Silicon
...beyond teaching

List, Tuple ,Dictionary and String

Pradipta Kumar Pattanayak

Silicon Institute of Tecnology

ppattanayak@silicon.ac.in

Sequence

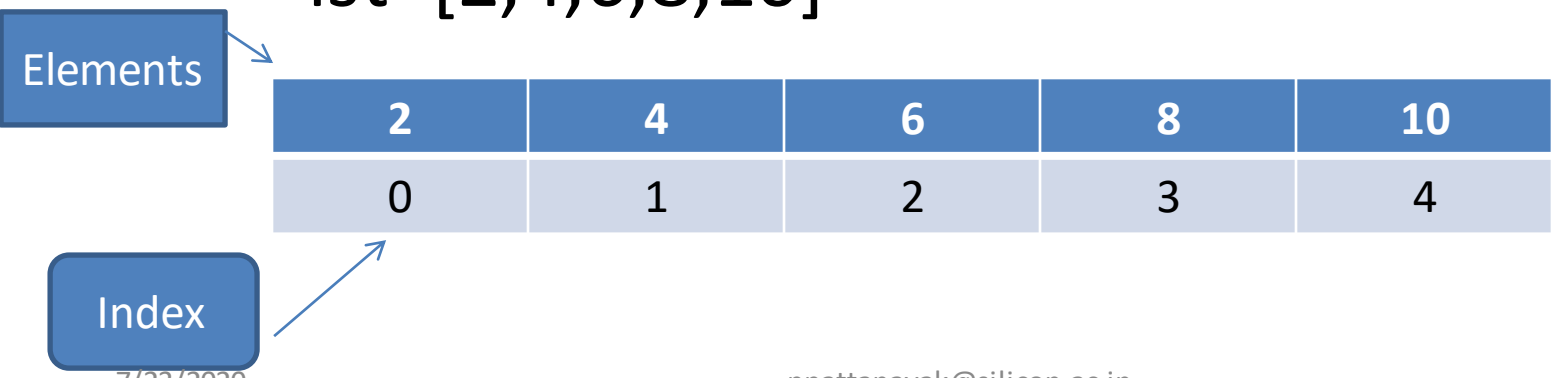
- The most basic data structure in Python is the **sequence**.
- Each element of a sequence is assigned a number - its position or index.
- The first index is zero, the second index is one, and so forth.
- Python has various built-in types of sequences, but the most commonly used are list, tuple and string.

List

- The data structure list is an ordered sequence which is mutable and made up of one or more elements.
- **Example**:

List of even numbers between 1 and 11:

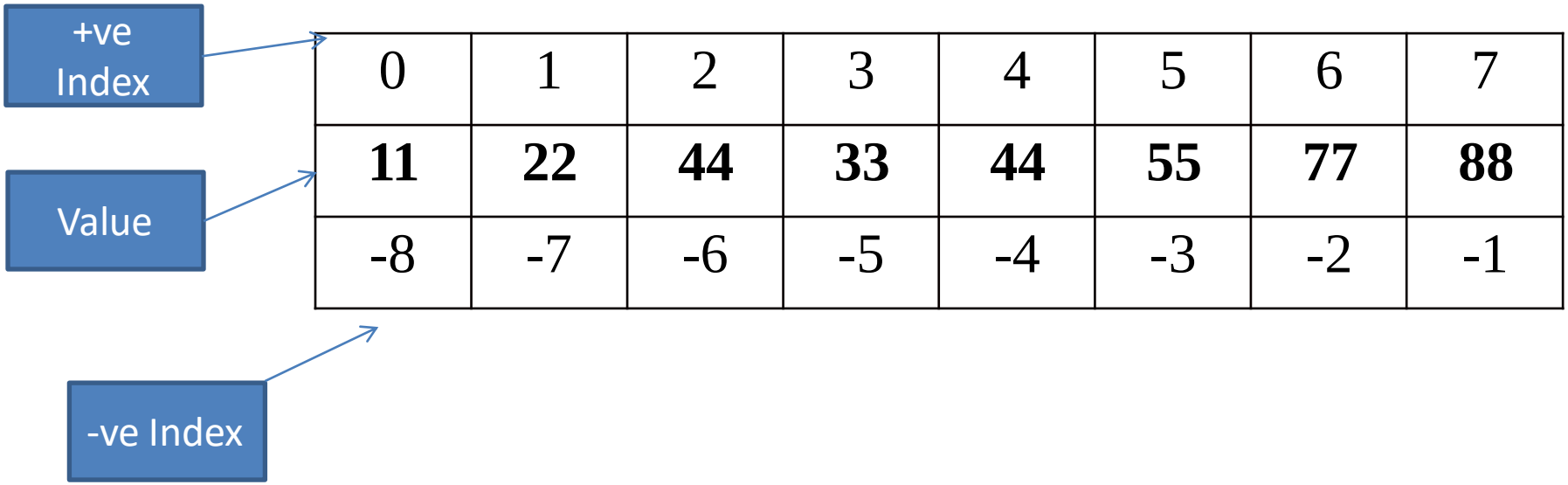
lst=[2,4,6,8,10]



ppattanayak@silicon.ac.in

Cont...

- A list can store element of different types:
- **Example**: lst=[“Amar”,”Amit”,”Rahul”,”Raja”,55,77,88,76]
- **Create an empty list**: lst=[ ]
- **Representation of List:**



ppattanayak@silicon.ac.in

Accessing list elements

0	1	2	3	4	5	6	7
11	22	44	33	44	55	77	88
-8	-7	-6	-5	-4	-3	-2	-1

- To access an element, use square brackets with the index of that element.

  lst[0]: Fetch the first element of the list(i.e 11)

  lst[5]: Fetch the 6th elements of the list (i.e 44)

- We may also use negative index value to access elements starting from the last element in the list, having index value -1.

  lst[-1]: Fetch the last element of the list (i.e 88)

  lst[-4] : Fetch the 4th elements of the list from right. (i.e 44)

ppattanayak@silicon.ac.in

Slicing of List

Syntax: **[lb: ub: step]** # Fetch the element **lb to ub-1**

Example:

a=list(range(4,35,2))

print(a)

output: **[4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34]**

print(a[:]) or print(a[::])

output: [4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34]

print(a[2::]) or print(a[2:])

output: [8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34]

print(a[2:5:]) or print(a[2:5])

output: [8, 10, 12]

print(a[2:20:2])

output: [8, 12, 16, 20, 24, 28, 32]

Cont...

print(a[-1]) : Fetch the last element

print(a[:-1])

output: [4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32]

print(a[-4:-1])

output: [28, 30, 32]

print(a[-4:-8:-2])

output: [28, 24]

print(a[::-1]) # print the list in reverse order

Traversing list

- We can access each element of the list or traverse a list using a for loop or a while loop.

Example:

```
myList = ['Red','Green','Blue','Yellow','Black']
for item in myList:
    print(item,end=" ")
```

OR (use index)

```
for i in range(len(myList )):
    print(myList[i],end=" " )
```

output: **Red Green Blue Yellow Black**

Write a program that will create a list of N numbers and search an element using linear search.

```
#Initialize the list...
arr=[ ]  #Create an empty list
num=int( input("How many elements you want to process?") )

#Initialization, by user input...
for i in range(0,num):
    arr.append( int( input("Enter an element") ))
print(arr)
```

Cont...

```
flag=0 #Indicate element not present...
item=int( input("enter a number"))  # read the item to be searched
for i in range(0,len(arr)):
    if item == arr[i]:
        flag=1 #Element found break...
        break

if flag==1:
    print("The element ", item," Present in index number ",i)
else:
    print("The element ", item," is not present in the list")
```

List Comprehension

List comprehensions provide a concise way to create lists.

It consists of brackets containing an expression followed by a for clause, then zero or more for or if clauses. The expressions can be anything, meaning you can put in all kinds of objects in lists.

Synatx: [expression for item in list if conditional]

Which is equvalent to :

```
for item in list:
    if conditional:
        expression
```

Example: Create a list of even number with specific range.

```
number_list = [ x for x in range(1,20) if x % 2 == 0]
print(number_list) # output: [ 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

Example: Create a binary list

```
number_list = [ x%2 for x in range(1,10) ]
print(number_list)# output: [1, 0, 1, 0, 1, 0, 1, 0, 1]
```

Cont...

Example: Create a list which will display the square of the number from 0 to 10.

```
squares = [i * i for i in range(11)]
print(squares)
```

output: [0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

Show the first letter of each word:

```
listOfWords = ["This","is","a","list","of","words"]
items = [ word[0] for word in listOfWords ]
print(items)
```

output: ['T', 'i', 'a', 'l', 'o', 'w']

list comprehension in functions:

```
def fun(x):
    return x*x
v=[fun(x) for x in range(10)]
print(v)
```

Intializing the list

```
l=[]
#intialze by using list comprehension...
l= [ int(x) for x in input("Enter number:").split()]
print(l)
```

output:

```
Enter number:11 22 33 44 55 66
[11, 22, 33, 44, 55, 66]
```

Basic list operation

Silicon ...beyond teaching

Python Expression	Results	Description
len([11, 22, 33,34])	4	Length of the list
[1, 2, 3,4,5] + [4, 5, 6,7]	[1, 2, 3, 4, 5, 4, 5, 6, 7]	Concatenation of two list
['Hi!'] * 3	['Hi!', 'Hi!', 'Hi!']	3 times repeated
2 in [1, 2, 3,4,5]	True	Membership
2 not in [1, 2, 3,4,5]	False	Check Membership
for x in [11, 22, 33]: print x,	11 22 33	Iteration

ppattanayak@silicon.ac.in

Cont...

Sr.No.	Function with Description
1	len(list) : Returns the length of the list.
2	max(list) : Returns item from the list with max value.
3	min(list) : Returns item from the list with min value.

Built-in functions for list manipulation

Sr.No.	Methods with Description
1	**append(self, object, /):** Append object to the end of the list. **Example**: lst=[1,2,3] lst.append(4) print(lst) **Output**: **[1, 2, 3, 4]**
2	**clear(self, /) :** Remove all items from list
3	**copy(self, /) :** Return a shallow copy of the list.
4	**count(self, value, /): Return number of occurrences of value**

ppattanayak@silicon.ac.in

Cont...

Sr.No.	Methods with Description

5	extend(self, iterable, /)
	Extend list by appending elements from the iterable.
	Example:
	l1=[1,2,3]
	l2=[4,5,6]
	l1.extend(l2) #list l1 get extended
	print(l1)
	output: [1, 2, 3, 4, 5, 6]

6	list.index(obj) : Returns the lowest index in list that obj appears. Example: l1=[1, 2, 3, 4, 2, 6, 2] print (l1.index(2)) Output: 1

Cont...

Sr.No.	Methods with Description
7	insert(self, index, object, /) : Insert object on a specific index **Example**: l1=[11, 22, 33, 44, 66] l1.insert(4,55) print(l1) **output**: [11, 22, 33, 44, 55, 66]
8	pop(self, index=-1, /) Remove and return item at index (default last). Raises IndexError if list is empty or index is out of range.

Cont...

Sr.No.	Methods with Description
9	remove(self, value, /) Remove first occurrence of value. Raises ValueError if the value is not present.
10	reverse(self, /) : Reverses objects of list in place

Cont...

Sr.No.	Methods with Description
11	**sort(self, /, *, key=None, reverse=False)** Stable sort *IN PLACE*. Example-1: l=[11,44,22,33,66] l.sort() print(l) **output**:[11, 22, 33, 44, 66] Example-2 l=[11,44,22,33,66] l.sort(reverse=True) print(l) **output**: [66, 44, 33, 22, 11]

List of List

- We can have list of list:
- **Example:**

```
myList=[ [11,22,33],[44,55,66] ,[77,88,99]]
print(myList)  #[[11, 22, 33], [44, 55, 66], [77, 88, 99]]
print(myList[1]) # [44, 55, 66]
print(myList[2][2])  #99
```

- **Iterate the list:**

```
for i in range ( len(myList)):
    for j in range ( len(myList[i] ) ):
        print(myList[i][j], end=" ")
    print()
```

Output:
11 22 33
44 55 66
77 88 99

ppattanayak@silicon.ac.in

Sorting

- Arranging the elements in order.
- **Example**:

```
arr = [0, 11, 4, 9, 116, 25, 36, 249, 64, 81, 100]
print("Before Sorting: ",arr)
arr.sort()
print("After Sorting: ",arr)
arr.sort(reverse=True)
print(“Reverse Sorting: ”, arr)
```

output:

```
Before Sorting:  [0, 11, 4, 9, 116, 25, 36, 249, 64, 81, 100]
After Sorting:  [0, 4, 9, 11, 25, 36, 64, 81, 100, 116, 249]
Reverse Sorting : [249, 116, 100, 81, 64, 36, 25, 11, 9, 4, 0]
```

Bubble Sort.

```
myList=[22,44,11,23,55,67,9,29,22]  # You can take user input
print("Before Sorting: ", end="")
print(myList)
for i in range(len(myList)-1):
    for j in range(len(myList)-i-1):
        if myList[j]>myList[j+1]:
            temp=myList[j]
            myList[j]=myList[j+1]
            myList[j+1]=temp
print("After Sorting: ", end="")
print(myList)
```

output:

Before Sorting: [22, 44, 11, 23, 55, 67, 9, 29, 22]

After Sorting: [9, 11, 22, 22, 23, 29, 44, 55, 67]

Insertion Sort

```
arr=[22, 44, 11, 23, 55, 67, 9, 29, 10]
print("Before Sorting: ", end="")
print(arr)
for j in range(1, len(arr)):
    key = arr[j]
    i = j-1
    while i >=0 and arr[i] >key:
        arr[i+1] = arr[i]
        i=i-1
    arr[i+1] = key
print("After Sorting: ", end="")
print(arr)
```

output:
Before Sorting: [22, 44, 11, 23, 55, 67, 9, 29, 10]
After Sorting: [9, 10, 11, 22, 23, 29, 44, 55, 67]

Tuple

- A tuple is an ordered sequence of elements of different data types, such as integer, float, string, list or even a tuple.
- Elements of a tuple are enclosed in parenthesis (round brackets) and are separated by commas.
- Like list, elements of a tuple can be accessed using index values, starting from 0.
- Tuple is an **immutable**. It means that the elements of a tuple cannot be changed after it has been created.

Cont...

Example: Tuple of integer.

T=(1,2,3,4)

Acess the touple elements: T[1]

T[2]=5 #error, because touple in immutable

.

Example: Tuple of mixed data type.

Tp=(“Math”,81,”Comp”,82,”Phy”,78)

Built-in Tuple Functions:

Silicon ...beyond teaching

len(tuple)

Gives the length of the tuple.

max(tuple)

Returns item from the tuple with max value

min(tuple)

Returns item from the tuple with min value

Basic operation

Python Expression	Results	Description
len((1, 2, 3,4,5))	5	length
(0,1, 2, 3) + (4, 5, 6,7)	(0,1, 2, 3, 4, 5, 6,7)	concatenation
('pkp') * 3	('pkp','pkp','pkp'')	repetition
3 in (1, 2, 3)	True	membership
for x in (1, 2, 3): print x	1 2 3	iteration

Basic Function

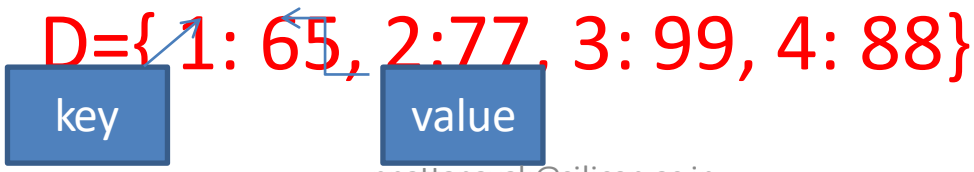
Methods	Description
count()	**count(self, value, /):** Return number of occurrences of value. **Example**: T=(1,2,3,2,4,5,2) print(T.count(2)) **#output** 3
index()	**index(self, value, start=0, stop=9223372036854775807, /)** Return first index of value. Raises ValueError if the value is not present. **Example:** print(T.index(2)) #**Output** 1

Where to use tuple

- If you have data that doesn't change, implementing it as tuple will guarantee that it remains write-protected.

Dictionary

- Dictionary is an unordered collection of key-value pairs.
- It is generally used when we have a huge amount of data.
- Dictionaries are optimized for retrieving data. We must know the key to retrieve the value.
- Dictionaries are defined within braces **{ }** with each item being a pair in the form **key:value**.
- Key and value can be of any type.
- **Example**: **<stdu-Roll, avg-mark>**



Create Dictionary

- **Empty dictionary :**

  dic={} #initialize empty dictionary

- **Note:** Keys can be any immutable values: int, float, bool, string, tuple but not list or dictionary.
- **Example:**

```
released = {
   "iphone" : 2007, "iphone 3G" : 2008,
   "iphone 3GS" : 2009, "iphone 4" : 2010,
   "iphone 4S" : 2011,"iphone 5" : 2012
   }
print( released)
```

output:{'iphone': 2007, 'iphone 3G': 2008, 'iphone 3GS': 2009, 'iphone 4': 2010, 'iphone 4S': 2011, 'iphone 5': 2012}

Acess Dictionary

- **released={'iphone': 2007, 'iphone 3G': 2008, 'iphone 3GS': 2009, 'iphone 4': 2010, 'iphone 4S': 2011, 'iphone 5': 2012}**
- We use key to retrieve the respective value.
- print(released ['iphone']) # output: 2007
- print(released ['iphone 4S']) # output: 2011
- **Iterating the dictionary:**

```
for i in release:
    print(i," : ",release[i], end=", ")
```

output: iphone : 2007, iphone 3G : 2008, iphone 3GS : 2009, iphone 4 : 2010, iphone 4S : 2011, iphone 5 : 2012,

Print all keys an value:

```
for key, value in released.items():
    print(k," : ",value,end=" " )
```

output: iphone : 2007 iphone 3G : 2008 iphone 3GS : 2009 iphone 4 : 2010 iphone 4S : 2011 iphone 5 : 2012

Update Dictionary

- **Example:** stud={ 1: 66, 2: 55, 3: 88, 4: 60,11:98}

```
stud[5]=74
print(stud) # {1: 66, 2: 55, 3: 88, 4: 60, 11: 98, 5: 74}
#modify an elements in a dictionary…
stud[2]=75  #key is immutable
print(stud)  #{1: 66, 2: 75, 3: 88, 4: 60, 11: 98, 5: 74}
print( len(stud) )  # 6 -Length of dictionary...
dict(stud) #{1: 66, 2: 75, 3: 88, 4: 60, 11: 98, 5: 74}
```

Delete element from Dictionary:

```
del(dict[3])      # remove the element having key 3
del(dic)          # delete the dictionary
```

Dictionary comprehension:

- Dictionary comprehension expressions are used to create a new dictionary at the runtime from another iterator object.
- **Syntax: {key: value for (key, value) in iterable}**
- **Example:**

```
tuple_1 = (1, 2, 3, 4, 5, 6, 7, 8, 9)
dict_1 = {item: item * 2 for item in tuple_1}
print(dict_1)
```

output:{1: 2, 2: 4, 3: 6, 4: 8, 5: 10, 6: 12, 7: 14, 8: 16, 9: 18}

Nested Dictionary

Example:

```
people = {1: {'name': 'John', 'age': '27', 'sex': 'Male'},
          2: {'name': 'Marie', 'age': '22', 'sex': 'Female'}}
print(people[1]) #{'name': 'John', 'age': '27', 'sex': 'Male'}
print(people[1]['name'])  #John
```


ppattanayak@silicon.ac.in

Iterating Through a Nested Dictionary

Example:

```
people = {1: {'name': 'John', 'age': '27', 'sex': 'Male'},
                                    2: {'name': 'Marie', 'age': '22', 'sex':
'Female'}}
for id in people:
    for info in people[id]:
        print(info," : ",people[id][info])
```

output:

```
name  :  John
age  :  27
sex  :  Male
name  :  Marie
age  :  22
sex  :  Female
```

Method in dictionary:

Methods	Description
copy()	This method returns a shallow copy of the dictionary. **Example:** stud={ 1: 66, 2: 55, 3: 88, 4: 60,5:99,11:98} CSE=stud.copy() print(CSE) #{1: 66, 2: 55, 3: 88, 4: 60, 5: 99, 11: 98}
clear()	Removes all items from the dictionary. stud.clear() print(stud) # { }
pop()	Removes and returns an element from a dictionary having the given key. raised KeyError, if not found. **Example**: stud={ 1: 66, 2: 55, 3: 88, 4: 60,11:98} print(stud.pop(4))**Output**: 60
get()	Return the value for key if key is in the dictionary, else default. **Example:** print(stud.get(2)) **# 55** print(stud.get(8)) **#none**

Cont...

Methods	Description
keys()	Returns list of dictionary dict's keys. **Example:** l=stud.keys() print(l) #dict_keys([1, 2, 3, 4, 5, 11])
values()	Returns list of dictionary dict's keys . **Example:** l=stud.values() print(l) #dict_values([66, 55, 88, 60, 99, 98])
fromkeys()	fromkeys(iterable, value=None, /) : Create a new dictionary with keys from iterable and values set to value. **x=d.fromkeys([1,2,3],0)** **print(x) #output: {1: 0, 2: 0, 3: 0}**
popitem()	D.popitem() -> (k, v), remove and return some (key, value) pair as a 2-tuple; but raise KeyError if D is empty. d={1:"hello",2:"Hi"} d.popitem() #output: (2, 'Hi')

Cont...

Methods	Description
update()	The update() method adds element(s) to the dictionary if the key is not in the dictionary. If the key is in the dictionary, it updates the key with the new value. **Example:** d = {1: "one", 2: "three"} d1 = {2: "two",3:"three"} d.update(d1) print(d) **#output** {1: 'one', 2: 'two', 3: 'three'}
setdefault()	**setdefault(key, default=None, /)** : Insert key with a value of default if key is not in the dictionary. Return the value for key if key is in the dictionary, else default. Example: d.setdefault(5,"hi") print(d) **# output:** {1: 'one', 2: 'two', 3: 'three', **5: 'hi'**}

ppattanayak@silicon.ac.in

Acess the key from a vlue in dictionary

Example:We can fetch key from a value by matching all the values and then print the corresponding key to given value.

```
my_dict ={"java":100, "python":112, "c":11}
val=100
for key, value in my_dict.items():
    if val == value:
        print(key)
        break

print("java".values())
```

output: java

#Program to count the number of times a character appears in a given string using a dictionary.

```
myString="I am a student of silicon"
dic={}
for i in myString:
    if i not in dic:
        dic[i]=1
    else:
        dic[i]=dic[i]+1
print(dic)
```

output:

```
{'I': 1, ' ': 5, 'a': 2, 'm': 1, 's': 2, 't': 2, 'u': 1, 'd': 1, 'e': 1, 'n': 2, 'o': 2, 'f': 1, 'i': 2, 'l': 1, 'c': 1}
```

Program to count and display frequency of a word appears in a given string using a dictionary.

```
myString="This is nice experience to learn python prog with association with Silicon. This is one of my best experience"
myList=myString.split()
dic={}
for i in myList:
    if i not in dic:
        dic[i]=1
    else:
        dic[i]=dic[i]+1
print(dic)
```

String

String is a sequence of characters enclosed with single cotes or double cotes.

Example:

S=”Silicon Institute of Technology”

Or

S1=’Silicon Institute of Technology’

Display String:

print(S) #print the entire string

Accessing String

Example: String elements are acess through index, it can be +ve or -ve:

0	1	2	3	4	5	6	7
P	**R**	**A**	**D**	**I**	**P**	**T**	**A**
-8	-7	-6	-5	-4	-3	-2	-1

Example: String Slicing

```
string ="PRADIPTA"
print(string[0], "  ", string[ 0:5])    # P    PRADI
print(string[-1]," ",string[-5:-2])    #A    DIP
print(string[3:-2])      #DIP
```

String operation

Let S1="Hello" and s2="Python"

Operator	Description	Example
+	Concatenation - Adds values on either side of the operator	s1 + s2 will give HelloPython
*	Repetition - Creates new strings, concatenating multiple copies of the same string	s1*2 will give - HelloHello
in	Membership - Returns true if a character exists in the given string	H in s1 will give 1
not in	Membership - Returns true if a character does not exist in the given string	M not in s1 will give 1

Built-in String Function

capitalize() : Capitalizes first letter of string

```
s="hello"
s=s.capitalize()
print(s)
```

Output: Hello

Find a string:

find(str, beg=0 end=len(string))

Determine if str occurs in string or in a substring of string if starting index beg and ending index end are given returns index if found and -1 otherwise.

Search substring ion a string

find(str, beg=0 end=len(string)):

It will search the substring in a string, within the range and return index if found and -1 otherwise.

Example:

Example:

```
s="Hello"
x=s.find("ol")
print(x)     #output: -1
y=s.find("lo")
print(y)   #output: 3
```

Cont...

index(str, beg=0, end=len(string))

Same as find(), but raises an exception if str not found.

Example-1:

```
s="Hello how are you"
v=s.index("are",0,len(s))
print(v)      #output 10
v=s.index("are",07)   #Output: Exception: ValueError
```

Check for Alphanumeric

Silicon ...beyond teaching

isalnum()

Return True if the string is an alpha-numeric string, False otherwise.

Example:

```
x="123"
print(x.isalnum())    #Output: True
y=”abc23”
print(x.isalnum())  #Output: True
z=”123@”
print(x.isalnum())  #Output: False
```

Cont...

isalpha()

Returns True if string has at least 1 character and all characters are alphabetic and false otherwise.

isdigit()

Returns True if string contains only digits and False otherwise.

islower()

Returns True if string has at least 1 cased character and all cased characters are in lowercase otherwise False

isupper()

Returns True if string has at least one cased character and all characters are in uppercase otherwise False

Change of case

lower():Converts all uppercase letters in string to lowercase.

Example:

```
s="Pradipta Kumar Pattanayak"
print(s.lower())    #output: pradiptakumarpattanayak
```

upper() **:** Converts lowercase letters in string to uppercase.

Example:

```
s="hello"
print(s.upper())
output: HELLO
```

split

split(str="", num=string.count(str))

Splits string according to delimiter str (space if not provided) and returns list of substrings; split into at most num substrings if given.

Example:

```
s="Pradipta Kumar Pattanayak"
v=s.split(" ")  #split by space
print(v)
```

output: ['Pradipta', 'Kumar', 'Pattanayak']

strip

strip([chars])

Performs both lstrip() and rstrip() on string.

Example:

```
s="  hello  "
print(s.strip())  # removes the white space from
                                        beginning and end
```

output: hello

ppattanayak@silicon.ac.in

join

join(seq)

Merges (concatenates) the string representations of elements in sequence seq into a string, with separator string.

Example:

```
# Joining with empty separator
list1 = ["Slicon","institute","of","Technology"]
print("".join(list1))
```

output: SliconinstituteofTechnology

Program to display the word of a string in revrese order

```
myString="Silicon Institute of Technology"
temp=myString.split()
print(temp)
resString=" "
for i in range(len(temp)-1,-1,-1):
    resString +=" "+ temp[i]
print(resString)
```

output: Technology of Institute Silicon

Problem

Alice and Bob each created one problem . A reviewer rates the two challenges, awarding points on a scale from 1 to 100 for three categories: problem clarity, originality, and difficulty.

The rating for Alice's challenge is the triplet a = (a[0], a[1], a[2]), and the rating for Bob's challenge is the triplet b = (b[0], b[1], b[2]).

The task is to find their comparison points by comparing a[0] with b[0], a[1] with b[1], and a[2] with b[2].

If a[i] > b[i], then Alice is awarded 1 point.

If a[i] , then Bob is awarded 1 point.

If a[i] = b[i], then neither person receives a point.

Comparison points is the total points a person earned.

Given a and b, determine their respective comparison points.

Cont...

Example

a = [1, 2, 3]

b = [3, 2, 1]

For elements 0, Bob is awarded a point .

For the equal elements a[1] and b[1], no points are earned.

Finally, for elements 2, a[2] > b[2] so Alice receives a point.

The return array is [1, 1] with Alice's score first and Bob's second.

Cont...

Return

int[2]: Alice's score is in the first position, and Bob's score is in the second.

Input Format

The first line contains 3 space-separated integers, a[0], a[1], and a[2], the respective values in triplet a.

The second line contains 3 space-separated integers, b[0], b[1], and b[2], the respective values in triplet b.

Constraints

$1 \le a[i] \le 100$

$1 \le b[i] \le 100$

Sample Input 0

5 6 7

3 6 10

Sample Output 0

1 1