tu06_re_IfsLoopsFunctions_HW

March 9, 2023

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'

[3]:
```
def reverse() :
    string = input('Enter a string:')
    backwards = ''
    backwards_lst = []
    for i in reversed(string) :
        backwards_lst.append(i)
    backwards_lst = backwards.join(backwards_lst)
    print(backwards_lst)

reverse()
```

```
Enter a string:able was i ere i saw elba
able was i ere i saw elba
```

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

[4]:
```
reverse()
```

```
Enter a string:race car
rac ecar
```

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.

[5]:
```
def is_prime(num):
  if num ==2:
    print(num, 'is not a prime number')
  elif num > 1:
      for i in range(2, num):
        if (num % i) == 0:
          print(num, "is not a prime number")
          break
      else:
          print(num, "is a prime number")
  else:
      print(num, "is not a prime number")
```

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.

[6]:
```
for i in range(10):
    is_prime(i)
```

```
0 is not a prime number
1 is not a prime number
2 is not a prime number
3 is a prime number
4 is not a prime number
5 is a prime number
6 is not a prime number
7 is a prime number
8 is not a prime number
9 is not a prime number
```

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.

[61]:
```
def prime(num) :
    try:
        if num > 500:
            q = input('this number is quite big it\'ll take a while. Do you␣
 ↪want to quit?')
            if q.lower() == 'yes':
                print('Bye!')
                return
```

```
        else:
            prime_list = []
            for num in range(num + 1):
                if num > 2:
                    for i in range(2, num):
                        if (num % i) == 0:
                            break
                    else:
                        prime_list.append(num)
            print(prime_list)
    except TypeError as type_err:
        print('hey! looks like your input wasn\'t the right type.')
```

[64]:
```
prime(90)
```

```
[3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73,
79, 83, 89]
```