GATEWAY CLASSES

PYTHON PROGRAMMING

UNIT 3
COMPLEX DATA TYPE

Today's Target

- Dictionary , Function
- AKTU PYQs

By PRAGYA RAJVANSHI

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

Gateway classes

Python Dictionary Method (AKTU 2022-23)

Functions Name	Descriptions
clear()	Removes all items from the dictionary
copy()	Returns a shallow copy of the dictionary
fromkeys()	Creates a dictionary from the given sequence
get()	Returns the value for the given key

Gateway classes

Functions Name	Descriptions
items()	Return the list with all dictionary keys with values
keys()	Returns a view object that displays a list of all the keys in the dictionary in order of insertion
pop()	Returns and removes the element with the given key
popitem()	Returns and removes the key-value pair from the dictionary
setdefault()	Returns the value of a key if the key is in the dictionary else inserts the key with a value to the dictionary
values()	Updates the dictionary with the elements from another dictionary
update()	Returns a list of all the values available in a given dictionary

Gateway classes

```
text = {1: "g", 2: "for"}
text.clear()
print('text =', text)
```

```
text = {}
```

main.py

```
original = {1: 'g', 2: 'for'}
new = original.copy(
new.clear()
print('new: ', new)
print('original: ', original)
```

```
new:  {}
original:  {1: 'g', 2: 'for'}
```

```
seq = {'a', 'b', 'c', 'd', 'e'}

# creating dict with default values as None
res_dict = dict.fromkeys(seq)

print("The newly created dict with None values : " + str(res_dict))

# creating dict with default values as 1
res_dict2 = dict.fromkeys(seq, 1)

print("The newly created dict with 1 as value : " + str(res_dict2))
```

input

```
The newly created dict with None values : {'c': None, 'b': None, 'd': None, 'a': None, 'e': None}
The newly created dict with 1 as value : {'c': 1, 'b': 1, 'd': 1, 'a': 1, 'e': 1}
```

Gateway classes

```python
d = {'coding': 'good', 'thinking': 'better'}
print(d.get('coding'))

```

```
good
```

```python
Dictionary1 = {'A': 'python', 'B': 'For', 'C': 'python'}
print(Dictionary1.keys())
```

input

```
dict_keys(['A', 'B', 'C'])
```

```python
student = {"rahul":7, "Aditya":1, "Shubham":4}
print(student)
suspended = student.pop("rahul")
print("suspended student roll no. = ",suspended)
print("remaining student", student)

```

```
{'rahul': 7, 'Aditya': 1, 'Shubham': 4}
suspended student roll no. =  7
remaining student {'Aditya': 1, 'Shubham': 4}
```

Gateway classes

```python
d = {1: '001', 2: '010', 3: '011'}
print(d.popitem())
```

```
(3, '011')
```

```python
d = {'a': 97, 'b': 98}
print("setdefault() returned:", d.setdefault('b', 99))
print("After using setdefault():", d)
```

```
setdefault() returned: 98
After using setdefault(): {'a': 97, 'b': 98}
```

```python
d = {'a': 97, 'b': 98, 'c': 99, 'd': 100}
# space key added using setdefault() method
d.setdefault(' ', 32)
print(d)
```

```
{'a': 97, 'b': 98, 'c': 99, 'd': 100, ' ': 32}
```

```python
dictionary = {"raj": 2, "striver": 3, "vikram": 4}
print(dictionary.values())
```

```
dict_values([2, 3, 4])
```

Gateway classes

main.py

```python

Dictionary1 = {'A': 'hello', 'B': 'For', }
Dictionary2 = {'B': 'bye'}

# Dictionary before Updation
print("Original Dictionary:")
print(Dictionary1)

# update the value of key 'B'
Dictionary1.update(Dictionary2)
print("Dictionary after updation:")
print(Dictionary1)

```

```
Original Dictionary:
{'A': 'hello', 'B': 'For'}
Dictionary after updation:
{'A': 'hello', 'B': 'bye'}
```

main.py

```python
# Dictionary with three items
Dictionary1 = { 'A': 'hello', 'B': 4, 'C': 'bye' }

print("Dictionary items:")

# Printing all the items of the Dictionary
print(Dictionary1.items())

```

```
Dictionary items:
dict_items([('A', 'hello'), ('B', 4), ('C', 'bye')])
```

Gateway classes

GW GATEWAY CLASSES

LIST (AKTU 2022-23)	DICTIONARY
The list is a collection of index value pairs like that of the array in C++.	The dictionary is a hashed structure of **the key and value** pairs.
The list is created by placing elements in **[ ]** separated by commas “,”	The dictionary is created by placing elements in **{ }** as “key”:”value”, each key-value pair is separated by commas “, “
The elements are accessed via indices.	The elements are accessed via key-value pairs.
The order of the elements entered is maintained.	There is no guarantee for maintaining order.
Lists are orders, mutable, and can contain duplicate values.	Dictionaries are unordered and mutable but they cannot contain duplicate keys.

Gateway classes

Append() AKTU (2022-23)	Extend()
Append single element at the end of the list	Append multiple elements fro, an iterable at the end of the list
Accepts single element as an argument	Accept an iterable (e.g. list,tuple, string) as an argument
List AKTU(2022-23)	**tuples**
List are mutable	Tuples are immutable
List are usually slower than tuples	They are faster than list
list1=[10,20,30]	Tup1=(10,20,30)

Gateway Classes

Array AKTU 2019-20)	list
Arrays have a fixed size set during creation.	Lists are dynamic and can change in size during runtime.
All elements in an array must be of the same data type.	Lists can accommodate elements of different data types.
Memory for the entire array is allocated at once during initialization.	Lists dynamically allocate memory as elements are added or removed.
Arrays are less flexible as their size cannot be easily changed.	Lists are more flexible, allowing easy addition or removal of elements.

GATEWAY CLASSES
Gateway classes

What will bet the output ? AKTU(2021-22)

main.py

```
1 def printalpha(abc_list, num_list):
2     for char in abc_list:
3         for num in num_list:
4             print(char,num)
5 printalpha(["a","b","c"],[1,2,3])
6
```

input

```
a 1
a 2
a 3
b 1
b 2
b 3
c 1
c 2
c 3
```

Write a python function to convert a decimal number to its binary ,octal and hexadecimal.

main.py

```
1 def bin_oct_hex(a):
2     print(bin(a))
3     print(oct(a))
4     print(hex(a))
5 bin_oct_hex(10)
6
```

```
0b1010
0o12
0xa
```

GATEWAY CLASSES
Gatewayclasses

Write a python function removekth (s,k) that takes as input a string and integer k>=0 and remove the character at index k. if k is beyond the length of s, the whole of s is returned

For example

removekth(“python”,2) return pyhon

Removekth(“mango”,1)return mngo

AKTU (2023-24 ODD)

AKTU (2020-21)

main.py

```
def removekth(s,k):
    str=""
    for i in range(len(s)):
        if i!=k:
            str=str+s[i]
    return str
print(removekth("mango",1))
print(removekth("mango",4))
print(removekth("mango",16))
```

```
mngo
mang
mango
```

Gateway Classes

Write a program factor(N) that return a list of all positive divisor of N (N>=1). AKTU (2020-21)

```python
def factor(num):
    flist=[]
    if num>=1:
        for i in range(1,num+1):
            if num%i==0:
                flist.append(i)
    return flist
print(factor(6))
```

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

Write a python program to add item in a tuple. AKTU (2019-20)

```python
#first method
t=(10,20,40,70,90)
print(t)
t=t+(20,)
print(t)
#second method
l=list(t)
l.append(3)
t=tuple(l)
print(t)
```

```
(10, 20, 40, 70, 90)
(10, 20, 40, 70, 90, 20)
(10, 20, 40, 70, 90, 20, 3)
```

GATEWAY CLASSES
Gateway classes

Write a python code to find out occurrence of an elements in a list.

```
1  vowels=['a','a','e','i','a','o','u','u']
2  print("count of a",vowels.count('a'))
3  print("count of e",vowels.count('e'))
4  print("count of i",vowels.count('i'))
5  print("count of o",vowels.count('o'))
6  print("count of u",vowels.count('u'))
```

input

```
count of a 3
count of e 1
count of i 1
count of o 1
count of u 2
```

Gateway classes

Write a python program ,count_square(N) that returns the count of perfect square less than or equal to N(N>=1) for example count_square(1) return 1 only prefect square less than equal to 1

count_square(5) return 2 (1, 4 are the prefect square less than equal to 2

count_square(55) return 7 (1,4,9,16,25,36,49)<=55 AKTU (2020-21)

```python
import math
def count_square(N):
    if N <= 1:
        return 0
    # Find the largest integer whose square is less than or equal to N
    largest_integer = int(math.sqrt(N))
    # The number of perfect squares less than or equal to N is the same
    # as the largest integer whose square is less than or equal to N
    return largest_integer
N = 55
print("count of perfect square less than or eqaul to",count_square(N))
```

input

```
count of perfect square less than or eqaul to 7
```

Gateway classes

Ques. Write a python function, searchMany(s,x,k) that takes an argument a sequence s and integersx,k(k>0) the function returns true if there are at most k occurrences of x in s .otherwise it returns false searchMany ([10,17,15,12],15,1) return true, searchMany([10,12,12,12],12,2) return falseAKTU (2020-21)

```
def searchMany(s,x,k):
    count=0
    for i in s:
        if i==x:
            count=count+1
    if count==k:
        return True
    else:
        return False

print(searchMany([10,17,15,12],15,1))
print(searchMany([10,17,15,12],17,3))
```

```
True
False
```

GATEWAY CLASSES
Gateway classes

A website requires the users to input username and password to register. Construct a program to check the validity of password input by users. Following are the criteria for checking the password

(Aktu 2023-24 ODD

1. At least 1 letter between [a-z]
2. At least 1 number between [0-9]
3. At least 1 letter between [A-Z]
4. At least 1 character from [$#@]
5. Minimum length of transaction password: 6
6. Maximum length of transaction password: 12
7. Your program should accept a sequence of comma separated passwords and will check them according to the above criteria. Passwords that match the criteria are to be printed, each separated by a comma

GATEWAY CLASSES
Gateway classes

GATEWAY CLASSES

main.py

```python
import re

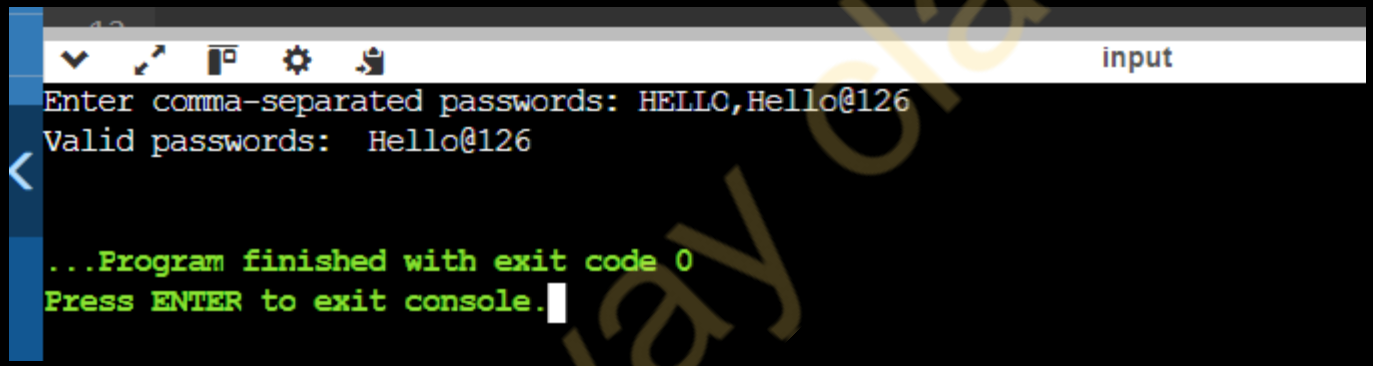
def is_valid_password(password):
    # Check if the password meets all the criteria
    if (6 <= len(password) <= 12 and
        re.search(r'[a-z]', password) and
        re.search(r'[0-9]', password) and
        re.search(r'[A-Z]', password) and
        re.search(r'[$#@]', password)):
        return True
    return False

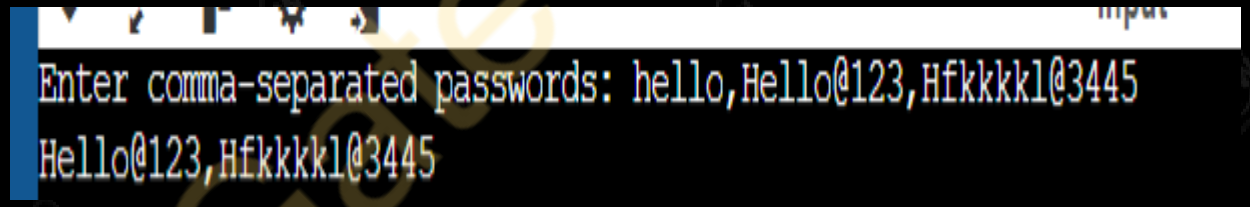
def check_passwords(passwords):
    # Split the input string into individual passwords
    password_list = passwords.split(',')
    # Filter valid passwords
    valid_passwords = [password for password in password_list if is_valid_password(password)]
    # Join the valid passwords with a comma and return
    return ','.join(valid_passwords)

# Example usage
passwords = input("Enter comma-separated passwords: ")
valid_passwords = check_passwords(passwords)
print("Valid passwords: ", valid_passwords)
```

Gateway classes

The import re statement in Python is used to import the re module, which provides support for working with regular expressions. Regular expressions (regex) are patterns used to match character combinations in strings. The re module allows you to perform tasks such as searching, matching, and replacing text using these patterns.





Gateway classes

FUNCTION

- **Python Functions is a block of statements that return the specific task. The idea is to put some commonly or repeatedly done tasks together and make a function so that instead of writing the same code again and again for different inputs, we can do the function calls to reuse code contained in it over and over again.**
- **Some Benefits of Using Functions**
- **Increase Code Readability**
- **Increase Code Reusability**

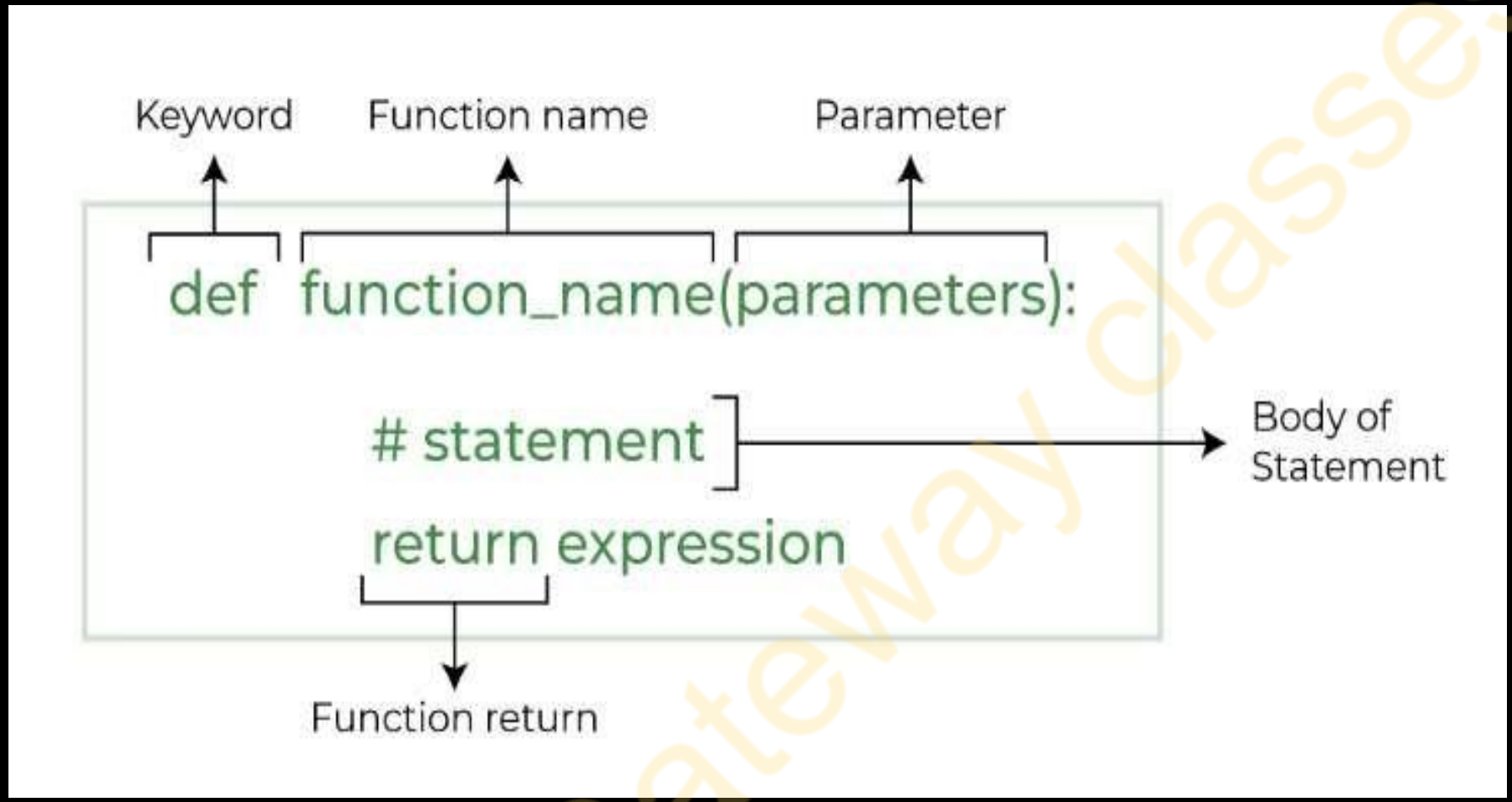
Types of Functions in Python

There are mainly two types of functions in Python

- Built-in library function: These are Standard functions in Python that are available to use.
- User-defined function: We can create our own functions based on our requirements.

Gateway classes

Python Function Declaration



GATEWAY CLASSES
Gateway classes

Python Function

- **Syntax of function definition :**
- def function name(arg 1, arg 2):

{

.....

}

Syntax of function call

Function_name(arg1,arg2)

Gateway classes

Python Function

- **Keyword:** the keyword ‘def’ is used to define function header.
- **Function name:** we define the function name for identification or to uniquely identify the function.
- **(:)** a colon to mark the end of function header.
- **Arguments** these are the values passed to the function between parentheses
- **Body of the function**: the body process the argument to do something use ful
- An optional **return statement** to return the value from the function.
- **function call** To execute a function , we have to call it

Gateway classes

Creating a Function in Python

- We can define a function in Python, using the def keyword. We can add any type of functionalities and properties to it as we require.

```
1  # A simple Python function
2  def fun():
3      print("Welcome ")
4
5
6  # Driver code to call a function
7  fun()
8
9
```

```
Welcome

...Program finished with exit code 0
Press ENTER to exit console.
```

Gateway classes

Python Function with Parameters

- If you have experience in C/C++ or Java then you must be thinking about the return type of the function and data type of arguments. That is possible in Python as well (specifically for Python 3.5 and above).
- Defining and calling a function with parameters
- def function_name(parameter: data_type) -> return_type:
- """Docstring"""
- # body of the function
- return expression.

Gateway classes

Python Function with Parameters

```python
def add(num1: int, num2: int) -> int:
    """Add two numbers"""
    num3 = num1 + num2

    return num3

# Driver code
num1, num2 = 5, 15
ans = add(num1, num2)
print(f"The addition of {num1} and {num2} results {ans}.")
```

input

```
The addition of 5 and 15 results 20.


...Program finished with exit code 0
Press ENTER to exit console.
```

Gateway classes

Types of Python Function Arguments AKTU (2022-23)

- Python supports various types of arguments that can be passed at the time of the function call. In Python, we have the following 4 types of function arguments.
- Default argument
- Keyword arguments (named arguments)
- Positional arguments
- Arbitrary arguments (variable-length arguments *args and **kwargs)

Gateway classes

Default Arguments

- A default argument is a parameter that assumes a default value if a value is not provided in the function call for that argument

```
1  # Python program to demonstrate
2  # default arguments
3  def myFun(x, y=50):
4      print("x: ", x)
5      print("y: ", y)
6
7
8  # Driver code
9  myFun(10)
10 myFun(11,20)
11
12
```

input

```
:  10
:  50
:  11
:  20
```

Gateway classes

Keyword Arguments

- The idea is to allow the caller to specify the argument name with values so that the caller does not need to remember the order of parameters.

```
1  # Python program to demonstrate Keyword Arguments
2  def student(firstname, lastname):
3      print(firstname, lastname)
4
5
6  # Keyword arguments
7  student(firstname='python', lastname='Practice')
8  student(lastname='Practice', firstname='python')
9
10
11
```

input

```
ython Practice
ython Practice


...Program finished with exit code 0
ress ENTER to exit console.
```

Gateway classes

Keyword Arguments

- Keyword-only arguments mean whenever we pass the arguments(or value) by their parameter names at the time of calling the function in Python in which if you change the position of arguments then there will be no change in the output.
- Benefits of using Keyword arguments over positional arguments
- On using keyword arguments you will get the correct output because the order of argument doesn't matter provided the logic of your code is correct. But in the case of positional arguments, you will get more than one output on changing the order of the arguments..

Gateway classes

Positional Arguments

- Position-only arguments mean whenever we pass the arguments in the order we have defined function parameters in which if you change the argument position then you may get the unexpected output. We should use positional Arguments whenever we know the order of argument to be passed.
- So now, we will call the function by using the position-only arguments in two ways, and In both cases, we will be getting different outputs from which one will be correct and another one will be incorrect

- We used the Position argument during the function call so that the first argument (or value) is assigned to name and the second argument (or value) is assigned to age. By changing the position, or if you forget the order of the positions, the values can be used in the wrong places, as shown in the Case-2 example below, where 27 is assigned to the name and Suraj is assigned to the age

Gateway classes

Positional Arguments

```python
def nameAge(name, age):
    print("Hi, I am", name)
    print("My age is ", age)


# You will get correct output because
# argument is given in order
print("Case-1:")
nameAge("Suraj", 27)
# You will get incorrect output because
# argument is not in order
print("\nCase-2:")
nameAge(27, "Suraj")
```

input

```
Hi, I am Suraj
My age is  27

Case-2:
Hi, I am 27
My age is  Suraj
```

Gateway classes

Arbitrary Keyword Arguments

- In Python Arbitrary Keyword Arguments, *args, and **kwargs can pass a variable number of arguments to a function using special symbols. There are two special symbols:
- *args in Python (Non-Keyword Arguments)
- **kwargs in Python (Keyword Arguments)

Gateway classes

*args in Python (Non-Keyword Arguments)

- We can declare a variable length length argument with the*symbol
- The *args allows a function to accept any number of positional arguments
- The arguments are collected as a tuple within the function

Gateway classes

*args in Python (Non-Keyword Arguments)

main.py

```
# beginning of the method
def sum(*args):
    resultfinal = 0
#beginning of for loop
    for num in args:
        resultfinal = resultfinal + num

    return resultfinal
#printing the values
print(sum(10, 20))              # 30
print(sum(10, 20, 30))          # 60
print(sum(10, 20, 2))           # 32
```

```
30
60
32
```

Gateway classes

***kwargs in Python (Keyword Arguments)

- The *kwargs parameter allows a function to accept any number of keyword arguments.
- The arguments are collected as a dictionary within the function

```python
# Let's write a Python program
# *kwargs for a variable number of keyword arguments

def myPrg(**kwargs):
    for key, value in kwargs.items():
        print (key, "==" , value)

# Driver code for kwargs in python
myPrg(first ='Hello', mid ='Welcome', last='Hello')
```

input

```
first == Hello
mid == Welcome
last == Hello


...Program finished with exit code 0
Press ENTER to exit console.
```

Gateway classes

Calculator using function AKTU (2019-20)

main.py

```
# Python program for simple calculator

# Function to add two numbers
def add(num1, num2):
    return num1 + num2

# Function to subtract two numbers
def subtract(num1, num2):
    return num1 - num2

# Function to multiply two numbers
def multiply(num1, num2):
    return num1 * num2

# Function to divide two numbers
def divide(num1, num2):
    return num1 / num2

print("Please select operation -\n" \
      "1. Add\n" \
      "2. Subtract\n" \
      "3. Multiply\n" \
      "4. Divide\n")
```

GATEWAY CLASSES
Gateway classes

Calculator using function

```
# Take input from the user
select = int(input("Select operations form 1, 2, 3, 4 :"))

number_1 = int(input("Enter first number: "))
number_2 = int(input("Enter second number: "))

if select == 1:
    print(number_1, "+", number_2, "=",
                    add(number_1, number_2))

elif select == 2:
    print(number_1, "-", number_2, "=",
                    subtract(number_1, number_2))

elif select == 3:
    print(number_1, "*", number_2, "=",
                    multiply(number_1, number_2))

elif select == 4:
    print(number_1, "/", number_2, "=",
                    divide(number_1, number_2))
else:
    print("Invalid input")
```

Gateway classes

Calculator using function

```
Please select operation -
1. Add
2. Subtract
3. Multiply
4. Divide

Select operations form 1, 2, 3, 4 :2
Enter first number: 7
Enter second number: 8
7 - 8 = -1


...Program finished with exit code 0
Press ENTER to exit console.
```

Gateway classes

Python Scope variable AKTU (2019-20)

- The location where we can find a variable and also access it if required is called the scope of a variable
- **Python Local variable**
- Local variables are those that are initialized within a function and are unique to that function. It cannot be accessed outside of the function

```
1
2  def f():
3
4      # local variable
5      s = "I love India"
6      print(s)
7
8
9  # Driver code
10 f()
11
```

```
I love India
```

Gateway classes

Python Scope variable

- **Python Local variable**
- **If we will try to use this local variable outside the function**

```
1  def f():
2
3      # local variable
4      s = "I love python"
5      print("Inside Function:", s)
6
7  # Driver code
8  f()
9  print(s)
10
```

input

```
Inside Function: I love python
Traceback (most recent call last):
  File "/home/main.py", line 9, in <module>
    print(s)
NameError: name 's' is not defined


...Program finished with exit code 1
Press ENTER to exit console.
```

Gateway classes

Python Scope variable

- **Python Global variables**
- **Global variables are the ones that are defined and declared outside any function and are not specified to any function. They can be used by any part of the program.**

```
1  # This function uses global variable s
2  def f():
3      print(s)
4
5
6  # Global scope
7  s = "I love python"
8  f()
9
10
```

input

```
I love python


...Program finished with exit code 0
Press ENTER to exit console.
```

Gateway classes

GENERATORS(AKTU 2023-24)

- A generator function is like a normal function but uses the yield keyword instead of return.
- When called, it does not execute immediately but returns a generator object (an iterator).
- This generator object can be used to fetch values one by one using next().
- The function remembers its state after yielding a value and resumes execution when next() is called again.

```
def my_generator():
    yield 10
    yield 20
gen = my_generator()
# gen is an iterator (generator object)
print(next(gen))
print(next(gen))
```

```
10
20
```

Gateway Classes

```
def fun(max):
    cnt = 1
    while cnt <= max:
        yield cnt
        cnt += 1
ctr = fun(5)
for n in ctr:
    print(n)
```

```
1
2
3
4
5
```

lambda function

- **Python Lambda Functions** are anonymous functions means that the function is without a name.
- As we already know the *def* keyword is used to define a normal function in Python.
- Similarly, the *lambda* keyword is used to define an anonymous function in Python.

SYNTAX:

Syntax: *lambda arguments : expression*

Gateway classes

- **lambda:** The keyword to define the function.
- **arguments:** A comma-separated list of input parameters (like in a regular function).
- **expression:** A single expression that is evaluated and returned.

lambda function to add two numbers

```
1  add = lambda x, y: x + y
2  print(add(2, 3))
```

```
5
```

Square a Number

main.py

```
1  square = lambda n: n * n
2  print(square(4))
```

```
16
```

Gateway classes

How lambda function can be used within In a list comprehension

You can use lambda functions inside list comprehensions to perform quick operations on elements.

```
1  nums = [1, 2, 3, 4, 5]
2  squared = [(lambda x: x**2)(x) for x in nums]
3  print(squared)
4
```

input

```
[1, 4, 9, 16, 25]
```

Gateway classes

Write a python program that use lambda function within a list comprehension to convert the list of temperature in Celsius to Fahrenheit

```
celsius_temps = [0, 10, 20, 30, 40, 100]
fahrenheit_temps = [(lambda c: (c * 9/5) + 32)(c) for c in celsius_temps]
print(fahrenheit_temps)
```

```
[32.0, 50.0, 68.0, 86.0, 104.0, 212.0]
```

Gateway classes

Find the LCM OF TWO NUMBER

```
def lcm(a, b):
    def gcd(x, y):
        while y:
            x, y = y, x % y
        return x
    return abs(a * b) // gcd(a, b)
num1 = int(input("Enter first number: "))
num2 = int(input("Enter second number: "))
print(f"The LCM of {num1} and {num2} is {lcm(num1, num2)}")
```

```
Enter first number: 12
Enter second number: 15
The LCM of 12 and 15 is 60
```

Gateway classes

Thank You

Gatewayclasses