5,column=0,columnspan=2,sticky="W",padx=10,pady=2)

# Main event GUI that will be executed
window.mainloop()
```

The main part is displaying the results of our functions in the program, which is primarily handled here:

```
# Displays in tkinter Label whether a number is prime or not
def isPrime(new_val):
    if primalCheck(new_val)==True:
        primeLabel["text"] = str(f"{new_val:,}") + " is a prime number."
    else: primeLabel["text"] = str(f"{new_val:,}") + " is not a prime number."

# Displays the factorial of a value
def factorial(new_val):
    factLabel["text"] = str(new_val)+ "! is " + str(fact(new_val)) + "."

# Function to handle labels given input
def handle_click():
    value = entry.get() # Gets number value from textbox input

# Regular expression to determine if input is a valid number
# e.g. 1,234,567 is considered valid, but 1,23,456 is not a valid number
    if re.search(r"^\d{1,3}(?:,\d{3})*$", value):
        value = value.replace(',','')          # Removes all the commas in the value so it can
be used for prime and factorial
```

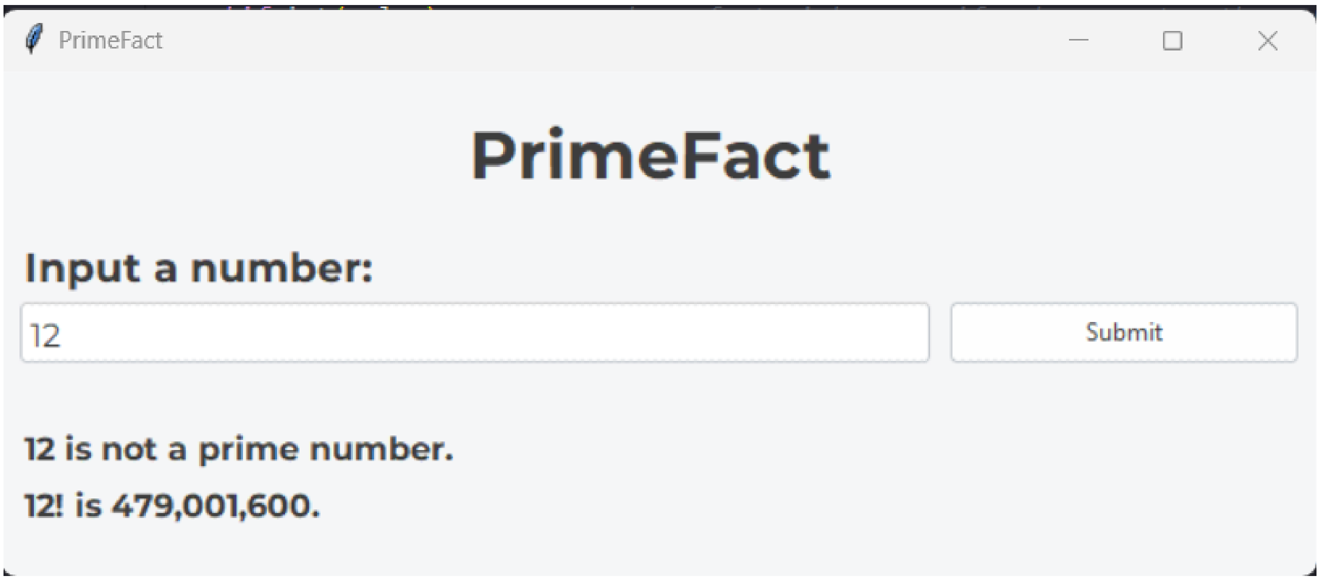
```
# Invalid input warning occurs when:
    # Invalid characters on input (i.e)
    # Value is less than zero (i.e. negative numbers)
    # Input starts with zero followed by a nonzero number (i.e. phone numbers)
    if value.isdigit() == False or int(value) < 0 or value[0]=='0':
        primeLabel["text"] = "" # Resets prime label
        factLabel["text"] = "" # Resets factorial label
        warningLabel["text"] = "Invalid input!" # Display error

# Factorial or prime error: required due to computational limitations
    elif int(value) > 1558: # Throw factorial error if value greater than 1558
        primeLabel["text"] = ""
        factLabel["text"] = "Factorial error: Please input a number less than or equal to 1558."

        if int(value) > 67280421310721: # Throw prime error if value greater than 67280421310721
            primeLabel["text"] = "Prime error: Highest possible value reached."
        else: isPrime(int(value))
        warningLabel["text"] = "" # Resets warning label

# Proceed with function as usual if no errors
    else:
        warningLabel["text"] = ""
        isPrime(int(value))
        factorial(int(value))
```

The `handle_click()` function is integrated in the submit button of the program. When a user inputs something, it checks if (1) it is valid or not and (2) it is a value larger than what the program can handle. The specific values for factorial and prime errors are there for a reason, explained further in Error Handling

Error Handling

During initial testing, I realized how it could be heavy for the program to compute when values are increased. To prevent this, user inputs are limited by 1558 (factorial) and 67280421310721 (prime). There's also specific errors for special characters, and exceptions for valid comma-separated numbers.

Valid Input



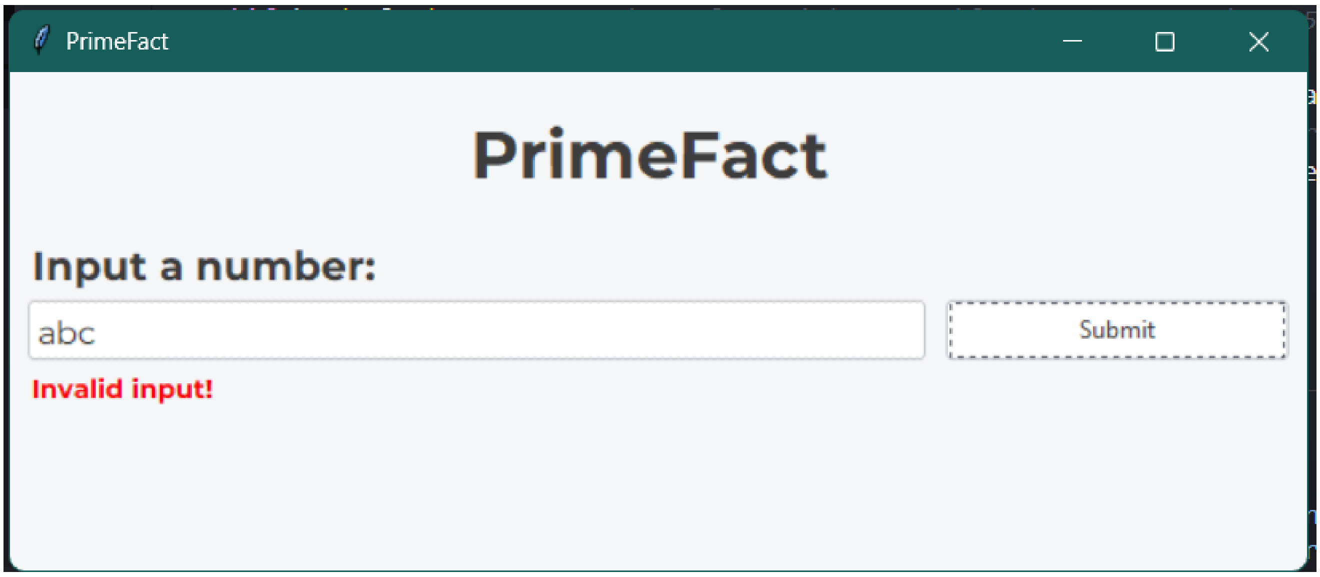
12 is not a prime number.

12! is 479,001,600.

Valid

Any number (within the prescribed limit) is valid, and will display the correct result.

Invalid Input: Incorrect Characters


PrimeFact

PrimeFact

Input a number:

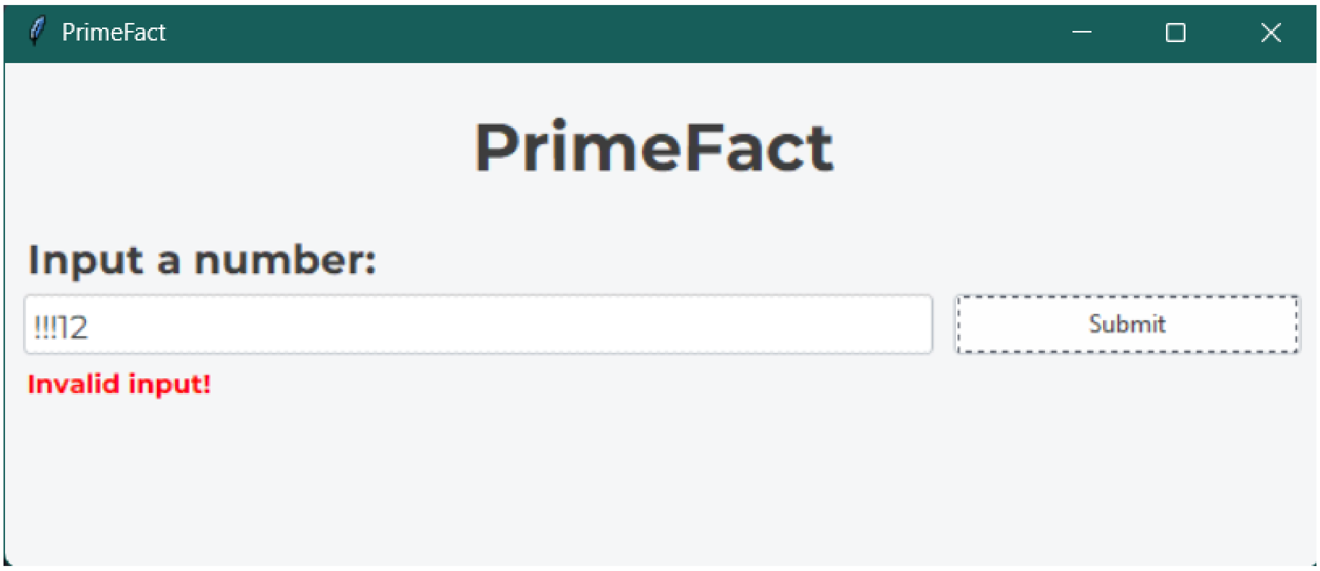
abc

Submit

Invalid input!

Invalid

User that enters alphabetical characters (with the exception of commas) will raise an error.


PrimeFact

PrimeFact

Input a number:

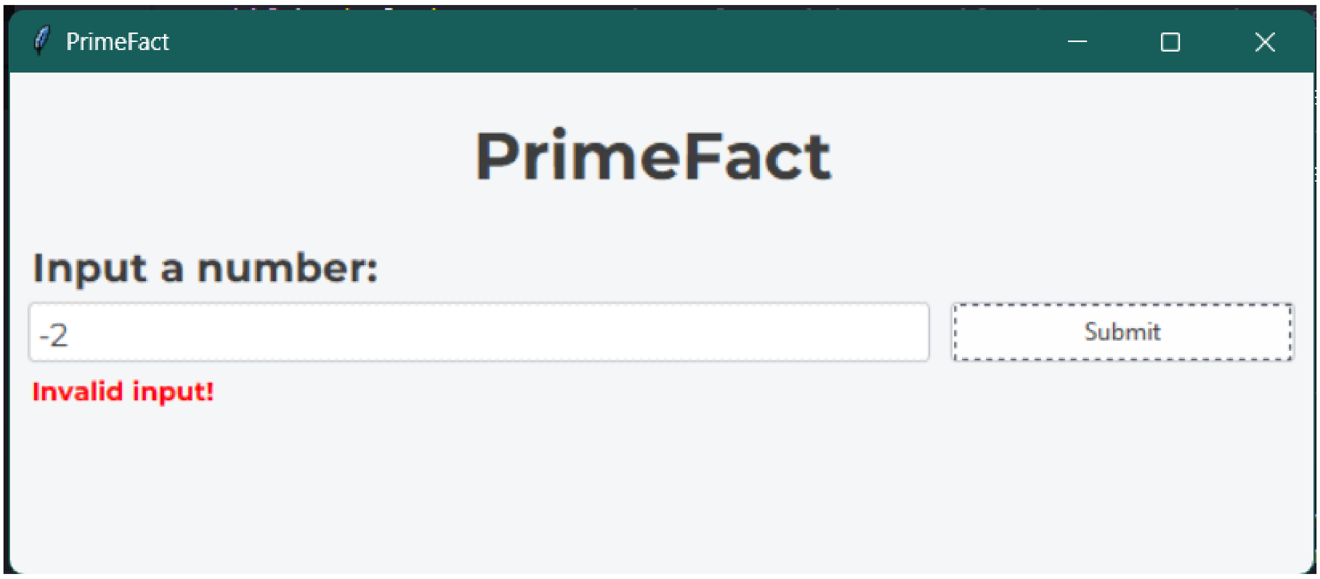
!!!12

Submit

Invalid input!

Invalid

Special characters are invalid, with the exception of commas (which also has some exceptions).


PrimeFact

PrimeFact

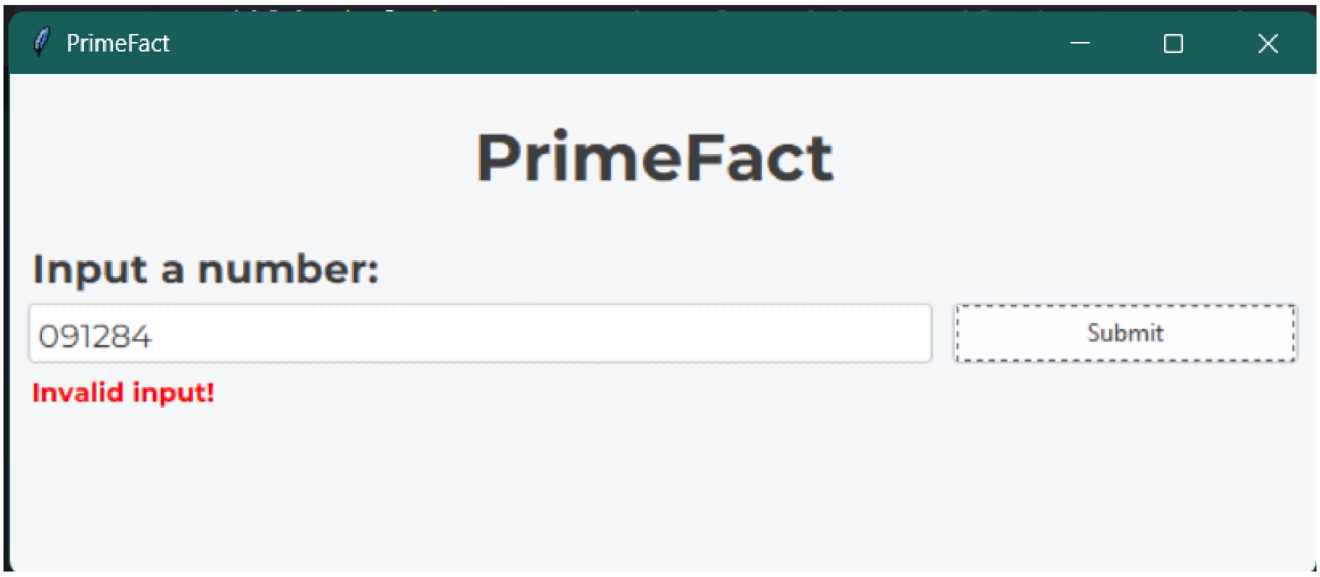
Input a number:

-2

Invalid input!

Invalid

Negative numbers are invalid, since they we cannot determine its primality and/or its factorial equivalent.


PrimeFact

PrimeFact

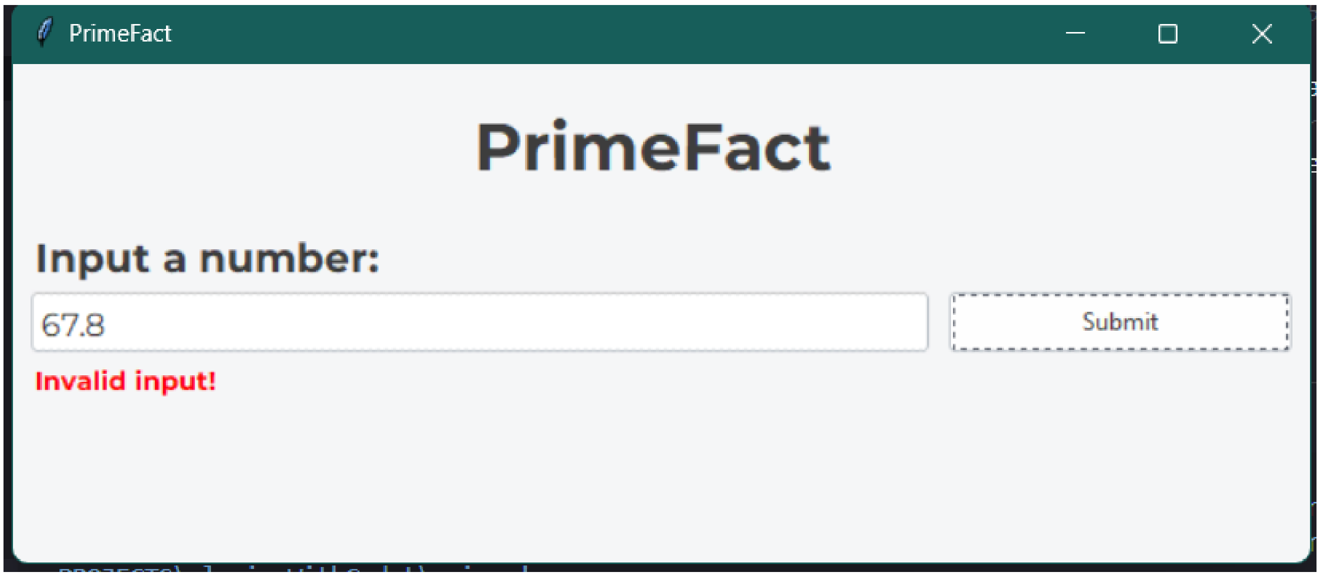
Input a number:

091284

Invalid input!

Invalid

Values that start with 0 are invalid. Most programs (including Python) would forego the zero and consider it valid (as it would be read as 91,284). I decided to consider it as an invalid input.



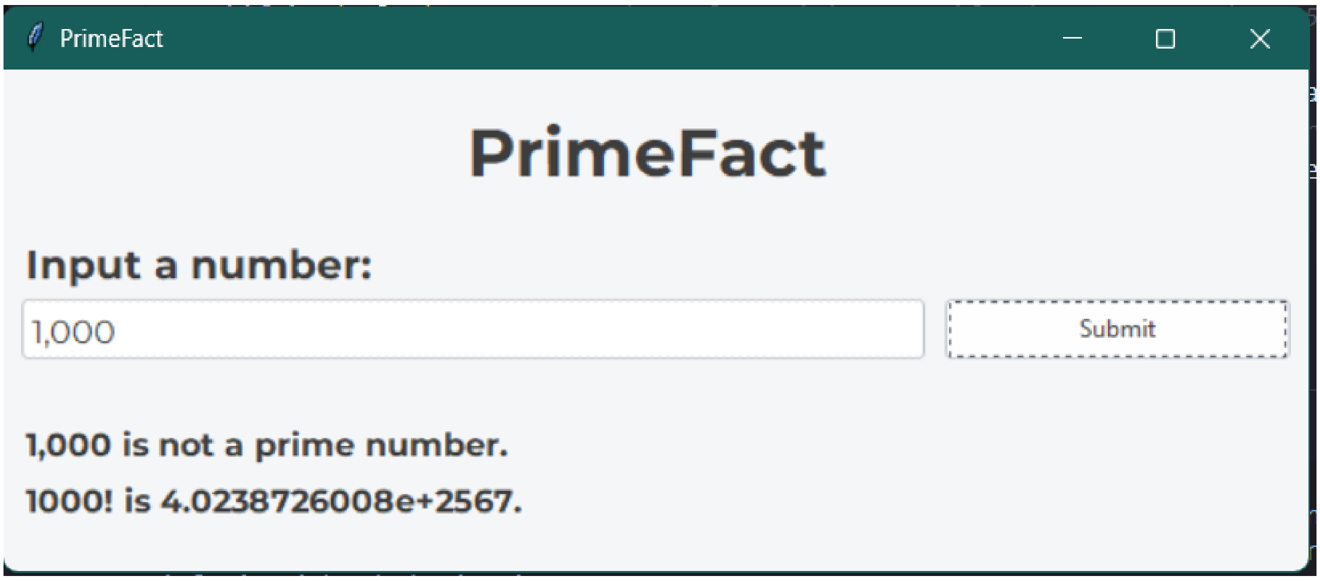
```
Invalid input!
```

Invalid

Decimal values are also invalid.

Exceptions: Commas

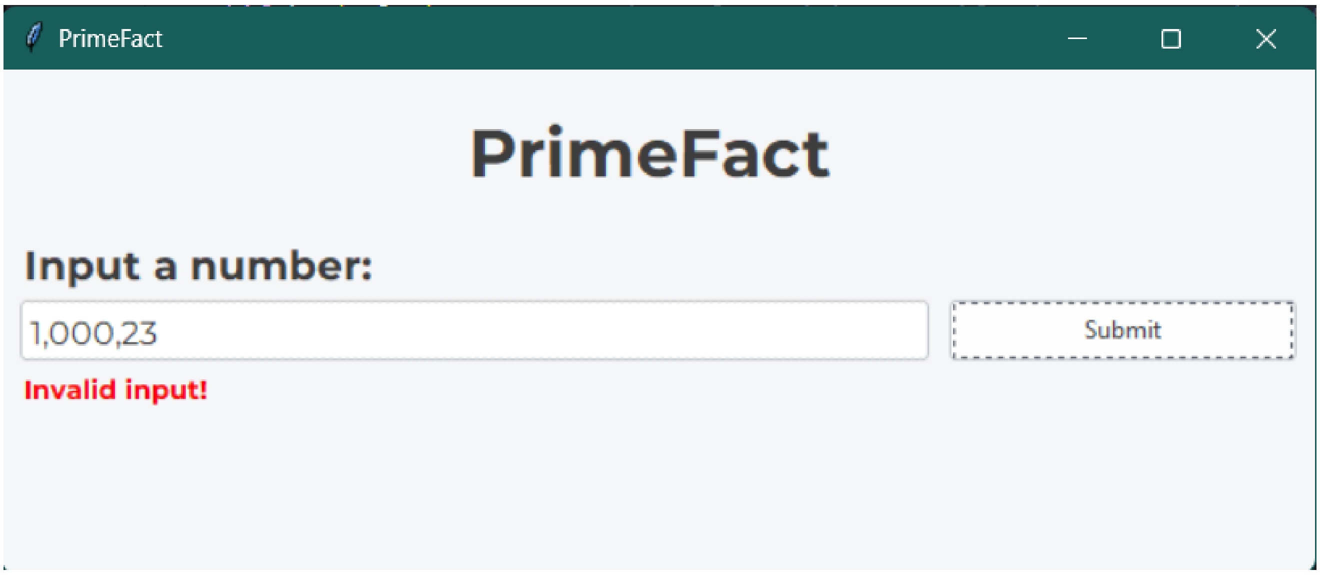
There are regular expressions included in the source code, because I want to consider comma-separated numbers as valid. Granted, they must be formatted correctly or it will be considered invalid.



```
1,000 is not a prime number.
1000! is 4.0238726008e+2567.
```

Valid

This input lacks one more digit at the rightmost part of the value, so this is considered invalid.



1,000,23

Invalid input!

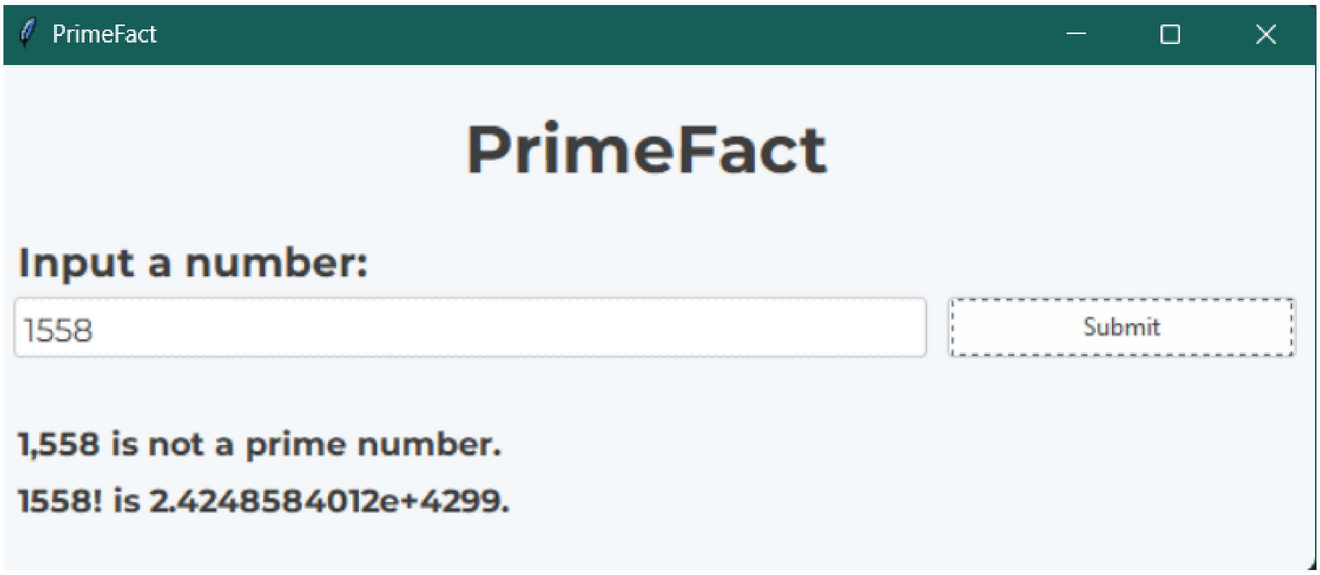
Invalid

Factorial Limitations

The reason why 1558 was the limit for getting the factorial is because factorials grow exponentially fast, and it will take time for regular computers to display the result. Without the `Decimal` function earlier, the limit would be 170. Any more and it will throw an error when I try to convert the value to scientific notation:

```
OverflowError: int too large to convert to float
```

Using the `Decimal` function, it manages to increase the limits for our factorial function. It can convert more than 4300 digits into scientific notation.

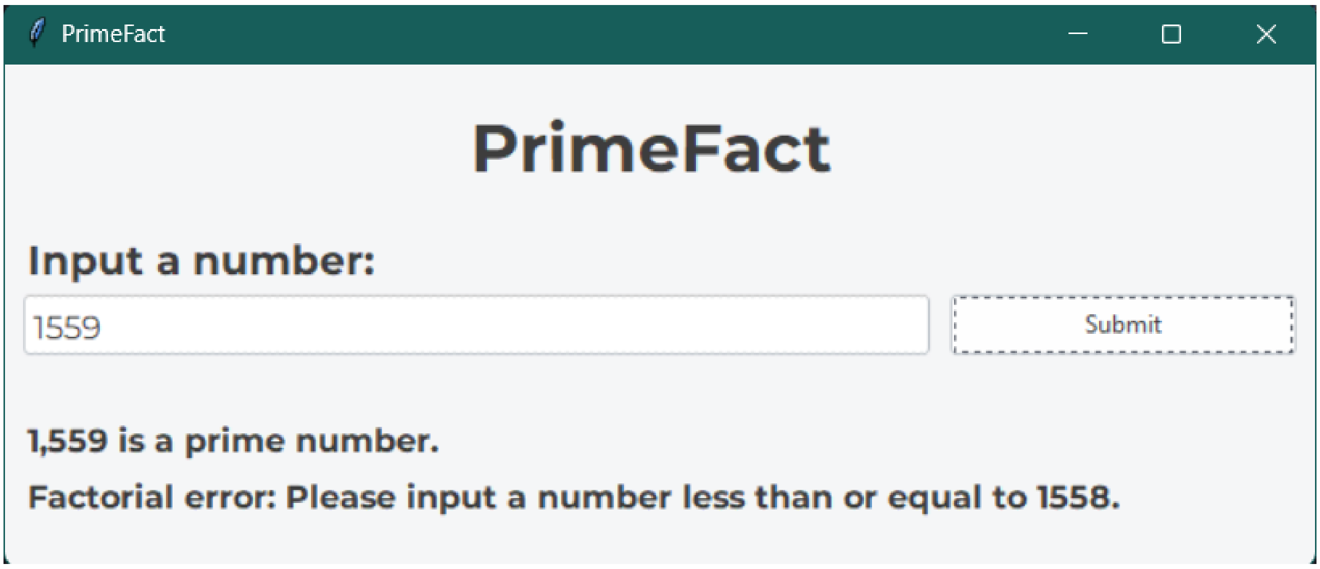


1558

1,558 is not a prime number.
1558! is 2.4248584012e+4299.

Limits for factorial

If user inputs 1559, it will display this error.



1559

1,559 is a prime number.

Factorial error: Please input a number less than or equal to 1558.

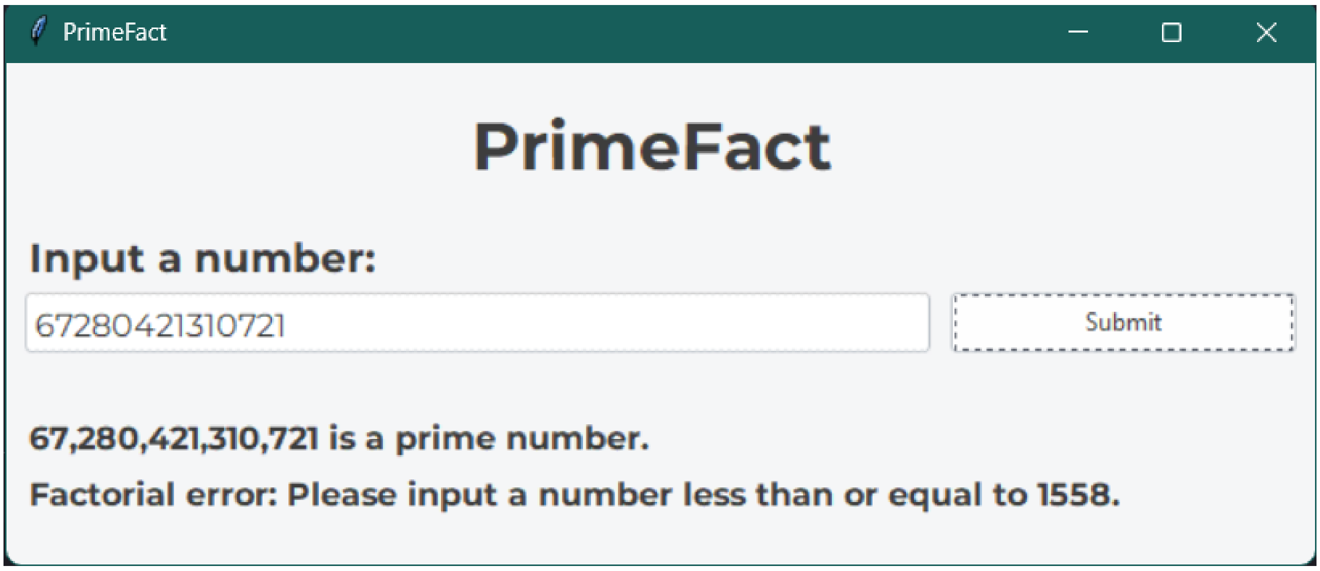
Invalid

Without this error and it passes through the `fact()` function, the Python interpreter wouldn't be able to convert the value to scientific notation either, releasing this exception

```
ValueError: Exceeds the limit (4300 digits) for integer string conversion;
use sys.set_int_max_str_digits() to increase the limit
```

Prime Number Limitations

Likewise, the biggest possible prime number that the program can still check is 67,280,421,310,721 (14 digits).



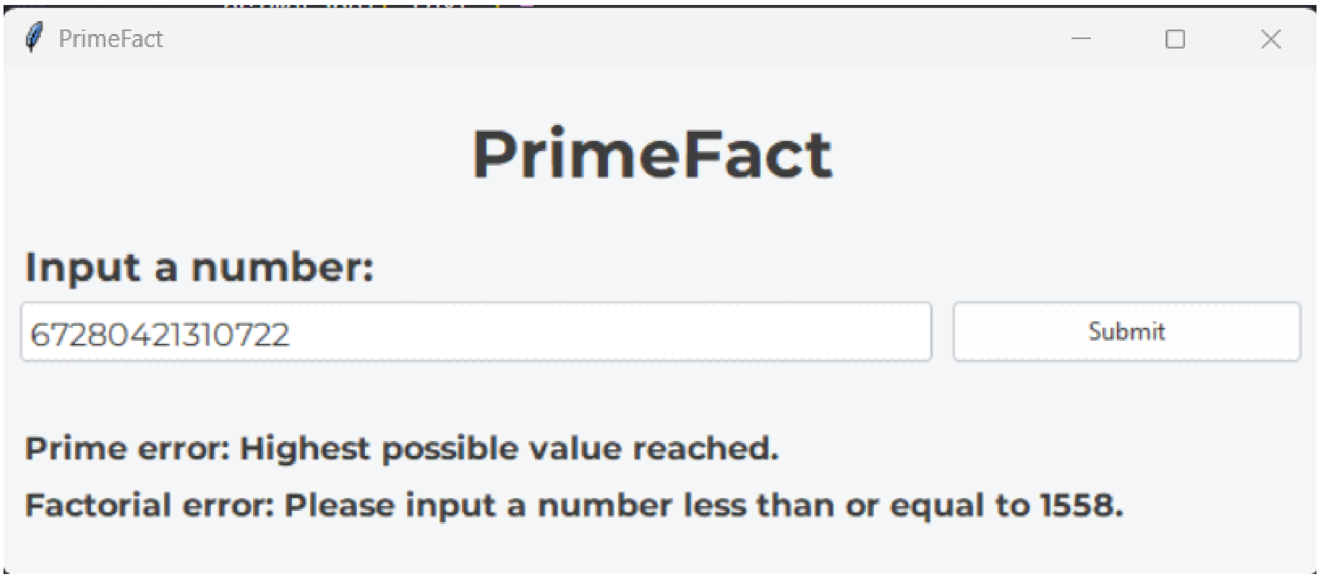
67280421310721

67,280,421,310,721 is a prime number.

Factorial error: Please input a number less than or equal to 1558.

Valid

The next biggest prime number is 170,141,183,460,469,231,731,687,303,715,884,105,727 (39 digits). When tested, it crashes the program altogether. I have no assurance as to what the next possible number this program can run in, but for assurance, I decided that this is the limit.



67280421310722

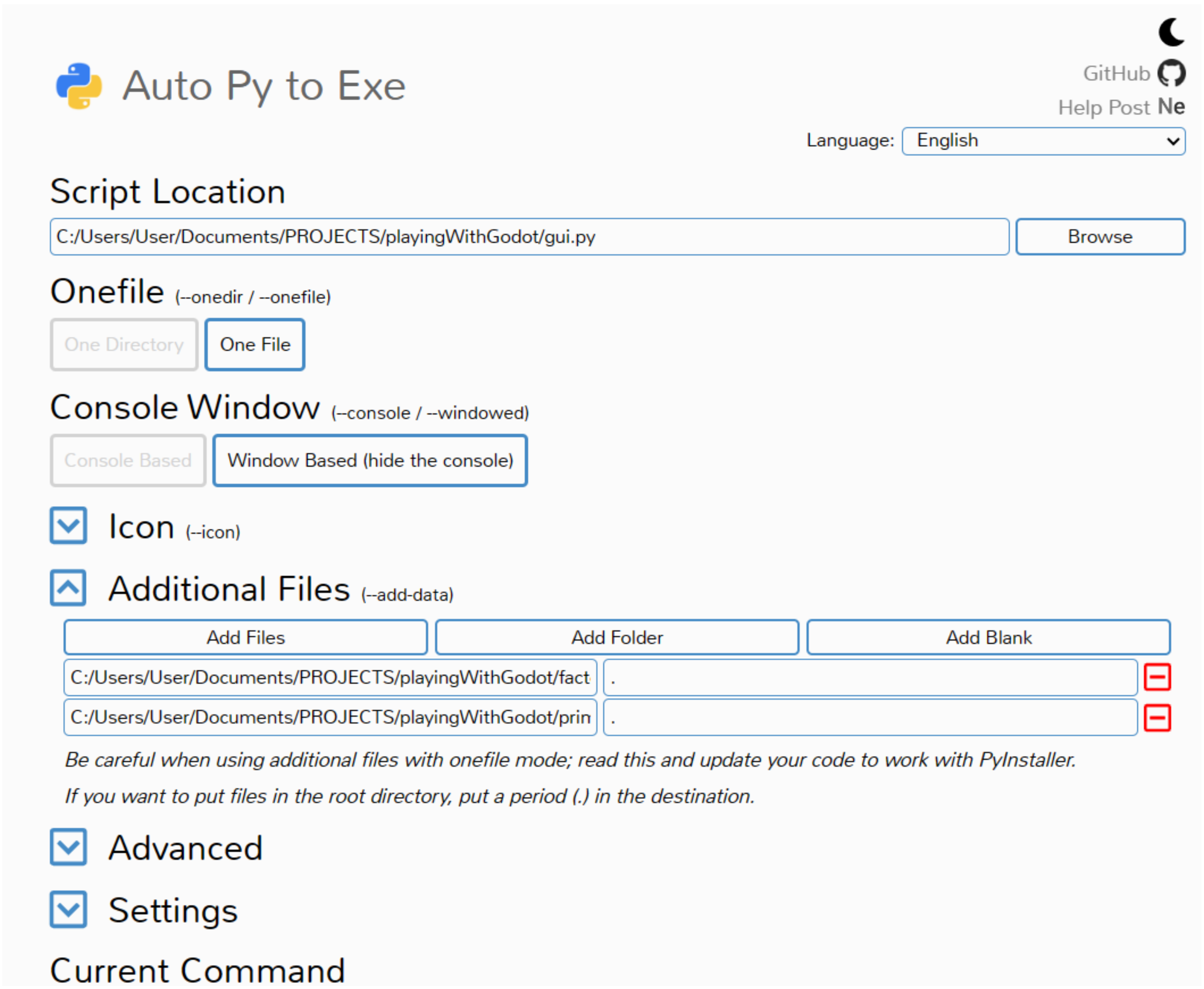
Prime error: Highest possible value reached.

Factorial error: Please input a number less than or equal to 1558.

Invalid

Executable File (.exe)

Running pyinstaller raises several errors during compilation, so I've settled on the auto-py-to-exe, which is essentially pyinstaller with a user interface. This allows me to compile all modules and dependencies in a single file.



Language: English

Script Location

C:/Users/User/Documents/PROJECTS/playingWithGodot/gui.py

Onefile (--onedir / --onefile)

One File

Console Window (--console / --windowed)

Window Based (hide the console)

Icon (--icon)

Additional Files (--add-data)

C:/Users/User/Documents/PROJECTS/playingWithGodot/fact	.
C:/Users/User/Documents/PROJECTS/playingWithGodot/prin	.

Be careful when using additional files with onefile mode; read this and update your code to work with PyInstaller.

If you want to put files in the root directory, put a period (.) in the destination.

Current Command

Pasted image 20240430013713.png