FINISHED_tu06_re_IfsLoopsFunctions_HW

January 30, 2024

Ifs, loops, and function homework

0.1 1. A function to reverse a string

Write and test a function that reverses a string entered by a user. This function will have one input value (a string) and one output value (also a string).

Test your function on, among other things, Napoleon's quote 'able was i ere i saw elba'

[1]:
```
napo_quote = "able was i ere i saw elba" #Just inputing the string varaible.
run_backwards = [] #This list is empty and used to store characters in the␣
 ↪initial string, but in reverse.
for i in reversed(napo_quote) : #Using a for loop and reversing it so it can␣
 ↪append i to the run_backwards.
    run_backwards.append(i)
run_backwards = ''.join(run_backwards) #After reversing it, it puts it into␣
 ↪strings.
print(run_backwards) #Just printing the backwards one.
```

```
able was i ere i saw elba
```

[2]:
```
rando_quote = "Hello World, today is Tuesday!" #Testing the same code on a␣
 ↪different string.
run_backwards = []
for i in reversed(rando_quote) :
    run_backwards.append(i)
run_backwards = ''.join(run_backwards)
print(run_backwards)
```

```
!yadseuT si yadot ,dlroW olleH
```

Optional challenge: run the above on "race car" and then fix the resulting string.

[ ]:
```

```

0.2 2. Determine if a number is prime

Write some code to test whether a number is prime or not, a prime number being an integer that is evenly divisible only by 1 and itself.

Hint: another way to think about a prime number is that, if the smallest number (other than 1) that divides evenly into a number *is* that number, than the number is a prime.

The easiest solution involves one `while` loop and one `if` test.

[4]:
```
def primeNumber(num) : # making the function.
    i = num - 1
    while i > 1:
        if num % i == 0:
            return False # saying that it is not a prime number.
        i = i -1
    return True #Saying that if it is a prime number it will be 'True'.
```

[6]:
```
print(primeNumber(5))
```

```
True
```

[7]:
```
print(primeNumber(4))
```

```
False
```

0.3 3. Find the first 10 primes

Extend your code above to find the first 10 prime numbers. This will involve wrapping your existing code in another "outer" loop.

[8]:
```
def primeNumber(num) : # making the function
    i = num - 1
    while i > 1:
        if num % i == 0:
            return False # saying that it is not a prime number
        i = i -1
    return True

def findingFirstPrimes(n):
    primes = []
    number = 3

    while len(primes) < n:
        if primeNumber(number):
            primes.append(number)
        number += 1
    return primes

firstTenPrimes = findingFirstPrimes(10)
print("These are the first ten prime numbers:", firstTenPrimes)
```

```
These are the first ten prime numbers: [3, 5, 7, 11, 13, 17, 19, 23, 29, 31]
```

0.4 4. Make a function to compute the first n primes

Functionalize (is that a word?) your above code. A user should be able to call your code with one integer argument and get a list back containing that number of primes. Make sure your function handles inputs of an incorrect type gracefully. You should also warn the user if they enter a really big number (which could take a long time…), and give them the option of either bailing or entering a different number.

[9]:
```
def primeNumber(num): #This first part checks if a 'num' is a prime number.
    i = num - 1
    while i > 1:
        if num % i == 0:
            return False
        i = i - 1
    return True

def findingFirstPrimes(n):
    primes = [] #Used to make an empty list.
    number = 3

    while len(primes) < n:
        if primeNumber(number):
            primes.append(number)
        number += 1
    return primes

def PrimeListGen():#The main function of the code.
    try:
        n = int(input("How many prime numbers do you want: "))
        if n <= 0:
            print("Enter a positive integer.")
            return []
        if n > 1000:
            response = input("Warning: It will take a long time to generate a␣
 ↪long list of primes. Do you wish to continue? (yes/no): ").lower()
            if response != 'yes':
                print("Canceled!")
                return []
        return findingFirstPrimes(n)
    except ValueError:
        print("Invalid. Enter a valid integer.")
        return []

if __name__ == "__main__":
    primeList = PrimeListGen()
    if primeList:
        print(f"Prime numbers: {primeList}")
```

```
How many prime numbers do you want: 7
Prime numbers: [3, 5, 7, 11, 13, 17, 19]
```

[10]:
```python
def primeNumber(num):
    i = num - 1
    while i > 1:
        if num % i == 0:
            return False
        i = i - 1
    return True

def findingFirstPrimes(n):
    primes = []
    number = 3

    while len(primes) < n:
        if primeNumber(number):
            primes.append(number)
        number += 1
    return primes

def PrimeListGen():
    try:
        n = int(input("How many prime numbers do you want: "))
        if n <= 0:
            print("Enter a positive integer.")
            return []
        if n > 1000:
            response = input("Warning: It will take a long time to generate a␣
 ↪long list of primes. Do you wish to continue? (yes/no): ").lower()
            if response != 'yes':
                print("Canceled!")
                return []
        return findingFirstPrimes(n)
    except ValueError:
        print("Invalid. Enter a valid integer.")
        return []

if __name__ == "__main__":
    primeList = PrimeListGen()
    if primeList:
        print(f"Prime numbers: {primeList}")
```

```
How many prime numbers do you want: b
Invalid. Enter a valid integer.
```