GATEWAY CLASSES

PYTHON PROGRAMMING

UNIT 3
COMPLEX DATA TYPE

Today's Target

- List and Tuples
- AKTU PYQs

By PRAGYA RAJVANSHI

B.Tech, M.Tech(C.S.E.)

Gateway Classes

LIST AKTU (2022-23)

- The list is a **sequence data type** which is used to store the **collection of data**
- Lists in Python can be created by just placing the sequence inside the **square brackets[ ]**. .
- list doesn't need a **built-in function** for its creation of a list. .
- the list may contain **mutable element**
- list may contain **duplicate values**

with **their distinct positions**

```
l1=[]
print(l1)
l2=[1,2,3,4]
print(l2)
l3=[1,2.3,"hello"]
print(l3)
l4=[1,2,3,3,3,4,4,5,5]
print(l4)
l5=[1,4,5]
l5[2]=99
print(l5)
```

```
[]
[1, 2, 3, 4]
[1, 2.3, 'hello']
[1, 2, 3, 3, 3, 4, 4, 5, 5]
[1, 4, 99]
```

Gateway Classes

LIST METHODS (AKTU 2022-23)

1 append()

- **Used for adding elements to the end of the List.**

```
1  l=[2,5,6,7]
2  l.append(8)
3  print(l)
```

```
[2, 5, 6, 7, 8]
```

2 max()

Calculates the maximum of all the elements of the List

```
1  l = [1.2, 1.3, 0.1]
2  max_value = max(l)
3  print(max_value)
4
```

```
1.3
```

3 min()

Calculates the minimum of all the elements of the List

```
1  numbers = [3, 2, 8, 5, 10, 6]
2  small = min(numbers);
3  print("The smallest number is:", small)
4
```

```
The smallest number is: 2
```

Gateway Classes

4 reverse() Reverses objects of the List in place

```
list1 = [1, 2, 3, 4, 1, 2, 6]
list1.reverse()
print(list1)
```

```
[6, 2, 1, 4, 3, 2, 1]
```

5 remove() Removes a given object from the List.

```
lis = ['a', 'b', 'c']
lis.remove("b")
print(lis)
```

```
['a', 'c']
```

6.Sort() Sort a List in ascending, descending, or user defined order

```
numbers = [1, 3, 4, 2]
numbers.sort(reverse=True)
print(numbers)
```

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

```
numbers = [1, 3, 4, 2]
numbers.sort()
print(numbers)
```

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

Gateway Classes

7 pop() Removes and returns the last value from the List or the given index value.

```
my_list = [1, 2, 3, 6,9, 8,12,5]
print("Popped element:", my_list.pop(5))
print("List after pop():", my_list)

```

```
Popped element: 8
List after pop(): [1, 2, 3, 6, 9, 12, 5]
```

```
my_list = [1, 2, 3, 6]
print("Popped element:", my_list.pop())
print("List after pop():", my_list)

```

```
Popped element: 6
List after pop(): [1, 2, 3]
```

8insert() method inserts an item at a specific index in a list.

```
fruit = ["banana","cherry","grape"]
fruit.insert(1,"apple")
print(fruit)
```

```
['banana', 'apple', 'cherry', 'grape']
```

9 index() method searches for a given element from the start of the list and returns the position of the first occurrence.

```
Animals= ["cat", "dog", "tiger","dog"]
print(Animals.index("dog"))

```

```
1
```

GW GATEWAY CLASSES
Gateway Classes

10 clear() removes all items from the List making it an empty/null list.

```
lis = [1, 2, 3]
lis.clear()
print(lis)
```

```
[]
```

11.Count() returns the count of the occurrences of a given element in a list.

main.py

```
fruits=["apple","cherry","apple"]
print(fruits.count("apple"))
```

```
2
```

12 extend()

It is method adds items of an iterable (list, tuple, dictionary, etc) at the end of a list.

```
cars = ["Audi","BMW","Jaguar"]
other_cars = ["Maruti", "Tata"]
cars.extend(other_cars)
print(cars)
```

```
['Audi', 'BMW', 'Jaguar', 'Maruti', 'Tata']
```

13 Copy()

A shallow copy creates a new object, but it does not create new copies of nested objects (like lists inside a list). Instead, it copies references to those nested objects. So, changes to nested objects affect both the original and the copied object.

GATEWAY CLASSES
Gateway Classes

```python
import copy
original = [1, [2, 3], 4]
shallow_copy = copy.copy(original)
shallow_copy[1][0] = 99
print(original)
print(shallow_copy)
```

```
[1, [99, 3], 4]
[1, [99, 3], 4]
```

```python
import copy
original = [1,3,4]
shallow_copy = copy.copy(original)
shallow_copy[1] = 99
print(original)
print(shallow_copy)

```

```
[1, 3, 4]
[1, 99, 4]
```

Gateway Classes

GATEWAY CLASSES

```python
import copy
original = [1, [2, 3], 4]
deep_copy = copy.deepcopy(original)
deep_copy[1][0] = 42
print(original)
print(deep_copy)
```

```
[1, [2, 3], 4]
[1, [42, 3], 4]
```

Gateway Classes

LIST ARE MUTUBLE (AKTU 2023-24 ODD)

- Yes, lists in Python are mutable, which means that their elements can be changed after the list has been created. This includes adding, removing, or modifying elements.

```
[1, 2, 99, 4, 5]
[1, 2, 3, 4]
[1, 99, 2, 3, 4]
[1, 99, 2, 3, 4, 5, 6]
```

```
#MODIFYING ELEMENT
L = [1, 2, 3, 4, 5]
L[2] = 99
print(L)
#ADDING ELEMENT
L1 = [1, 2, 3]
L1.append(4)
print(L1)
L1.insert(1, 99)
print(L1)
L1.extend([5, 6])
print(L1)
```

```
l3=[1,2,3,7,4,5]
del l3[0]
print (l3)
```

```
[2, 3, 7, 4, 5]
```

```
# deleting element
l2=[1,2,3,4,5]
l2.remove(3)
print(l2)
popped_element=l2.pop(1)
print(l2)
```

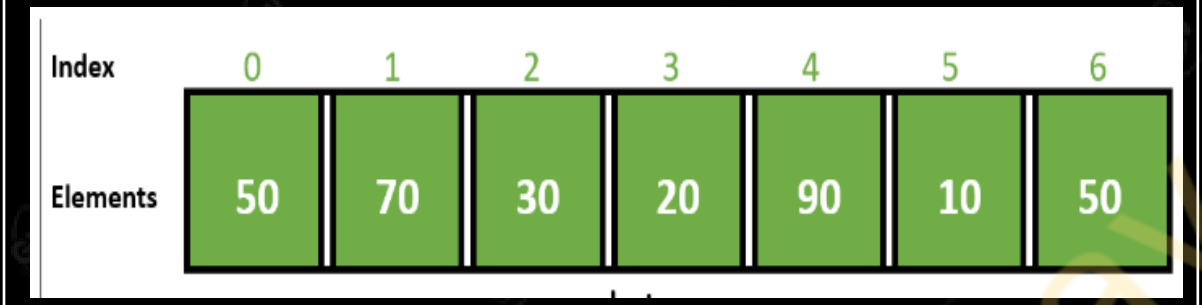
```
[1, 2, 4, 5]
[1, 4, 5]
```

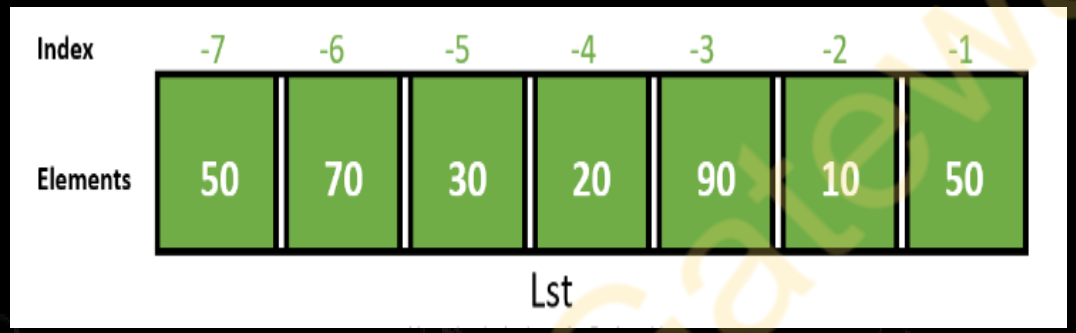
Gateway Classes

LIST SLICING

In order to access a range of elements in a list, you need to slice a list. One way to do this is to use the simple slicing operator i.e. colon(:).

Lst[Initial : End : IndexJump]





TO PRINT THE COMPLETE LIST

```
1  Lst = [50, 70, 30, 20, 90, 10, 50]
2  print(Lst[::])
3
```

```
[50, 70, 30, 20, 90, 10, 50]
```

```
1  Lst = [50, 70, 30, 20, 90, 10, 50]
2  print(Lst[-7::1])
```

```
[50, 70, 30, 20, 90, 10, 50]
```

PRINT ONLY 2,3,4 ,5 index element

```
1  Lst = [50, 70, 30, 20, 90, 10, 50]
2  print(Lst[1:5])
```

```
[70, 30, 20, 90]
```

Gateway Classes

1 Print from 4 element of the list then take jump of 2 upto the end of the list.

```
List = [1, 2, 3, 4, 5, 6, 7, 8, 9]
print(List[3:9:2])
print(List[::2])
```

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

Write a function to find ASCII value of the character.

```
def ascii_value(a):
    print(ord(a))
ascii_value("A")
```

```
65
```

Write a program to reverse a list

```
my_list = [1, 2, 3, 4, 5]
reversed_list = my_list[::-1]
print(reversed_list)
```

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

Gateway Classes

List comprehension (AKTU 2022-23)

It is used to create a new list from the existing list

Syntax: newList = [expression(element) **for** element **in** oldList **if** condition]

```
numbers = [12, 13, 14,]
doubled = [x *2 for x in numbers]
print(doubled)
```

```
[24, 26, 28]
```

```
list = [i+3 for i in range(11) if i % 2 == 0]
print(list)
```

```
[3, 5, 7, 9, 11, 13]
```

Question:- Write a function lessthan (lst,k) to return list of number less than k from the list lst. The function mnust use list comprehension

Lessthan([1,-2,0,5,-3],0) returns[-2.-3] (AKTU 2020-21)

```
def lessthan(lst,k):
    return[i for i in lst if i<k]
print(lessthan([1,-2,0,5,-3],0))
```

```
[-2, -3]
```

Gateway Classes

Illustrate different list slicing constructs for the following operations on the following list:L= [1, 2, 3, 4, 5, 6, 7, 8, 9]

(AKTU 2023-24 ODD)

1. Return a list of numbers starting from the last to second item of the list
2. Return a list that start from 3rd item to second last item.
3. Return a list that has only even position elements of list L.
4. Return a list that starts from the middle of the list L.
5. Return a list that reverses all the elements starting from element at index 0 to middle index only and return the entire list.
6. Divide each element of the list by 2 and replace it with the remainder.

```
l = [1, 2, 3, 4, 5, 6, 7, 8, 9]
#1 solution
result1=l[::-1][:-1]
print(result1)
#2 solution
result2=l[2:-1]
print(result2)
#3 solution
result3=l[0::2]
print(result3)
#4 solution
middle_index=len(l)//2
result4=l[middle_index:]
print(result4)
```

GW GATEWAY CLASSES
Gateway Classes

lustrate different list slicing constructs for the following operations on the following list:L= [1, 2, 3, 4, 5, 6, 7, 8, 9]

1. Return a list of numbers starting from the last to second item of the list
2. Return a list that start from 3rd item to second last item.
3. Return a list that has only even position elements of list L to list M.
4. Return a list that starts from the middle of the list L.
5. Return a list that reverses all the elements starting from element at index 0 to middle index only and return the entire list.
6. Divide each element of the list by 2 and replace it with the remainder.

```
l = [1, 2, 3, 4, 5, 6, 7, 8, 9]
#5 solution
middle_index=len(l)//2
result5=l[:middle_index+1][::-1]+l[middle_index+1:]
print(result5)
#6 solultion
result6=[x%2 for x in l]
print(result6)
```

```
[5, 4, 3, 2, 1, 6, 7, 8, 9]
[1, 0, 1, 0, 1, 0, 1, 0, 1]
```

Gateway Classes

Write python function, alternating(lst) that takes as a argument a sequence list. The function return true if the elements in list are alternatively odd and even starting with even number otherwise it return false (AKTU 2020-21)

```
def alternating(lst):
    if len(lst)==0:
        return True
    elif len(lst)==1:
        if lst[0]%2==0:
            return True
        return False
    elif lst[0]%2==0:
        for i in range(len(lst)-1):
            if lst[i]%2==lst[i+1]%2:
                return False
            return True
    else:
        return False
print(alternating([]))
print(alternating([10]))
print(alternating([7]))
print(alternating([10,7,8]))
print(alternating([10,8,8]))
print(alternating([7,12,3]))
```

```
True
True
False
True
False
False
```

GW GATEWAY CLASSES
GW GATEWAY CLASSES
Gateway Classes

Write a function make Pairs that takes as input two lists of equal length and return a single list of same length where kth element is the pair of k element from the input list

For example

Make Pairs([1,,3,4,5],[2,3,4,9])

Return[(1,2),(3,3),(4,4),(5,9)]

Make Pairs([],[])

Return[] (AKTU 2020-21)

```
1  def makePairs(list1,list2):
2      x=len(list1)
3      y=len(list2)
4      if x==y:
5          pairlist=[(list1[i],list2[i]) for i  in range(0,x)]
6      return pairlist
7
8  print(makePairs([1,3,5,7],[2,4,6,8]))
9
10
```

input

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

GATEWAY CLASSES
Gateway Classes

Consider following filter (AKTU 2023-24 ODD)

1 filter all the numbers

```
text=["apple","123","234","bye"]
z=[word for word in text if word.isdigit()]
print(z)
```

```
['123', '234']
```

2 filter all the string starting with vowels

```
text=["apple","123","234","bye","ego","hi"]
vowels=["a","e","i","o","u","A","E","I","O","U"]
z=[ word for word in text if word[0] in vowels]
print(z)
```

```
['apple', 'ego']
```

GATEWAY CLASSES
Gateway Classes

filters all the strings that contains any of the following noun: agra ,Ramesh , tomato, Patna

```
# List of nouns to filter
nouns = {"agra", "Ramesh", "tomato", "Patna"}

# Sample list of strings
strings = ["I visited agra last year", "Ramesh is a good boy", "I love eating tomato",
"This is a random sentence"]

# Filtering strings that contain any of the nouns
filtered_strings = [s for s in strings if  any(noun in s for noun in nouns)]

print("Filtered list:", filtered_strings)
```

```
Filtered list: ['I visited agra last year', 'Ramesh is a good boy', 'I love eating tomato']
```

GATEWAY CLASSES
Gateway Classes

TUPLES(AKTU 2022-23)

- Tuples are similar as list but only difference is that they are immutable

Creating Python Tuples

- Using round brackets

```
var = ("hello", "for", "bye")
print(var)
print(type(var))
```

```
('hello', 'for', 'bye')
<class 'tuple'>
```

- With one item

```
mytuple = ("hi",)
print(type(mytuple))
#NOT a tuple
mytuple = ("hi")
print(type(mytuple))
```

```
<class 'tuple'>
<class 'str'>
```

- Tuple Constructor

```
tuple_constructor = tuple(("dsa", "developement", "deep learning"))
print(tuple_constructor)
```

Gateway Classes

What is Immutable in Tuples?

- Tuples in Python are similar to Python list but not entirely. Tuples are immutable and ordered and allow duplicate values. Some Characteristics of Tuples in Python.
- We can find items in a tuple since finding any item does not make changes in the tuple.
- One cannot add items to a tuple once it is created.
- Tuples cannot be appended or extended.
- We cannot remove items from a tuple once it is created.

```
1  mytuple = (1, 2, 3, 4, 5)
2
3  # tuples are indexed
4  print(mytuple[1])
5  print(mytuple[4])
6
7  # tuples contain duplicate elements
8  mytuple = (1, 2, 3, 4, 2, 3)
9  print(mytuple)
10
11 # adding an element
12 mytuple[1] = 100
13 print(mytuple)
14
```

input

```
2
5
(1, 2, 3, 4, 2, 3)
Traceback (most recent call last):
  File "/home/main.py", line 12, in <module>
    mytuple[1] = 100
TypeError: 'tuple' object does not support item assignment
```

Gateway Classes

Unpacking a Tuple (AKTU 2023-24 ODD)

In Python, there is a very powerful tuple assignment feature that assigns the right-hand side of values into the left-hand side. In another way, it is called unpacking of a tuple of values into a variable. In packing, we put values into a new tuple while in unpacking we extract those values into a single variable.

Tuple unpacking in Python allows you to assign elements of a tuple to multiple variables in a single statement. This is useful when you have a tuple and want to extract its values directly into separate variables

```python
# this lines PACKS values
# into variable a
a = ("MNNIT Allahabad", 5000, "Engineering")

# this lines UNPACKS values
# of variable a
(college, student, type_ofcollege) = a

# print college name
print(college)

# print no of student
print(student)

# print type of college
print(type_ofcollege)
```

```
MNNIT Allahabad
5000
Engineering
```

GATEWAY CLASSES
Gateway Classes

Operation on tuples

```
1  # 1concatenation
2  tuple1 = (0, 1, 2, 3)
3  tuple2 = ('python', 'unit2')
4  print(tuple1 + tuple2)
5  #nesting
6  tuple1 = (0, 1, 2, 3)
7  tuple2 = ('python', 'hi')
8  tuple3 = (tuple1, tuple2)
9  print(tuple3)
10 #3 repetition
11 tuple3 = ('python',)*3
12 print(tuple3)
13 # deletion is not allowed
14
```

```
(0, 1, 2, 3, 'python', 'unit2')
((0, 1, 2, 3), ('python', 'hi'))
('python', 'python', 'python')
```

main.py

```
1  # slicing
2  tuple1 = (0 ,1, 2, 3)
3  print(tuple1[1:])
4  print(tuple1[::-1])
5  print(tuple1[2:4])
6
7
8
```

```
1, 2, 3)
3, 2, 1, 0)
2, 3)
```

Gateway Classes

Accessing Values in Python Tuples

- **Positive index**

```
var = ("hi", "for", "bye")
print("Value in Var[0] = ", var[0])
print("Value in Var[1] = ", var[1])
print("Value in Var[2] = ", var[2])
```

- **Negative index**

```
var = (1, 2, 3)
print("Value in Var[-1] = ", var[-1])
print("Value in Var[-2] = ", var[-2])
```

Gateway Classes

Write thon program to find repeated items in a tuple:

```
t = (1, 2, 3, 2, 4, 5, 6, 1, 7, 8, 2)
repeated_items = {item for item in t if t.count(item) > 1}
print("Repeated items:", repeated_items)
```

```
Repeated items: {1, 2}
```

Write a program to convert list to tuple

```
1  L = [1, 2, 3, 4, 5]
2  T = tuple(L)
3  print("Tuple:", T)
4  print(type(T))
```

```
Tuple: (1, 2, 3, 4, 5)
<class 'tuple'>
```

remove any item from the tuple

```
t=(1,24,33,44,95)
z=list(t)
z.remove(33)
t2=tuple(z)
print(t2)
```

```
(1, 24, 44, 95)
```

Gateway Classes

Write a program to reverse the element of the tuple

```
t = (1, 2, 3, 4, 5)
reversed_t = t[::-1]
print("Reversed tuple:", reversed_t)
```

```
Tuple:  (1,  2,  3,  4,  5)
<class 'tuple'>
```

Write a program to multiply all the element in a tuple.

```
t = (1, 2, 3, 4, 5)
product = 1
for num in t:
    product *= num
print("Product of all numbers in the tuple:", product)
```

```
Product of all numbers in the tuple: 120
```

write a program remove duplicate elements from a list and print the updated list (AKTU 2023-24)

```
L = [1, 2, 2, 3, 4, 4, 5]
L = list(set(L))
print("List after removing duplicates:", L)
```

Write a program to add item in tuple

```
t=(1,2,3,4)
add=0
for num in t:
    add=add+num
print(add)
```

```
10
```

Gateway Classes

Thank You

GatewayClasses