GATEWAY CLASSES

PYTHON PROGRAMMING

UNIT 3
COMPLEX DATA TYPE

Today's Target

- String
- AKTU PYQs

By PRAGYA RAJVANSHI

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

Gateway Classes

String

the Updating or deletion of characters from a String is not allowed.

String are immutable

```
1  String1 = "gatewayclasses"
2  print("Initial String: ", String1)
3  #string slicing
4  print( String1[-3])# negative index
5  print(String1[3])#positive index
6  print(String1[1:3])
7  String1[0]="k"
8
```

```
Initial String:  gatewayclasses
s
e
at
Traceback (most recent call last):
  File "/home/main.py", line 7, in <module>
    String1[0]="k"
TypeError: 'str' object does not support item assignment
```

Len() It can be used to find the length of an object

```
1  list1 = ['hi','for','3444',2022]
2  print(len(list1))
3  tuple1 = (1,2,3,4)
4  print(len(tuple1))
5  string1 = "python"
6  print(len(string1))
7
8
```

```
4
4
6
```

Concatenation

.

```
1  str1="hello"
2  str2="students"
3  print(str1+str2)
4
```

```
hellostudents
```

GATEWAY CLASSES
Gateway Classes

Write a Python program to get a string made of the first 2 and last 2 characters of a given string. If the string length is less than 2, return the empty string instead.

```
1 def string_transform(s):
2     if len(s) < 2:
3         return ""
4     return s[:2] + s[-2:]
5 print(string_transform("python"))
6 print(string_transform("w3"))
7 print(string_transform("w"))
```

```
pyon
w3w3
```

GATEWAY CLASSES
Gateway Classes

GATEWAY CLASSES

write a progarm to take two string and fid out the common letter between them

GATEWAY CLASSES

```
def common_letters(s1, s2):
    return "".join(sorted(set(s1) & set(s2)))
print(common_letters("hello", "world"))
print(common_letters("python", "java"))
print(common_letters("abc", "xyz"))
```

Gateway Classes

Write a Python function to get a string made of 4 copies of the last two characters of a specified string (length must be at least 2)

```
def repeat_last_two(s):
    if len(s) < 2:
        return "String must be at least 2 characters long."
    return s[-2:] * 4
print(repeat_last_two("hello"))
print(repeat_last_two("ab"))
print(repeat_last_two("x"))
```

```
lolololo
abababab
String must be at least 2 characters long.
```

Write a program to Reverse string if length is a multiple of 4

```
s = input("Enter a string: ")
if len(s) % 4 == 0:
    s = s[::-1]
print("Output:", s)
```

```
Enter a string: hell
Output: lleh
```

GATEWAY CLASSES
Gateway Classes

Write a Python program to change a given string to a newly string where the first and last chars have been exchange

```
s = input("Enter a string: ")
if len(s) < 2:
    print("Output:", s)
else:
    new_s = s[-1] + s[1:-1] + s[0]
    print("Output:", new_s)
```

```
Output: oellh
```

Write a Python program to remove characters that have odd index values in a given string.

```
s = input("Enter a string: ")
new_s = s[::2]
print("Output:", new_s)
```

```
Enter a string: python
Output: pto
```

GW GATEWAY CLASSES
Gateway Classes

Write a program that accepts a sentence and calculate the number of digit, uppercase and lowercase (AKTU 2021-22)

```
def string_test(s):#s="the Q 1"
    d={"uppercase":0,"lowercase":0,"digit":0}
    for c in s:
        if c.isupper():
            d["uppercase"]=d["uppercase"]+1
        elif c.islower():
            d["lowercase"]=d["lowercase"]+1
        elif c.isdigit():
            d["digit"]=d["digit"]+1
        else:
            pass
    print("number of upper case characters",d["uppercase"])
    print("number of lower case characters",d["lowercase"])
    print("number of digit",d["digit"])

string_test("the Q 1")
```

```
number of upper case characters 1
number of lower case characters 3
number of digit 1
```

Write a program to change a given string to a new string where the first and last chars have been changed(AKTU 2021-22)

main.py

```
def change_string(str1):
    return str1[-1:] + str1[1:-1] + str1[:1]
print(change_string('abcd'))
```

```
dbca
```

GW GATEWAY CLASSES
Gateway Classes

Write a program to count the vowels present in given input . (AKTU 2022-23)

```python
def vowel_count(str):
    count = 0
    vowel = set("aeiouAEIOU")
    for alphabet in str:
        if alphabet in vowel:
            count = count + 1
    print("No. of vowels :", count)

str = "pythonprogramming"
vowel_count(str)
```

```
No. of vowels : 4
```

Write a program to check whether a string is a palindrome or not (AKTU 2022-23)

```python
def ispalindrome(s):
    return s==s[::-1]
k=input("enter the string\n")
ans=ispalindrome(k)
if ans:
    print("yes plaindrome")
else:
    print("no palindrome")
```

```
enter the string
malayalam
yes plaindrome
```

Gateway Classes

How can you randomize the items of a list in place in python ? (AKTU 2021-22)

```
1 import random
2 test_list = [1, 4, 5, 6, 3]
3 print("The original list is : " + str(test_list))
4 random.shuffle(test_list)
5 print("The shuffled list is : " + str(test_list))
6
```

input

```
The original list is : [1, 4, 5, 6, 3]
The shuffled list is : [6, 4, 1, 5, 3]
```

What will be the output

```
1 def cube(x):
2     return x*x*x
3 x=cube(3)
4 print(x)
```

```
27
```

GATEWAY CLASSES
Gateway Classes

Write a program that accepts sequence of lines as input and print the lines after making all character in the sentence is capitalized(AKTU 2022-23)

```
lines = []
print("Enter multiple lines (press Enter on an empty line to stop):")
while True:
    line = input()
    if line == "":
        break
    lines.append(line.upper())

print("\nCapitalized Output:")
for line in lines:
    print(line)
```

```
Enter multiple lines (press Enter on an empty line to stop):
hello
bye


Capitalized Output:
HELLO
BYE
```

GATEWAY CLASSES
Gateway Classes

Consider a function perfect_square(number) that returns a number if it is perfect square otherwise return-1 (AKTU 2023-24 ODD)

```python
import math
def perfect_square(number):
    z= math.sqrt(number)
    if z*z==number:
        return number
    else:
        return-1
print(perfect_square(4))
print(perfect_square(2))
```

```
4
-1
```

Split()

```python
text= 'Split this string'

# splits using space
print(text.split())

grocery = 'Milk, Chicken, Bread'

# splits using ,
print(grocery.split(', '))

grocery1 = 'Milk:Chic:ken, Bread'
print(grocery1.split(':'))

```

```
['Split', 'this', 'string']
['Milk', 'Chicken', 'Bread']
['Milk', 'Chic', 'ken, Bread']
```

GATEWAY CLASSES
Gateway Classes

Construct a program that accepts a comma separated sequence of word as an input and prints the word in a comma- separated sequence after sorting them alphabetically(AKTU 2023-24 ODD)

```
1 items = input("Input comma-separated sequence of words\n")
2 words = [word for word in items.split(",")]
3 print(",".join(sorted(list(set(words)))))
4
```

input

```
Input comma-separated sequence of words
without,hello,bag,world
bag,hello,without,world
```

What will be the output. (AKTU 2022-23)

```
1 list1=['M','N','K','Y']
2 print("@".join(list1))
3
```

```
M@N@K@Y
```

Final result is string

GATEWAY CLASSES
Gateway Classes

Construct a function ret_smaller(l) that return smaller list from a nested list. if two list have same length then return the first list that is encountered (AKTU 2023-24 ODD)

```python
def ret_smaller(l):
    # If the list is empty, return an empty list
    if not l:
        return []

    # Start by assuming the first sublist is the smallest
    smallest_list = l[0]

    # Iterate over the rest of the sublists
    for sublist in l[1:]:
        # If a smaller sublist is found, update smallest_list
        if len(sublist) < len(smallest_list):
            smallest_list = sublist

    # Return the smallest sublist found
    return smallest_list

# Example usage
nested_list = [[3, 4, 5], [1, 2], [6, 7, 8, 9], [10]]
print(ret_smaller(nested_list))  # Output: [1, 2]

```

input

```
[10]
```

GATEWAY CLASSES
Gateway Classes

W rite a Python function to insert a string in the middle of a string.

```
1  def insert_middle(original, insert):
2      mid = len(original) // 2
3      return original[:mid] + insert + original[mid:]
4  print(insert_middle("abcdef", "XYZ"))
5  print(insert_middle("hello", "123"))
6
```

```
abcXYZdef
he123llo
```

GATEWAY CLASSES
Gateway Classes

sets

- A Set in Python is used to store a collection of items with the following properties.
- No duplicate elements. If try to insert the same item again, it overwrites previous one.
- An unordered collection. When we access all items, they are accessed without any specific order and we cannot access items using indexes as we do in lists.
- Mutable, meaning we can add or remove elements after their creation, the individual elements within the set cannot be changed directly

```
s = {10, 50, 20}
print(s)
print(type(s))
```

```
{10, 50, 20}
<class 'set'>
```

Note:
There is no specific order for set elements to be printed

```
my_set = {10, 5, 3, 8, 1,1,2,2}
print(my_set)
```

```
{1, 2, 3, 5, 8, 10}
```

GATEWAY CLASSES
Gateway Classes

1. Mutability of Sets (Adding & Removing Elements)

```python
my_set = {1, 2, 3}
print("Original Set:", my_set)
# Adding elements
my_set.add(4)
print("After Adding 4:", my_set)
#Removing elements
my_set.remove(2)
print("After Removing 2:", my_set)
# Discarding an element (won't raise an error if not found)
my_set.discard(5)
print("After Discarding 5 (not present):", my_set)
# Clearing all elements
my_set.clear()
print("After Clearing:", my_set)
```

```
Original Set: {1, 2, 3}
After Adding 4: {1, 2, 3, 4}
After Removing 2: {1, 3, 4}
After Discarding 5 (not present): {1, 3, 4}
After Clearing: set()
```

Gateway Classes

Key points:

add() inserts a new element

remove() deletes an element but raises an error if it's not found.

discard() deletes an element but does not raise an error if it's missing

clear() removes all elements.

2. Elements Inside a Set Must Be Immutable

Although we can modify the set itself, we cannot change the elements inside the set directly.

Attempting to Add a Mutable Element (List)

```
1  my_set = {1, 2, [3, 4]}
2
```

input

```
Traceback (most recent call last):
  File "/home/main.py", line 1, in <module>
    my_set = {1, 2, [3, 4]}  # Invalid, lists are mutable
             ^^^^^^^^^^^^^^
TypeError: unhashable type: 'list'
```

GW GATEWAY CLASSES
Gateway Classes

Using a Tuple Instead of a List

```
1  my_set = {1, 2, (3, 4)}
2  print(my_set)
```

```
{1, 2, (3, 4)}
```

- **Mutable elements** (like lists, dictionaries, sets) cannot be stored in a set
- .**Immutable elements** (like numbers, strings, tuples) are allowed.

Conclusion

- **Sets are mutable** → We can add or remove elements.
- **Set elements must be immutable** → We cannot modify elements directly.
- Lists and dictionaries cannot be elements of a set, but tuples and strings can.

Gateway Classes

Heterogeneous Element with Python Set

Python sets can store heterogeneous elements in it, i.e., a set can store a mixture of string, integer, boolean, etc datatypes.

```
1  s = {"hi", "for", 10, 52.7, True}
2  print(s)
```

```
{True, 'hi', 52.7, 10, 'for'}
```

UNION

The union of two sets combines all unique elements from both sets.

```
A = {1, 2, 3}
B = {3, 4, 5}
# Using `|` operator
print("Union:", A | B)
# Using `union()` method
print("Union (method):", A.union(B))
```

```
Union: {1, 2, 3, 4, 5}
Union (method): {1, 2, 3, 4, 5}
```

Gateway Classes

Intersection

```
A = {1, 2, 3}
B = {3, 4, 5}
# Using `&` operator
print("Intersection:", A & B)
# Using `intersection()` method
print("Intersection (method):", A.intersection(B))
```

```
Union: {1, 2, 3, 4, 5}
Union (method): {1, 2, 3, 4, 5}
```

Diffrence

```
A = {1, 2, 3}
B = {3, 4, 5}
# Using `-` operator
print("Difference (A - B):", A - B)
print("Difference (B - A):", B - A)
# Using `difference()` method
print("Difference (A - B) (method):", A.difference(B))
print("Difference (B - A) (method):", B.difference(A))
```

```
Difference (A - B): {1, 2}
Difference (B - A): {4, 5}
Difference (A - B) (method): {1, 2}
Difference (B - A) (method): {4, 5}
```

Gateway Classes

Clearing python set

```python
my_set = {1, 2, 3, 4, 5}
print("Before clearing:", my_set)
# Clearing the set
my_set.clear()
print("After clearing:", my_set)
```

```
Before clearing: {1, 2, 3, 4, 5}
After clearing: set()
```

GATEWAY CLASSES
Gateway Classes

Write a program to find permutation of a given string.

main.py

```
1  from itertools import permutations
2
3  def find_permutations(string):
4      perm_list = list(permutations(string))
5      perm_strings = [''.join(perm) for perm in perm_list]
6      return perm_strings
7  input_string = "abc"
8  permutations_list = find_permutations(input_string)
9  print(permutations_list)
```

input

```
['abc', 'acb', 'bac', 'bca', 'cab', 'cba']
```

What will be the output ? (AKTU 2022-23)

main.py

```
1  def count1(s):
2      vowels="AEIOUaeiou"
3      count=0
4      for c in s:
5          if c in vowels:
6              count=count+1
7      return count
8  print(count1("i love India"))
9
```

```
6
```

GATEWAY CLASSES
Gateway Classes

Question.1 Which of the following statement produce an error in python?

statement 1: x,y,z=1,2,3

statement 2: a,b=4,5,6

statement 3: u=7,8,9(AKTU 2020-21)

Statement 1 give no error Statement 2 gives error because 6 is not assigned to any variable

Statement 3 gives error as 8,9 is not declared to any variable

Q2 Explain why this program generate error

X=['12','hello',456]

X[0]*=3

X[1][1]='bye'

X[0]*=3 (AKTU 2020-21)

It does not give any error . X[0] refers to the string '12'. Multiplying a string by an integer in Python repeats the string that many times. This updates X[0] to '121212', and now the list X becomes:

X = ['121212', 'hello', 456]

X[1][1]='bye'

Here, X[1] refers to the string 'hello', and X[1][1] refers to the character at index 1 of 'hello', which is 'e'. The intention seems to be to replace the character 'e' with the string 'bye'. However, this line generates an error because strings in Python are immutable, meaning their individual characters cannot be changed.

GATEWAY CLASSES
Gateway Classes

Thank You

GatewayClasses