Day objectives

- Loops Cont...
- Strings
- Data structures

In [5]:
```
## prime number
## A number which is divisible by one and itself

n = int(input('enter number: '))
f_c = 0
for i in range(1,n+1):
    if(n%i == 0):
        f_c = f_c+1 ## f_c += 1
if(f_c == 2):
    print('Prime number')
else:
    print('Not prime number')
```

```
enter number: 347
Prime number
```

Task

- Check the given number is perfect or not

In [7]:
```
## Table
## input: 5
## output:
# 5*1 = 5
# 5*2 = 10
# --
# --
# 5*10 = 50

t = int(input())
for i in range(1,11):
    print(t,'*',i,'=',t*i)
```

```
78
78 * 1 = 78
78 * 2 = 156
78 * 3 = 234
78 * 4 = 312
78 * 5 = 390
78 * 6 = 468
78 * 7 = 546
78 * 8 = 624
78 * 9 = 702
78 * 10 = 780
```

while loop

- which is used to repeat block of code into multiple times
- iterations are unknown
- execution speed is faster than for loop

Syntax:

initialization while(condition): statements/logic inc/dec

In [8]:
```
## n natural numbers

i = 1
while(i<=10): ## 1<=10 2<=10 3<=10  --- 11<=10
    print(i,end=' ')
    i = i+1 ## i += 1 # i = 2 i = 3 -- i =11
```

```
1 2 3 4 5 6 7 8 9 10
```

In [10]:
```
## Number of digits in the given number
## input: 2356
## OUTPUT: 4

num = int(input('enter a number: '))
d_c = 0
while(num!=0):
    num = num//10
    d_c += 1
print('Digit count is:',d_c)
```

```
enter a number: 23984756
Digit count is: 8
```

In [11]:
```
6/4
```

Out[11]:
```
1.5
```

In [14]:
```

2356//10
```

Out[14]:
```
235
```

In [15]:
```
235//10
```

Out[15]:
```
23
```

In [16]:
```
23//10
```

Out[16]:
```
2
```

In [17]:
```
2//10
```

Out[17]:
```
0
```

Task

- Caluculate digit sum in the given number
- input: 234
- output: 9

In [19]:
```
## Reverse of the given number
## input: 345
## output: 543

y = int(input())
rev = 0
while(y!=0):
    rem = y%10
    rev = rev*10+rem
    y = y//10

print(rev)
```

```
13255
55231
```

In [20]:
```
3//10
```

Out[20]:
```
0
```

Task

- Check the given number is palindrom or not
- input: 121
- output: 121
- Check the given number is armstrong or not
- input: 153 ## 1^3+5^3+3^3 == 153
- output: Armstrong

Nested for loop

- Loop inside loop
- Pattern solving
- Syntax:

  for variable in range(start,end,step):

  ```
  for variable in range(start,end,step):
      statements/logic
  ```

In [22]:
```
## Prime Numbers Series
## input: 1 -- 10
## output: 2 3 5 7

sr = int(input('enter start value: '))
er = int(input('enter end value: '))
for i in range(sr,er+1):  ### i = 1 2 3 4 5 6 7 8 9 10
    f_c = 0 ## i = 2
    for j in range(1,i+1): ## j =  1 ## range(1,3) -- j = 1 2
        if(i%j == 0): ## 1%1 == 0 ## 2%1 == 0 ## 2%2 == 0
            f_c += 1 ## f_c = 1 ## f_c = 1  ## f_c = 2
    if(f_c == 2):
        print(i,end=' ')
```

```
enter start value: 65
enter end value: 234
67 71 73 79 83 89 97 101 103 107 109 113 127 131 137 139 149 151 157 163 167 17
3 179 181 191 193 197 199 211 223 227 229 233
```

Task

- Print perfect number series
- input: 1-- 10
- output: 6

In [25]:
```
## square pattern

# @ @ @
# @ @ @
# @ @ @

for i in range(3):
    for j in range(3):
        print('@ ',end=' ')
    print()
```

```
@  @  @
@  @  @
@  @  @
```

In [16]:
```
# $ $ $
# $   $
# $ $ $

n = int(input())
for i in range(n):
    for j in range(n):
        if(i == 0 or j == 0 or i == n-1 or j == n-1):
            print('e ',end=' ')
        else:
            print('  ',end=' ')
    print()
```

```
5
e  e  e  e  e 
e           e 
e           e 
e           e 
e  e  e  e  e 
```

In [7]:
```
for i in range(3):
    for j in range(3):
        if(i==0 or j==0 or i==2 or j==2):
            print('$ ',end=' ')
            #print(i,j,end=' ')
        else:
            print('  ',end=' ')
    print()
```

```
$  $  $ 
$     $ 
$  $  $ 
```

Tasks

- Print W pattern

w w w w w w w w w w w w w

- Print Z pattern

Nested while loop

Syntax:

initilization while(condition): statements/logic initilization while(condition): statements/logic inc/dec inc/dec

In [17]:
```
i = 1
while i<=10:
    j = 1
    while j <= 10:
        print(i*j,end=' ')
        j += 1
    print()
    i += 1
```

```
1 2 3 4 5 6 7 8 9 10 
2 4 6 8 10 12 14 16 18 20 
3 6 9 12 15 18 21 24 27 30 
4 8 12 16 20 24 28 32 36 40 
5 10 15 20 25 30 35 40 45 50 
6 12 18 24 30 36 42 48 54 60 
7 14 21 28 35 42 49 56 63 70 
8 16 24 32 40 48 56 64 72 80 
9 18 27 36 45 54 63 72 81 90 
10 20 30 40 50 60 70 80 90 100 
```

jump statements

- We can call it as unconditional jumps
- break,continue and pass
- break
    - it will skip all iterations when the control reaches the break
    - break is a keyword
- continue
    - it skips only current iteration and continue with next iteration
    - continue is also a keyword
- pass
    - pass is a keyword is used to do nothing
    - when we need condition or function or any class syntactically correct but we do not want to do any operation
    - It is a null operation

In [19]:
```
for i in range(5):
    if i == 3:
        break
    print(i,end= ' ')
```

```
0 1 2 
```

In [20]:
```
for i in range(5):
    if i == 3:
        continue
    print(i,end= ' ')
```

```
0 1 2 4 
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Strings

- Collection of items or sequence of character or group of characters
- Which is derived data type
- we are simply create by enclosing characters in quotations(","")
- Strings are immutable(unchangable-once defined they cannot be change)

In [23]:

```
a = 'strings'
print(a)
a
type(a)
```

```
strings
```

Out[23]:

```
str
```

In [25]:

```
y = input()
print(y)
```

```
abcds
abcds
```

In [11]:

```
a = 'stringabc'

print(len(a))
print(min(a))
print(max(a))
print(sorted(a))
```

```
9
a
t
['a', 'b', 'c', 'g', 'i', 'n', 'r', 's', 't']
```

In [7]:

```
chr(65)
chr(98)
```

Out[7]:

```
'b'
```

```
In [9]: ord('a')

Out[9]: 97

In [12]: r = 'abc'
         y = 'xyz'
         r+y

Out[12]: 'abcxyz'

In [13]: r = 'cat'
         r*4

Out[13]: 'catcatcatcat'

In [14]: ## Indexing -- Syntax: str_name[interger(Position)]
         ## forward indexing starts with zero (Positive indexing)
         ## backward indexing starts with -1  (Negative indexing)

         r

Out[14]: 'cat'

In [15]: r[0]

Out[15]: 'c'

In [16]: r[1]

Out[16]: 'a'

In [17]: r[-1]

Out[17]: 't'

In [23]: ## Slicing
         ## Syntax: [start:end:step]
         s = 'python workshop'
         print(s[0:6:1])
         print(s[:6:])
         print(s[11:15])
         print(s[::])
         print(s[:15:2])

         python
         python
         shop
         python workshop
         pto okhp
```

In [30]:
```
## Negative slicing (right to left)

s = 'python workshop'
s[-6:-1]
s[-4:-1]
s[::-1]
```

Out[30]:
```
'pohskrow nohtyp'
```

In [ ]:

In [ ]: