to check given number within boundary or not

In [1]:
```
n=int(input())
for i in range(1,10):
    if i==n:
        print(i, 'exit')
        break
else:
    print('not exit')
```

```
11
not exit
```

In [2]:
```
n=int(input())
x=int(input())
y=int(input())
if n>=x and n<=y:
    print('exist')
else:
    print('not exist')
```

```
6
1
10
exist
```

In [7]:
```
# to check the give number factor or not
n=int(input())
for i in range(1,n+1):
    if n%i==0:
        print(i,end=' ')
```

```
6
1 2 3 6
```

In [19]:
```
n=int(input())
c=0
for i in range(1,n+1):
    if n%i==0:
        c+=1
if c==2:
    print('prime')
else:
    print('not prime')
```

```
8
not prime
```

day objectives

```
In [11]: 'value'

Out[11]: 'value'

In [18]: 'i don't know'

  File "<ipython-input-18-2e07b9b7946c>", line 1
    'i don't know'
           ^
SyntaxError: invalid syntax


In [12]: " i don't known"

Out[12]: " i don't known"

In [17]: """i am good
         gyvnnftvyu
         klhhkjuig """

Out[17]: 'i am good\ngyvnnftvyu\nklhhkjuig '
```

In [20]:
```
dir(list)
```

Out[20]:
```
['__add__',
 '__class__',
 '__contains__',
 '__delattr__',
 '__delitem__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getitem__',
 '__gt__',
 '__hash__',
 '__iadd__',
 '__imul__',
 '__init__',
 '__init_subclass__',
 '__iter__',
 '__le__',
 '__len__',
 '__lt__',
 '__mul__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__reversed__',
 '__rmul__',
 '__setattr__',
 '__setitem__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 'append',
 'clear',
 'copy',
 'count',
 'extend',
 'index',
 'insert',
 'pop',
 'remove',
 'reverse',
 'sort']
```

In [32]:
```
dir(int)
```

Out[32]:
```
['__abs__',
 '__add__',
 '__and__',
 '__bool__',
 '__ceil__',
 '__class__',
 '__delattr__',
 '__dir__',
 '__divmod__',
 '__doc__',
 '__eq__',
 '__float__',
 '__floor__',
 '__floordiv__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getnewargs__',
 '__gt__',
 '__hash__',
 '__index__',
 '__init__',
 '__init_subclass__',
 '__int__',
 '__invert__',
 '__le__',
 '__lshift__',
 '__lt__',
 '__mod__',
 '__mul__',
 '__ne__',
 '__neg__',
 '__new__',
 '__or__',
 '__pos__',
 '__pow__',
 '__radd__',
 '__rand__',
 '__rdivmod__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__rfloordiv__',
 '__rlshift__',
 '__rmod__',
 '__rmul__',
 '__ror__',
 '__round__',
 '__rpow__',
 '__rrshift__',
 '__rshift__',
 '__rsub__',
 '__rtruediv__',
 '__rxor__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__sub__',
 '__subclasshook__',
 '__truediv__',
 '__trunc__',
 '__xor__',
 'bit_length',
 'conjugate',
 'denominator',
 'from_bytes',
 'imag',
 'numerator',
 'real',
 'to_bytes']
```

In [33]:
```
int.__add__(2,4)
```

Out[33]:
```
6
```

In [ ]:
```
```

In [21]:
```
help([].append)
```

```
Help on built-in function append:

append(object, /) method of builtins.list instance
    Append object to the end of the list.
```

In [26]:
```
l=[1,2,3,4]
l.append(5)
print(l)
```

```
[1, 2, 3, 4, 5]
```

help,?

In [27]:
```
l?
```

function

In [31]:
```
def fun():# fun def ,user-define
    print('hai')# predefine
fun() # fun calling
```

```
hai
```

system defined

- input
- len()
- print()
- range()

user defined

- 1.requried argument functions
- 2.keyword argument functions
- 3.default argument functions
- 4.variable length argument functions

1.requried argument functions

In [34]:
```
def add(v1,v2):
    print(v1+v2)
add(1,2)
```

```
3
```

2.keyword argument functions

In [36]:
```
def add(v1=1,v2=2):
    print(v1+v2)
add()
```

```
3
```

3.default argument functions

In [37]:
```
def add(v1=1,v2=2):
    print(v1+v2)
add(10,20)
```

```
30
```

4.variable length argument functions

In [38]:
```
def add(v1,v2,v3):
    print(v1+v2)
add(10,20,3)
```

```
30
```

In [40]:
```
def add(*args):
    sum=0
    print(*args)
    for i in args:
        sum+=i
    print(sum)
add(1,2,3,4,5,7)
```

```
1 2 3 4 5 7
22
```

In [10]:
```
def isprime(n):
    c=0
    for i in range(1,n+1):
        if n%i==0:
            c+=1
    if c==2:
        return True
    else:
        return False
isprime(34)
```

Out[10]:
```
False
```

In [6]:
```
n=int(input())
c=0
for i in range(1,n+1):
    if n%i==0:
        print(i)
        c+=1
print("number of factor given number",c)
if c==2:
    print('prime')
else:
    print('not prime')
```

```
5
1
5
number of factor given number 2
prime
```

In [11]:
```
def prime(*args):
    for i in args:
        if isprime(i):
            print(i)

prime(1,2,67,89,65)
```

```
2
67
89
```

In [22]:
```
def isprime(*args):
    print(args)
    for i in args:
        c=0
        for k in range(2,i+1):
            if i%k==0:
                c+=1
        if c==1:
             print( i,'prime')
        else:
            print(i,'not prime')
isprime(1,2,3,4,5,6,7,8,9,10)
```

```
(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
1 not prime
2 prime
3 prime
4 not prime
5 prime
6 not prime
7 prime
8 not prime
9 not prime
10 not prime
```

regular expressions

- it's a way of checking data as your required
- it checks match or not
- valid or invalid

in this we have 3 types

- 1.search()
- 2.match()
- 3.findall()

In [25]:
```
import re
```

1.search

In [26]:
```
name="mounika"
re.search('a',name)
```

Out[26]:
```
<re.Match object; span=(6, 7), match='a'>
```

2.match

In [27]:
```
re.match('m',name)
```

Out[27]:
```
<re.Match object; span=(0, 1), match='m'>
```

3.findall

In [28]:
```
re.findall('m',name)
```

Out[28]:
```
['m']
```

special characters

- .
- ^
- $
- {}
- []

1.

In [30]:
```
name="1234567dsfd"
re.search(".....",name)
```

Out[30]:
```
<re.Match object; span=(0, 5), match='12345'>
```

In [31]:
```
data=["pavani","rani","kumar","sri"]
for i in data:
    if re.search("...",i):
        print(i)
```

```
pavani
rani
kumar
sri
```

^

In [32]:
```
name="dite"
re.search("^d",name)
```

Out[32]:
```
<re.Match object; span=(0, 1), match='d'>
```

$

In [34]:
```
name="diet"
re.search("t$",name)
```

Out[34]:
```
<re.Match object; span=(3, 4), match='t'>
```

{}

In [35]:
```
data=["pavani","rani","kumar","sri"]
for i in data:
    if re.search("^[a-z]{4}$",i):
        print(i)
```

```
rani
```

[]

In [36]:
```
data=["pavani","rani","kumar","sri"]
for i in data:
    if re.search("^[a-z]{5}$",i):
        print(i)
```

```
kumar
```

tuples

- tuples are immutable itarator
- collection of different data elements
- we can't modifiy,update,change,insert
- can only count and find the index of elements

In [37]:
```
t=(1,2,3,4,2)
t.count(2)
```
Out[37]:
```
2
```

In [38]:
```
t.index(2)
```
Out[38]:
```
1
```

In [45]:
```
t=(1,2,3,4,2)
del t
```

In [49]:
```
t1=("mounika",2,3,4,[4,5,6])
```

In [50]:
```
t1
```
Out[50]:
```
('mounika', 2, 3, 4, [4, 5, 6])
```

sets

- collection unique elements

In [52]:
```
s={1,2,3,4,5,6,7}
```

In [54]:
```
s
```
Out[54]:
```
{1, 2, 3, 4, 5, 6, 7}
```

In [55]:
```
s1={1,2,3,4}
s2={2,3,4,5}
print(s1)
print(s2)
```
```
{1, 2, 3, 4}
{2, 3, 4, 5}
```

In [56]:
```
s1&s2
```
Out[56]:
```
{2, 3, 4}
```

In [57]:
```
s1|s2
```
Out[57]:
```
{1, 2, 3, 4, 5}
```

In [58]:
```
s1.intersection(s2)
```
Out[58]:
```
{2, 3, 4}
```

In [61]:
```
s1.union(s2)
```
Out[61]:
```
{1, 2, 3, 4, 5}
```

In [62]:
```
s1-s2
```
Out[62]:
```
{1}
```

In [63]:
```
s2-s1
```
Out[63]:
```
{5}
```

In [64]:
```
s1
```
Out[64]:
```
{1, 2, 3, 4}
```

In [65]:
```
s2
```
Out[65]:
```
{2, 3, 4, 5}
```

dictionary

In [67]:
```
d={}
d=dict()
type(d)
```
Out[67]:
```
dict
```

In [71]:
```
d={"name":"mouni","rollno":15,"age":18}
```

In [72]:
```
d
```
Out[72]:
```
{'name': 'mouni', 'rollno': 15, 'age': 18}
```

In [74]:
```
d.keys()
```
Out[74]:
```
dict_keys(['name', 'rollno', 'age'])
```

In [75]:
```
d.values()
```
Out[75]:
```
dict_values(['mouni', 15, 18])
```

In [77]:
```
d["mani"]=24
```

In [78]:
```
d
```
Out[78]:
```
{'name': 'mouni', 'rollno': 15, 'age': 18, 'mani': 24}
```

In [79]:
```
len(d)
```
Out[79]:
```
4
```

In [80]:
```
d.items()
```
Out[80]:
```
dict_items([('name', 'mouni'), ('rollno', 15), ('age', 18), ('mani', 24)])
```

In [82]:
```
d["name"]
```
Out[82]:
```
'mouni'
```

creat one dict for 5 students and find topper of the class

In [87]:
```
d={"chinni":99,"rani":100,"siri":98,"mouni":90}
for key,val in d.items():
    if max(d.values())==val:
           print(key," is topper")
```
```
rani  is topper
```

In [90]:
```
d={"chinni":99,"rani":100,"siri":98,"mouni":0}
for key,val in d.items():
    if min(d.values())==val:
           print(key," is fail")
```
```
mouni  is fail
```

modules

In [ ]:
```

```

In [95]:
```
 import random
```

In [93]:
```
import math
```

In [94]:
```
import re
```

In [96]:
```
dir(math)
```

Out[96]:
```
['__doc__',
 '__loader__',
 '__name__',
 '__package__',
 '__spec__',
 'acos',
 'acosh',
 'asin',
 'asinh',
 'atan',
 'atan2',
 'atanh',
 'ceil',
 'copysign',
 'cos',
 'cosh',
 'degrees',
 'e',
 'erf',
 'erfc',
 'exp',
 'expm1',
 'fabs',
 'factorial',
 'floor',
 'fmod',
 'frexp',
 'fsum',
 'gamma',
 'gcd',
 'hypot',
 'inf',
 'isclose',
 'isfinite',
 'isinf',
 'isnan',
 'ldexp',
 'lgamma',
 'log',
 'log10',
 'log1p',
 'log2',
 'modf',
 'nan',
 'pi',
 'pow',
 'radians',
 'remainder',
 'sin',
 'sinh',
 'sqrt',
 'tan',
 'tanh',
 'tau',
 'trunc']
```

In [97]:
```
math.pi
```

Out[97]:
```
3.141592653589793
```

In [98]:
```
22/7
```

Out[98]:
```
3.142857142857143
```

```
In [99]: dir(random)
```

```
Out[99]: ['BPF',
          'LOG4',
          'NV_MAGICCONST',
          'RECIP_BPF',
          'Random',
          'SG_MAGICCONST',
          'SystemRandom',
          'TWOPI',
          '_BuiltinMethodType',
          '_MethodType',
          '_Sequence',
          '_Set',
          '__all__',
          '__builtins__',
          '__cached__',
          '__doc__',
          '__file__',
          '__loader__',
          '__name__',
          '__package__',
          '__spec__',
          '_acos',
          '_bisect',
          '_ceil',
          '_cos',
          '_e',
          '_exp',
          '_inst',
          '_itertools',
          '_log',
          '_os',
          '_pi',
          '_random',
          '_sha512',
          '_sin',
          '_sqrt',
          '_test',
          '_test_generator',
          '_urandom',
          '_warn',
          'betavariate',
          'choice',
          'choices',
          'expovariate',
          'gammavariate',
          'gauss',
          'getrandbits',
          'getstate',
          'lognormvariate',
          'normalvariate',
          'paretovariate',
          'randint',
          'random',
          'randrange',
          'sample',
          'seed',
          'setstate',
          'shuffle',
          'triangular',
          'uniform',
          'vonmisesvariate',
          'weibullvariate']
```

In [100]:
```
random.random()
```

Out[100]:
```
0.45522517380746963
```

In [102]:
```
random.randint(1,5)
```

Out[102]:
```
5
```

lists

- collection of elements
- mutable

In [132]:
```
l=[1,2,3,4,5,6,7,8,9]
```

In [104]:
```
l
```

Out[104]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [106]:
```
even=[]
odd=[]
for num in l:
    if num%2==0:
        even.append(num)
    else:
        odd.append(num)
print(even)
print(odd)
```

```
[2, 4, 6, 8]
[1, 3, 5, 7, 9]
```

In [109]:
```
x=[1,2,3,4]
l.extend(x)
```

In [110]:
```
l
```

Out[110]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 1, 2, 3, 4]
```

In [112]:
```
y=l
z=l.copy()
print(y)
print(z)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 1, 2, 3, 4]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 1, 2, 3, 4]
```

In [113]:
```
l.append('python')
print(y)
print(z)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 1, 2, 3, 4, 'python']
[1, 2, 3, 4, 5, 6, 7, 8, 9, 1, 2, 3, 4]
```

In [115]:
```
l.remove(2)
```

In [116]:
```
l
```

Out[116]:
```
[1, 3, 4, 5, 6, 7, 8, 9, 1, 2, 3, 4, 'python']
```

In [117]:
```
l.clear()
```

In [118]:
```
l
```

Out[118]:
```
[]
```

In [119]:
```
l=[1,2,3,4]
```

In [121]:
```
l.reverse()
```

In [122]:
```
l
```

Out[122]:
```
[4, 3, 2, 1]
```

In [123]:
```
l.sort()
```

In [129]:
```
l
```

Out[129]:
```
[1]
```

In [130]:
```
l.pop()
```

Out[130]:
```
1
```

In [133]:
```
l.pop()
```
Out[133]:
```
9
```

In [137]:
```
x=[1,1,2,2,2,3,4,5]
uniq=[]
d={}
for val in x:
    if val not in uniq:
        uniq.append(val)
for i in uniq:
    d[i]=x.count(i)
d
```
Out[137]:
```
{1: 2, 2: 3, 3: 1, 4: 1, 5: 1}
```

In [141]:
```
list(set(x))
```
Out[141]:
```
[1, 2, 3, 4, 5]
```

In [ ]: