In [1]:
```
# gh cli install
#gh auth login
#alternate use ssh key based
#gh repo clone
#git add
#git commit -m "message"
#git push
```

In [4]:
```
n = 10
sum_ = 0
for i in range(1,n+1):
    sum_ += i
    print(sum_)
```

```
1
3
6
10
15
21
28
36
45
55
```

In [8]:
```
n = 10
sum_ = 0
for i in range(1,n+1):
    sum_ += i
    print(sum_)
```

```
1
3
6
10
15
21
28
36
45
55
```

In [13]:
```
try:
    x = int(input("enter a number"))
    print("Your number is",x)
except:
        print("You did not enter a valid number,valid numbers can be: 10,20,
```

```
enter a number89
Your number is 89
```

In [ ]:
```
current_year = 2026

while True:
    try:
        birth_year = int(input("Enter your birth year: "))

        if birth_year <= 0:
            print("Birth year cannot be zero or negative. Please enter a val
        elif birth_year > current_year:
            print("Birth year cannot be in the future. Enter a valid year.")
        else:
            age = current_year - birth_year
```

```
            print("Your current age is:", age)
            break

    except ValueError :
        print("Invalid input Please enter numbers only.")
```

data structure vs data type

In [1]:
```
d = [10,1.0,"string",[1,2,3]]
for i in d:
    print(i)
```

```
10
1.0
string
[1, 2, 3]
```

In [3]:
```
for i in range(len(d)):
    print(d[i])
```

```
10
1.0
string
[1, 2, 3]
```

In [ ]:
```
# mutability
# string -> imutable
# list -> mutable
```

In [4]:
```
b = [1,2,3]
b[0] = "0"
b

#b is a list, and lists in Python are mutable (you can change their elements

b[0] refers to the first element of the list.

You replace 1 with the string "0"
```

Out[4]:
```
['0', 2, 3]
```

In [ ]:
```
# list can be updatd..
can I
- add two list?
-multiply with a list?
 what is
    -append method?
    -extend method?
```

In [6]:
```
a = [2,3,4]
b =[8,9,5]
c = a+b
print(c)
```

```
[2, 3, 4, 8, 9, 5]
```

In [9]:
```
a = [1, 2]
print(a * 3)
```

```
[1, 2, 1, 2, 1, 2]
```

In [12]:
```
a = [1,2,3,4]
a.extend("1234")
print(a)
```

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

In [14]:
```
a = [1,2,3]
b =[1,2, "test" , [1,2,3,4]]
a.extend(b)
print(a)
```

```
[1, 2, 3, 1, 2, 'test', [1, 2, 3, 4]]
```

In [15]:
```
for i in a:
    print(i)
```

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

In [16]:
```
for i in a:
    for j in i:
        print(j,ends=",")
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[16], line 2
      1 for i in a:
----> 2     for j in i:
      3         print(j,ends=",")

TypeError: 'int' object is not iterable
```

In [21]:
```
a = []
for i in range(3):
    b = []
for j in range(3):
    b.append(j)
    a.append(i)
print(b)
print(a)
```

```
[0, 1, 2]
[2, 2, 2]
```

In [26]:
```
#wap to take marks for 3 students as input from user and give the total sco
a=int(input("for 1st student:"))
b=int(input("for 2nd student:"))
c=int(input("for 3rd student:"))

total  = a+b+c
average_score = total/3

print("total marks",total)
print("average score",average_score)
```

```
for 1st student:3
for 2nd student:6
for 3rd student:9
total marks 18
average score 6.0
```

In [1]:
```
#there is a list with string s_list = [abc,bcd,bcdefg,abba,cddc,opq] impleme

s_list = ["abc", "bcd", "bcdefg", "abba", "cddc", "opq"]

def shortest_string(lst):
    shortest = lst[0]   # take first string as smallest

    for i in lst:
        if len(i) < len(shortest):
            shortest = i

    print("The shortest string:", shortest)

shortest_string(s_list)
```

```
The shortest string: abc
```

In [2]:
```
#space and time complexity
5 in [1,2,3,4]
```

Out[2]:
```
False
```

In [3]:
```
min(["a","abc","abab","def"])
```

Out[3]:
```
'a'
```

In [4]:
```
min(["z","abc","abab","def"],key=len)
```

Out[4]:
```
'z'
```

In [5]:
```
max("abcdefghA")
```

Out[5]:
```
'h'
```

In [6]:
```
min("abcdefghA")
```

Out[6]:
```
'A'
```

In [7]:
```
ord("A")
```

Out[7]:
```
65
```

In [8]:
```
ord("a")
```

Out[8]:
```
97
```

In [9]:
```
chr(97)
```

Out[9]:
```
'a'
```

In [11]:
```
#sortingg
my_list = [1,2,1,2,1,2,1,2,1,2,1,1,1,1,1,1,1,1]
```

In [12]:
```
dir(my_list)
```

Out[12]:
```
['__add__',
 '__class__',
 '__class_getitem__',
 '__contains__',
 '__delattr__',
 '__delitem__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getitem__',
 '__getstate__',
 '__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 [13]:
```
None
```

In [14]:
```
sorted([1,2,3,1,1,1])
```

Out[14]:
```
[1, 1, 1, 1, 2, 3]
```

In [15]:
```
a = []
b = a
b.append(5)
print(a)
```

```
[5]
```

In [16]:
```
a = []
b = a.copy()
b.append(5)
print(a)
```

```
[]
```

In [17]:
```
from copy import deepcopy
```

In [18]:
```
# tuple -> immutable list -> cannot be chnaged..
```

In [19]:
```
#make an empty tuple
a = ()
type (a)
```

Out[19]:
```
tuple
```

In [20]:
```
#make tuple with 1 element

b = (1)
type(b)
```

Out[20]:
```
int
```

In [21]:
```
a = 1,
type(a)
```

Out[21]:
```
tuple
```

In [1]:
```
a = 1
b = 2
b,a = a,b # variable = tuple
```

In [2]:
```
b,a = 1,2
```

In [24]:
```
a = (1,2,3, [1,2,3])
a[-1].append(4)
```

In [3]:
```
list = ['A','B','C','B','A','C']
repeated = []

for i in list:
    if i in repeated:
        print("First repeating element:", i)
        break
    else:
        repeated.append(i)
```

```
First repeating element: B
```

In [4]:
```
my_list = [1,2,3,4,5,6]
seen = []
for i in my_list:
    if i in seen:
        first_seen = 1
        break
    seen.append(i)
    i = None
    print(i)
```

```
None
None
None
None
None
None
```

In [5]:
```
{1,2,3,4,1,1,1,2,3,5,6,3,2,1}
```

Out[5]:
```
{1, 2, 3, 4, 5, 6}
```

In [6]:
```
a  = {1,2,3,4,1,1,1,2,3,5,6,3,2,1}
#order can be different
#has only unique items
type (a)
```

Out[6]:
```
set
```

In [7]:
```
1 in a # 0(1)
```

Out[7]:
```
True
```

In [8]:
```
dir(a)
```

Out[8]:
```
['__and__',
 '__class__',
 '__class_getitem__',
 '__contains__',
 '__delattr__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getstate__',
 '__gt__',
 '__hash__',
 '__iand__',
 '__init__',
 '__init_subclass__',
 '__ior__',
 '__isub__',
 '__iter__',
 '__ixor__',
 '__le__',
 '__len__',
 '__lt__',
 '__ne__',
 '__new__',
 '__or__',
 '__rand__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__ror__',
 '__rsub__',
 '__rxor__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__sub__',
 '__subclasshook__',
 '__xor__',
 'add',
 'clear',
 'copy',
 'difference',
 'difference_update',
 'discard',
 'intersection',
 'intersection_update',
 'isdisjoint',
 'issubset',
 'issuperset',
 'pop',
 'remove',
 'symmetric_difference',
 'symmetric_difference_update',
 'union',
 'update']
```

In [9]:
```
a = {1,2,3,4,3,6,2}
b = {4,8,9,6,8,4}
```

In [10]:
```
dir(a)
```

Out[10]:
```
['__and__',
 '__class__',
 '__class_getitem__',
 '__contains__',
 '__delattr__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getstate__',
 '__gt__',
 '__hash__',
 '__iand__',
 '__init__',
 '__init_subclass__',
 '__ior__',
 '__isub__',
 '__iter__',
 '__ixor__',
 '__le__',
 '__len__',
 '__lt__',
 '__ne__',
 '__new__',
 '__or__',
 '__rand__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__ror__',
 '__rsub__',
 '__rxor__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__sub__',
 '__subclasshook__',
 '__xor__',
 'add',
 'clear',
 'copy',
 'difference',
 'difference_update',
 'discard',
 'intersection',
 'intersection_update',
 'isdisjoint',
 'issubset',
 'issuperset',
 'pop',
 'remove',
 'symmetric_difference',
 'symmetric_difference_update',
 'union',
 'update']
```

In [11]:
```
#a and b both are set

## a.add()
   # update,
    #union,issubset,difference
     #a- b
```

```
 #a+b
#what are the data types i put in sets? is there any one i cannnot put?

#make a set of length 0/ make a s#et with no element
```

In [12]:
```
a.add(9)
print(a)
```

```
{1, 2, 3, 4, 6, 9}
```

In [13]:
```
# for multiple through tuple
a.update([20, 30])
print(a)
```

```
{1, 2, 3, 4, 20, 6, 9, 30}
```

In [14]:
```
a.union(b)
```

Out[14]:
```
{1, 2, 3, 4, 6, 8, 9, 20, 30}
```

In [15]:
```
a.issubset(b)
```

Out[15]:
```
False
```

In [16]:
```
a.difference(b)
```

Out[16]:
```
{1, 2, 3, 20, 30}
```

In [17]:
```
## list,dictionary,set are not allowed
## tuple,int ,float,string,boolean

#list is mutable whereas tuple is imutable
#list can be updated
```

In [18]:
```
s = set()
print(s)
print(len(s))
```

```
set()
0
```

In [20]:
```
#{1, 2, 3}            # int
#{1.5, 2.6}           # float
#{"a", "b"}           # string
#{True, False}        # boolean
#{(1, 2), (3, 4)}     # tuple
#  allowed data types
```

In [21]:
```
a.update(['c', 'b'])
print(a)
```

```
{1, 2, 3, 4, 'c', 6, 9, 'b', 20, 30}
```

In [22]:
```
a.update([3.9, 7.3])
print(a)
```

```
{1, 2, 3, 4, 'c', 6, 3.9, 7.3, 9, 'b', 20, 30}
```

In [23]:
```
a = 1
b = 3
a | b
```

Out[23]:
```
3
```

In [24]:
```
a = 1
b = 3
a & b
```

Out[24]:
```
1
```

In [25]:
```
"0001"
"0010"
#| -> bitwise or
#& -> bitwise and
```

Out[25]:
```
'0010'
```

In [26]:
```
a = {1,2,3}
b = {3,4,5}
a & b
```

Out[26]:
```
{3}
```

In [27]:
```
a | b
```

Out[27]:
```
{1, 2, 3, 4, 5}
```

In [28]:
```
# hash
# magic function that takes input
# and gives random
```

In [29]:
```
a = [1,2,3]
id(a)
```

Out[29]:
```
132956955363456
```

In [30]:
```
#empty set
a = []
type (a)
```

Out[30]:
```
list
```

In [31]:
```
a = ()
type (a)
```

Out[31]:
```
tuple
```

In [32]:
```
a = {}
type (a)
```

Out[32]:
```
dict
```

In [33]:
```
a = set ()
type (a)
```

Out[33]:
```
set
```

In [35]:
```
#dictionary
a = {1: 1,2: 4,3: 9,4: 16}
```

In [36]:
```
a[2]
```

Out[36]:
```
4
```

In [37]:
```
# what can be put in key of dictionary?
# key hasable

a ={1:1, "1":1,False:1,(1,2,3):1}
print(a)
```

```
{1: 1, '1': 1, False: 1, (1, 2, 3): 1}
```

In [38]:
```
## Unhashable objects are mutable objects and cannot be used in sets or as c
```

In [39]:
```
marks = {
    "ram":{"english":100,"science":98,"maths":87}

}
```

In [40]:
```
print("""I am sad""")
```

```
I am sad
```

In [41]:
```
print("I am""this")
```

```
I amthis
```

In [42]:
```
a = [

    "ram"
    "hari",
    "sita"
]
a
```

Out[42]:
```
['ramhari', 'sita']
```

In [43]:
```
len(a)
```

Out[43]:
```
2
```

In [44]:
```
#list of even numbers
even = [i for i in range(10) if i % 2==0]
even
```

Out[44]:
```
[0, 2, 4, 6, 8]
```

In [45]:
```
a = []
a = []
for i in range(10):
    for j in range(10):
        if i>j:
            a.append((i,j))
a
```

Out[45]:
```
[(1, 0),
 (2, 0),
 (2, 1),
 (3, 0),
 (3, 1),
 (3, 2),
 (4, 0),
 (4, 1),
 (4, 2),
 (4, 3),
 (5, 0),
 (5, 1),
 (5, 2),
 (5, 3),
 (5, 4),
 (6, 0),
 (6, 1),
 (6, 2),
 (6, 3),
 (6, 4),
 (6, 5),
 (7, 0),
 (7, 1),
 (7, 2),
 (7, 3),
 (7, 4),
 (7, 5),
 (7, 6),
 (8, 0),
 (8, 1),
 (8, 2),
 (8, 3),
 (8, 4),
 (8, 5),
 (8, 6),
 (8, 7),
 (9, 0),
 (9, 1),
 (9, 2),
 (9, 3),
 (9, 4),
 (9, 5),
 (9, 6),
 (9, 7),
 (9, 8)]
```

In [46]:
```
#list comprehension
a =[(i,j)for i in range(10) for j in range(10) if i>j]

a
```

Out[46]:
```
[(1, 0),
 (2, 0),
 (2, 1),
 (3, 0),
 (3, 1),
 (3, 2),
 (4, 0),
 (4, 1),
 (4, 2),
 (4, 3),
 (5, 0),
 (5, 1),
 (5, 2),
 (5, 3),
 (5, 4),
 (6, 0),
 (6, 1),
 (6, 2),
 (6, 3),
 (6, 4),
 (6, 5),
 (7, 0),
 (7, 1),
 (7, 2),
 (7, 3),
 (7, 4),
 (7, 5),
 (7, 6),
 (8, 0),
 (8, 1),
 (8, 2),
 (8, 3),
 (8, 4),
 (8, 5),
 (8, 6),
 (8, 7),
 (9, 0),
 (9, 1),
 (9, 2),
 (9, 3),
 (9, 4),
 (9, 5),
 (9, 6),
 (9, 7),
 (9, 8)]
```

In [47]:
```
#Write a list comprehension to create a list of squares for numbers from 1 t
squares = [i*i for i in range(1, 11)]
print(squares)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

In [48]:
```
squares = []

for i in range(1, 11):
    squares.append(i * i)

print(squares)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

In [50]:
```
##Use a list comprehension to create a list of even numbers from the list nu
#7, 8, 9, 10].
```

In [51]:
```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

even_numbers = []
for n in numbers:
    if n % 2 == 0:
        even_numbers.append(n)

print(even_numbers)
```

```
[2, 4, 6, 8, 10]
```

In [52]:
```
even_numbers = [n for n in numbers if n % 2 == 0]
print(even_numbers)
```

```
[2, 4, 6, 8, 10]
```

In [53]:
```
## 14. Write a dictionary comprehension to create a dictionary where the key
##and the values are the cubes of the keys.
cubes = {x: x**3 for x in range(1, 6)}
print(cubes)

#{} this represent dictionary in python
```

```
{1: 1, 2: 8, 3: 27, 4: 64, 5: 125}
```

In [54]:
```
#15. Use a set comprehension to create a set of unique vowels from the strin
#"comprehensions are powerful".
sentence = "comprehensions are powerful"

vowels = {ch for ch in sentence if ch in "aeiou"}

print(vowels)
```

```
{'e', 'u', 'o', 'i', 'a'}
```

In [55]:
```
sentence = "comprehensions are powerful"

vowels = set()
for ch in sentence:
    if ch in "aeiou":
        vowels.add(ch)

print(vowels)
```

```
{'e', 'u', 'o', 'i', 'a'}
```

In [56]:
```
##16. Given a dictionary students = {'Alice': 85, 'Bob': 78, 'Charlie': 92,
##comprehension to create a new dictionary with students who scored above 80

a = [(1,2),(1,2),(1,2)]

for i,j in a:
    print(i,j)
```

```
1 2
1 2
1 2
```

In [57]:
```
my_list = [1,2,3,4,5,1,2,3]

#create a new list that says if the next element in list is higher or lower
```

In [58]:
```
#list(enumerate(my_list))
#comp = []
#for i,val in enumerate(my_list[:1],1):
```

```
 #   to_append = "l" if val > my-list[i] else "h"
  #  comp.append(to_append)
#comp
```

In [59]:
```
my_list = [1,2,3,4,5,1,2,3]
[
  "l" if curr > nxt else "h"
   for curr,nxt in zip(my_list, my_list[1:])
]
```

Out[59]:
```
['h', 'h', 'h', 'h', 'l', 'h', 'h']
```

In [60]:
```
print("a","b","c","d",1,2,3,4,5)
```

```
a b c d 1 2 3 4 5
```

In [64]:
```
vowels = "aeiou"
numbers1 = [1,5,9,15,21]
numbers2 = [1,2,3,4,5]
#create two dictionary where elements are
#key -> vowel ,value -> correspondig value from number1
#key -> vowel ,value -> correspondig value from number2
```

In [65]:
```
vowels = "aeiou"
numbers1 = [1,5,9,15,21]
numbers2 = [1,2,3,4,5]
{k: v for k,v in zip(vowels,numbers1)}
```

Out[65]:
```
{'a': 1, 'e': 5, 'i': 9, 'o': 15, 'u': 21}
```

In [66]:
```
dict(zip(vowels,numbers1))
```

Out[66]:
```
{'a': 1, 'e': 5, 'i': 9, 'o': 15, 'u': 21}
```

In [68]:
```
f =open("a.txt","w")
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: