Project report

- Introduction
  Conclusion
  → have to be there

- Report Structure
  → Intro
  → Body of report
  → Conclusion

- Code Cells

(1) Good

Codes

Before starting the exercises and each parts of the project, I implemented a function to help me later on with my other functions in the projects.

The function is called `isprime`. This is a basic function that helps me to check if the number is a prime number or not. Any integer ≤ 1 is returned False immediately. To check is the number is prime or not, I took the $\sqrt{n}$ and checked if the number would be divisible by the numbers between 2 and $\sqrt{n}$, if yes, then the number given is not a prime number, if not, then the number given is a prime number

In [12]:
```
def isprime(n):
    # Function of checking if the number is prime, return Boolean

    prime = True
    if n<2:
        return False
    # To check if `n` is either 0,1 or a negative integer

    root = int(n**0.5)+1
    # The square root of `n` plus 1, used in the range in for-loop

    for d in range(2,root):
        if n%d == 0:
            prime = False
            # Change boolean prime to false if `n` is divisible by d

    return prime
```

With the `isprime` function, I now can later implement it in other functions that requires me to check if a number is a prime number or not.

(2) (comments are too long)

In [1]:
```
def isprime(n): # Define a function named isprime(n). I can use it to judge whether n is prime.
    prime = True # Given the prime is True at the beginning
    if n<2: # This is a condition. 2 is the smallest prime number. So, other prime numbers must be greater than or equal to 2. I
f n satisfies the condition, then it retuen result of next row.
        return False # If the number is smaller than 2. It's not a prime number. So print False
    for d in range(2,int(n**0.5)+1): # This is a loop to find whether a number is prime number. A non-prime number can be decomp
ose by other numbers.So n divided by d. d is every term from 2 to int(n**0.5). The range doesn't test the last number, so it add
s 1 in the last number. And that the maximum factor of n is not greater than int(n**0.5).
        if n%d==0: # It's also a condition. If remainder is zero, n is divisible by d. If the remainder is not zero, it will tes
t the next d value until it finish all tests.
            prime=False # When the remainder is equal to zero, n is not prime. Print False
            break # Break means stop running when the remainder is equal to zero. Because it doesn't need to test next d values.
n is not a prime number. It also can save time.

    return prime # Finally, it goes back to the prime. If n is a prime number, prime = True doesn't change. It can return Ture.
```

(3)

No text
No comments

In [1]:
```
def isprime(n): # The function test if n is a prime or not.
    prime=True
    if n<2:
        return False
    for d in range(2,int(n**0.5)+1):
        if n%d==0:
            prime=False
            break

    return prime
```

In [2]:
```
def myprimes(n): # The function return a list of primes which are less or equal than n.
    prime=[]
    for i in range(2,n+1):
        if isprime(i):
            prime.append(i)
    return prime
```

In [3]:
```
def primary(k): # The function return a list of decomposition of k.
    decomposition = []
    for d in myprimes(k):
        while k%d==0:
            k=k/d
            decomposition.append(d)
        if k==1:
            break
    return decomposition
```

- Comments

  def f(n):
  ' ~~~~~ ']

- References " ~~~~~ " [where you got it from]

- Running Time (Anything above 10 min's is sketchy)

- Audience / Definitions

  ↳ Roommate not in the class (w/ some progr. exp. some math knowledge)

- Keywords

In [4]:
```
def myprimes(x):
    myprimesList=[]
    for all in range(2, int(x+1)):
        if isprime(all)==True:
            myprimesList.append(all)
    return myprimesList
```

In [14]:
```
def isprimelike(p):
    sum=0
    a = list(range(0,p))
    for i in a:
        if pow(i,p,p)==i%p:
            sum=sum+0
        else:
            sum=sum+1
    if sum>0:
        return False
    else:
        return True
```

- Output / format / layout

~~The steps below should be done with a for loop~~

✗

```
In [103]: #Step 8: Now we will print out the primary decomposition for the 20 false primes
          primary_decomposition(falsePrimes[0])
Out[103]: [3, 11, 17]

In [104]: primary_decomposition(falsePrimes[1])
Out[104]: [5, 13, 17]

In [105]: primary_decomposition(falsePrimes[2])
Out[105]: [7, 13, 19]

In [106]: primary_decomposition(falsePrimes[3])
Out[106]: [5, 17, 29]

In [107]: primary_decomposition(falsePrimes[4])
Out[107]: [7, 13, 31]
```

✓

```
561 's prime decomposition is [3, 11, 17]
1105 's prime decomposition is [5, 13, 17]
1729 's prime decomposition is [7, 13, 19]
2465 's prime decomposition is [5, 17, 29]
2821 's prime decomposition is [7, 13, 31]
6601 's prime decomposition is [7, 23, 41]
8911 's prime decomposition is [7, 19, 67]
10585 's prime decomposition is [5, 29, 73]
15841 's prime decomposition is [7, 31, 73]
29341 's prime decomposition is [13, 37, 61]
41041 's prime decomposition is [7, 11, 13, 41]
46657 's prime decomposition is [13, 37, 97]
52633 's prime decomposition is [7, 73, 103]
62745 's prime decomposition is [3, 5, 47, 89]
63973 's prime decomposition is [7, 13, 19, 37]
75361 's prime decomposition is [11, 13, 17, 31]
101101 's prime decomposition is [7, 11, 13, 101]
115921 's prime decomposition is [13, 37, 241]
126217 's prime decomposition is [7, 13, 19, 73]
162401 's prime decomposition is [17, 41, 233]
```

for loop + str. formatting

✗

```
[[3, 11, 17], [5, 13, 17], [7, 13, 19], [5, 17, 29], [7, 13, 31], [7, 23, 41], [7, 19, 67], [5, 29, 73], [7, 31, 73], [13, 37,
61], [7, 11, 13, 41], [13, 37, 97], [7, 73, 103], [3, 5, 47, 89], [7, 13, 19, 37], [11, 13, 17, 31], [7, 11, 13, 101], [13, 37,
241], [7, 13, 19, 73], [17, 41, 233], [7, 13, 31, 61]]
```

- Errors

Kernel → Restart & Run All