

# Python Recap – II

Madhavan Mukund

<https://www.cmi.ac.in/~madhavan>

Programming, Data Structures and Algorithms using Python  
Week 1

# Checking primality

- A prime number  $n$  has exactly two factors, 1 and  $n$ 
  - Note that 1 is **not** a prime

# Checking primality

- A prime number  $n$  has exactly two factors, 1 and  $n$ 
  - Note that 1 is **not** a prime
- Compute the list of factors of  $n$

```
def factors(n):  
    fl = []    # factor list  
    for i in range(1,n+1):  
        if (n%i) == 0:  
            fl.append(i)  
    return(fl)
```

# Checking primality

- A prime number  $n$  has exactly two factors, 1 and  $n$ 
  - Note that 1 is **not** a prime
- Compute the list of factors of  $n$
- $n$  is a prime if the list of factors is precisely  $[1, n]$

```
def factors(n):  
    fl = []    # factor list  
    for i in range(1, n+1):  
        if (n%i) == 0:  
            fl.append(i)  
    return(fl)
```

```
def prime(n):  
    return(factors(n) == [1, n])
```

# Counting primes

- List all primes upto  $m$

```
def primesupto(m):  
    pl = []    # prime list  
    for i in range(1,m+1):  
        if prime(i):  
            pl.append(i)  
    return(pl)
```

# Counting primes

- List all primes upto  $m$
- List the first  $m$  primes
  - Multiple simultaneous assignment

```
def primesupto(m):  
    pl = []    # prime list  
    for i in range(1,m+1):  
        if prime(i):  
            pl.append(i)  
    return(pl)
```

```
def firstprimes(m):  
    (count,i,pl) = (0,1,[])  
    while (count < m):  
        if prime(i):  
            (count,pl) = (count+1,pl+[i])  
        i = i+1  
    return(pl)
```

# Counting primes

- List all primes upto  $m$
- List the first  $m$  primes
  - Multiple simultaneous assignment
- `for` vs `while`
  - Is the number of iterations known in advance?
  - Ensure progress to guarantee termination of `while`

```
def primesupto(m):  
    pl = []    # prime list  
    for i in range(1,m+1):  
        if prime(i):  
            pl.append(i)  
    return(pl)
```

```
def firstprimes(m):  
    (count,i,pl) = (0,1,[])  
    while (count < m):  
        if prime(i):  
            (count,pl) = (count+1,pl+[i])  
        i = i+1  
    return(pl)
```

# Computing primes

- Directly check if  $n$  has a factor between 2 and  $n - 1$

```
def prime(n):  
    result = True  
    for i in range(2,n):  
        if (n%i) == 0:  
            result = False  
    return(result)
```



# Computing primes

- Directly check if  $n$  has a factor between 2 and  $n - 1$
- Terminate check after we find first factor
  - Breaking out of a loop

```
def prime(n):  
    result = True  
    for i in range(2,n):  
        if (n%i) == 0:  
            result = False  
    return(result)
```

```
def prime(n):  
    result = True  
    for i in range(2,n):  
        if (n%i) == 0:  
            result = False  
            break    # Abort loop  
    return(result)
```

# Computing primes

- Directly check if  $n$  has a factor between 2 and  $n - 1$
- Terminate check after we find first factor
  - Breaking out of a loop
- Alternatively, use `while`

```
def prime(n):  
    result = True  
    for i in range(2,n):  
        if (n%i) == 0:  
            result = False  
            break    # Abort loop  
    return(result)
```

```
def prime(n):  
    (result,i) = (True,2)  
    while (result and (i < n)):  
        if (n%i) == 0:  
            result = False  
            i = i+1  
    return(result)
```

# Computing primes

- Directly check if  $n$  has a factor between 2 and  $n - 1$
- Terminate check after we find first factor
  - Breaking out of a loop
- Alternatively, use `while`
- Speeding things up slightly
  - Factors occur in pairs
  - Sufficient to check factors upto  $\sqrt{n}$
  - If  $n$  is prime, scan  $2, \dots, \sqrt{n}$  instead of  $2, \dots, n - 1$

```
import math
def prime(n):
    (result,i) = (True,2)
    while (result and (i <= math.sqrt(n))):
        if (n%i) == 0:
            result = False
        i = i+1
    return(result)
```

# Properties of primes

- There are infinitely many primes

# Properties of primes

- There are infinitely many primes
- How are they distributed?

# Properties of primes

- There are infinitely many primes
- How are they distributed?
- Twin primes:  $p, p + 2$

# Properties of primes

- There are infinitely many primes
- How are they distributed?
- Twin primes:  $p, p + 2$
- **Twin prime conjecture**  
There are infinitely many twin primes

# Properties of primes

- There are infinitely many primes
- How are they distributed?
- Twin primes:  $p, p + 2$
- **Twin prime conjecture**  
There are infinitely many twin primes
- Compute the differences between primes



# Properties of primes

- There are infinitely many primes
- How are they distributed?
- Twin primes:  $p, p + 2$
- **Twin prime conjecture**  
There are infinitely many twin primes
- Compute the differences between primes
- Use a dictionary
- Start checking from 3, since 2 is the smallest prime

# Properties of primes

- There are infinitely many primes
- How are they distributed?
- Twin primes:  $p, p + 2$
- **Twin prime conjecture**  
There are infinitely many twin primes
- Compute the differences between primes
- Use a dictionary
- Start checking from 3, since 2 is the smallest prime

```
def primediffs(n):  
    lastprime = 2  
    pd = {} # Dictionary for  
            # prime differences  
    for i in range(3,n+1):  
        if prime(i):  
            d = i - lastprime  
            lastprime = i  
            if d in pd.keys():  
                pd[d] = pd[d] + 1  
            else:  
                pd[d] = 1  
    return(pd)
```