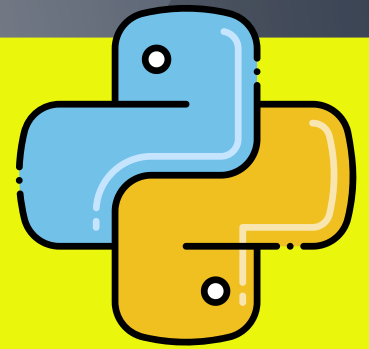
AKTU

B.Tech III-Sem

GATEWAY CLASSES

Python

Crash Course



Unit-1 in One Shot

Pragya Ma'am

Gateway Classes

IMPORTANT QUESTION OF UNIT 1

1.	What is dynamic typing? Explain with example and also explain the advantages and disadvantages of it
2.	Write a program of swapping of two number with the help of third variable / without the help of third/without the help of third varaible
3.	What is type conversion?
4.	What is data type? Explain different type of data type used in python
5.	Solve the following question a & b << 2 // 5 ** 2 + c ^ b a=3,b=5,c=10
6	Solve the following question a**2<<2>>b**2^c**3 a=3, b=5,c=10

Gateway classes

PYTHON PROGRAMMING

AKTU SYLLABUS UNIT-1

Introduction to Python: Python variables, Python basic Operators, Understanding python blocks. Python Data Types, Declaring and using Numeric data types: int, float etc.

GATEWAY CLASSES
Gateway classes

PYTHON PROGRAMMING

UNIT -1 :Introduction to Python

Lec-1

Today's Target

- Basic about python
- Variable, print(), input()
- Some basic program
- AKTU PYQs

Gateway classes

What is python and its features?

- Python is a **high-level, general-purpose** programming language that is widely used for various applications like **web development, data science, automation, artificial intelligence,** and more.
- It was created by **Guido van Rossum** and released in **1991**.

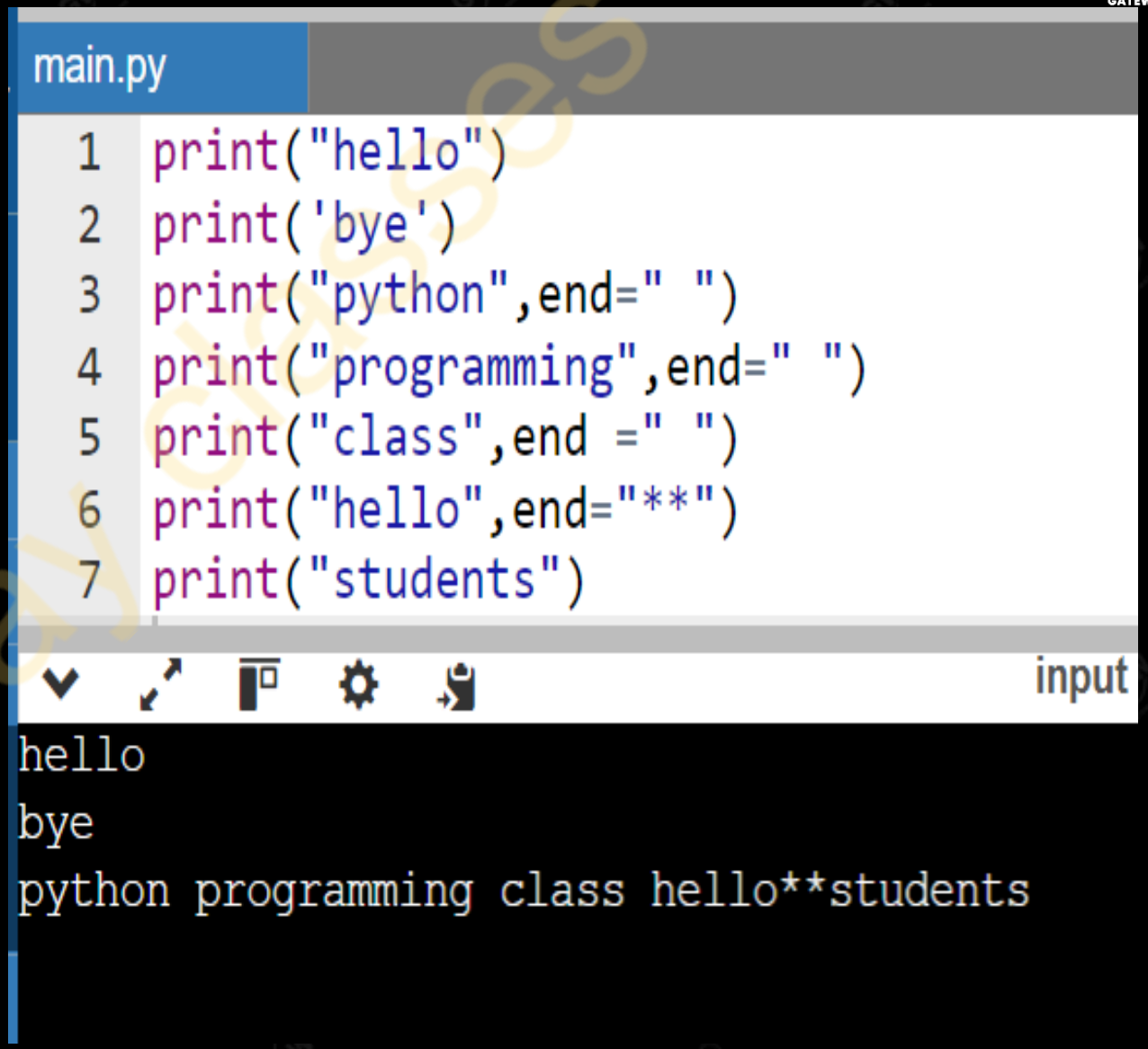
Features:

1. **Simple** and Easy to Learn:
2. It is an **interpreted language.**
3. Programs written in Python can run on various **operating systems** like Windows, macOS, Linux, etc., without any modifications. **(Cross platform compatible)**
4. Python is **open-source.**
5. Python is **dynamically typed**, meaning you don't need to declare variable types explicitly. Python automatically assigns data types at runtime.

Gateway Classes

Print ( )

- print( ) function prints the message to the screen or any other standard output device.
- The end keyword is used to specify the content that is to be printed at the end of the execution of the print( ) function.
- By default, it is set to "\n", which leads to the change of line after the execution of print( ) statement



GATEWAY CLASSES
Gateway Classes

Variable

- Python Variable is **containers that store values.**
- Python is not **"statically typed".** We do not need to declare variables before using them or declare their type.
- The value stored in a variable can be changed during program execution.
- A **Variables** in Python is only a name given to a **memory location.**

```
1  x = 10
2  print(x)
3  print(type(x))
4  x = "Hello"
5  print(x)
6  print(type(x))
7  x = 5.7
8  print(x)
9  print(type(x))
```

```
10
<class 'int'>
Hello
<class 'str'>
5.7
<class 'float'>
```

GATEWAY CLASSES
Gateway Classes

Input ( )

- This function first takes the input from the user and converts it into a string.
- The type of the returned object always will be <class 'str'>.

main.py

```
1 x=input("enter the number\n")
2 print(x)
3 print(type(x))
```

```
enter the number
3
3
<class 'str'>
```

- But if we want to input the data of any other type then we must convert it to that type (type casting/Explicit conversion)

main.py

```
1 a=int(input("enter the number\n"))
2 b=int(input("enetr the number\n"))
3 c=a+b
4 print("sum=",c)
```

input

```
enter the number
5
enetr the number
6
sum= 11
```

GATEWAY CLASSES
Gateway Classes

What is dynamic typing? (AKTU 2023-24 EVEN)

- When you assign a value to a variable, Python's interpreter automatically infers the variable's type based on the assigned value.
- The type of a variable can change during the execution of the program, as Python doesn't enforce strict typing.
- Python automatically determines the type of the variable based on the value assigned to it at runtime.

```
1  x = 10
2  print(x)
3  print(type(x))
4  x = "Hello"
5  print(x)
6  print(type(x))
7  x = 5.7
8  print(x)
9  print(type(x))
```

```
10
<class 'int'>
Hello
<class 'str'>
5.7
<class 'float'>
```

GATEWAY CLASSES

What is dynamic typing? (AKTU 2023-24 EVEN)

main.py

```
1  name = input("Enter your name: ")
2  age = int(input("Enter your age: "))
3  print("Hello, " + name + "! You are " + str(age) + " years old.")
4
```

input

```
Enter your name: XYZ
Enter your age: 20
Hello, XYZ! You are 20 years old.
```

KEYPOINTS:

1. **The type of the variable is determined when a value is assigned.**
2. **No need to declare the type of a variable.**
3. **Variables can store different types of values at different times in the program, this provide greater flexibility.**

GATEWAY CLASSES
Gateway classes

ADVANTGES	DISADVANTAGES
➢ **Ease of Coding:** You don't have to worry about declaring types explicitly, making code faster and easier to write, especially for beginners. ➢ **Flexibility:** You can reuse the same variable for different types of data, which can be convenient in certain cases like handling multiple data types dynamically.	**Runtime Errors:** Errors related to type mismatches are only caught during execution (runtime), making it harder to catch bugs early compared to statically typed languages. **Less Explicit Code**: Since types are not declared, it may be harder to understand the intended usage of variables, leading to confusion and less maintainable code in large programs.

Gateway classes

GATEWAY CLASSES

GATEWAY CLASSES

```
1  age = 25
2  age = "twenty-five"
3  age = age + 5
```

input

```
Traceback (most recent call last):
  File "/home/main.py", line 3, in <module>
    age = age + 5
TypeError: can only concatenate str (not "int") to str
```

Gateway classes

Write a program of swapping of two number(using third /intermediate/temporary variable)

AKTU 2021-22

```
a=int(input("enter the first number\n "))
b=int(input("enter the second number\n"))
print("value of a and b before swapping")
print("the value of a",a)
print("the value of b",b)
c=a
a=b
b=c
print("value of a and b after swapping")
print("the value of a",a)
print("the value of b",b)
```

```
enter the first number
 8
enter the second number
9
value of a and b before swapping
the value of a 8
the value of b 9
value of a and b after swapping
the value of a 9
the value of b 8
```

GATEWAY CLASSES
Gateway classes

Write a program of swapping of two number(without any third variable)

```
a=int(input("enter the first number\n "))
b=int(input("enter the second number\n"))
print("value of a and b before swapping")
print("the value of a",a)
print("the value of b",b)
a=a+b
b=a-b
a=a-b
print("value of a and b after swapping")
print("the value of a",a)
print("the value of b",b)
```

```
enter the first number
 8
enter the second number
9
value of a and b before swapping
the value of a 8
the value of b 9
value of a and b after swapping
the value of a 9
the value of b 8
```

GATEWAY CLASSES
Gateway Classes

PYTHON

UNIT -1 : Introduction to python

Lec-2

Today's Target

- Type conversion
- Data Type
- AKTU PYQs

By Pragya rajvanshi

B.tech,M.tech

Gateway classes

Type conversion

The act of changing an **object's data type** is known as type conversion.

Implicit conversion(Automatic conversion)

- Python interpreter **automatically** converts **one data type to another** without any **user involvement.**
- There is **no loss of data.**

```
1  x=10
2  y=20.5
3  z=x+y
4  print(z)
```

```
30.5
```

Explicit conversion(type casting)

- The data type is **manually changed by** the user as per their requirement
- There is a **risk of data loss** since we are forcing an expression to be changed in some specific data type.

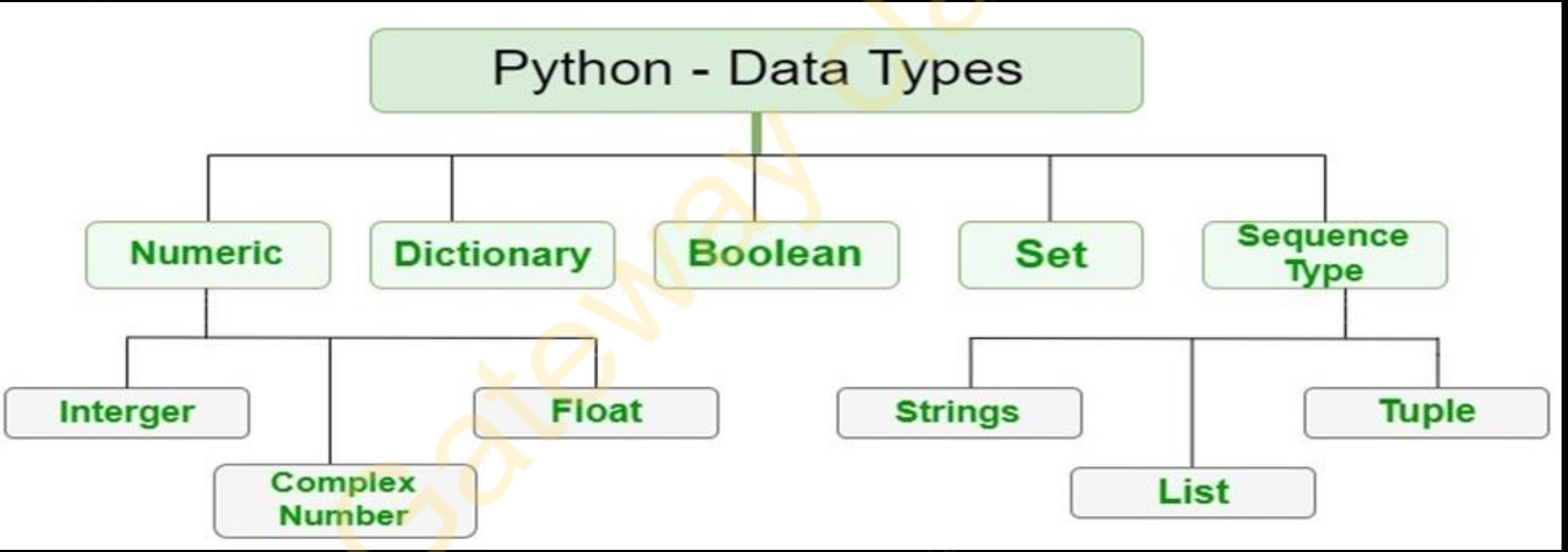
```
1  s="10010"
2  z=int(s,2)
3  print(z)
4  w=float(z)
5  print(w)
```

```
18
18.0
```

Gateway classes

DATA TYPE

- Python Data types are the classification or categorization of data items.
- A data type defines the kind of values a variable can hold and determines what operations can be performed on that data.



GATEWAY CLASSES
Gateway Classes

NUMERIC

- Python, numeric data type is used to hold **numeric values.**
- **Int -** holds signed integers of non-limited length.
- **float -** holds floating decimal points and it's accurate up to 15 decimal places.
- **complex -** holds complex numbers.

```
1  a = 5
2  print("Type of a: ", type(a))
3  b = 5.0
4  print("Type of b: ", type(b))
5  c = 2 + 4j
6  print("Type of c: ", type(c))
```

```
Type of a:  <class 'int'>
Type of b:  <class 'float'>
Type of c:  <class 'complex'>
```

GATEWAY CLASSES
Gateway Classes

SEQUENCING TYPE

String

- **The sequence of characters** in the quotation marks can be used to describe the **string.**

```
1 str1="hello"
2 str2='hi'
3 print(str1,"is of type",type(str1))
4 print(str2,"is of type",type(str2))
```

```
hello is of type <class 'str'>
hi is of type <class 'str'>
```

```
1 str1="hellostudents"
2 str2="python"
3 print(str1[0:2])
4 print(str2[2:4])
5 print(str1[3])
6 print(str2[3])
7 print(str1*2)
8 print(str2*3)
9 print(str1+str2)
```

```
he
th
l
h
hellostudentshellostudents
pythonpythonpython
hellostudentspython
```

GATEWAY CLASSES
Gateway classes

LIST

- List is an **ordered collection** of **similar or different types of items** separated by **commas** and enclosed within brackets [ ].

```
[1, 3, 4, 5]
<class 'list'>
3
5
[1, 3, 4]
[1, 3, 4, 5]
[1, 3, 4, 5, 2, 3, 4, 'hello']
[1, 3, 4, 5, 1, 3, 4, 5]
```

```
l1=[1,3,4,5]
l2=[2,3,4,"hello"]
print(l1)
print(type(l1))
#indexing
print(l1[1])
print(l1[-1])
#slicing
print(l1[0:3])
print(l1[0:])
#concatenation
print(l1+l2)
#repetition
print(l1*2)
```

Gateway Classes

Tuple

- Tuple is an **ordered sequence** of items same as a list. The only difference is that tuples are **immutable**. Tuples once **created cannot be modified.**
- we use the **parentheses ( )** to store items of a tuple.

```
tup1=(1,2,3,4)
print(type(tup1))
tup2=("hello",2.3,4,3,4)
print(tup1[1])
print(tup2[-1])
print(tup1[1:3])
print(tup1+tup2)
print(tup1*2)
tup1[1]=6
```

```
<class 'tuple'>
2
4
(2, 3)
(1, 2, 3, 4, 'hello', 2.3, 4, 3, 4)
(1, 2, 3, 4, 1, 2, 3, 4)
Traceback (most recent call last):
  File "/home/main.py", line 9, in <module>
    tup1[1]=6
TypeError: 'tuple' object does not support item assignment
```

Gateway Classes

Dictionary data type

- Dictionary is an **ordered collection of items**. It stores elements in **key/value pairs.**
- Here, **keys are unique identifiers** that are associated with each value .

```
1  S1={"sectionA":"good","sectionB":"excellent","sectionC":"good"}
2  print(S1)
3  print(type(S1))
4  print(S1.keys())
5  print(S1.values())
6  print(S1["sectionA"])
```

```
{'sectionA': 'good', 'sectionB': 'excellent', 'sectionC': 'good'}
<class 'dict'>
dict_keys(['sectionA', 'sectionB', 'sectionC'])
dict_values(['good', 'excellent', 'good'])
good
```

Gateway classes

Boolean

- The Boolean value can be of two types only i.e. either True or False.

  The output indicates the variable is a Boolean data.

```
a=True
print(a)
print(type(a))
b=False
print(b)
print(type(b))
```

```
True
<class 'bool'>
False
<class 'bool'>
```

Sets

- Set items are unordered, unchangeable, and do not allow duplicate values.

GATEWAY CLASSES
Gateway classes

GATEWAY CLASSES

```
s={"banana","grapes","papita","mango","mango"}
print(s)
print(type(s))
s2={"1","hello",2.5}
print(s2)
s[1]="bye"
print(s)
```

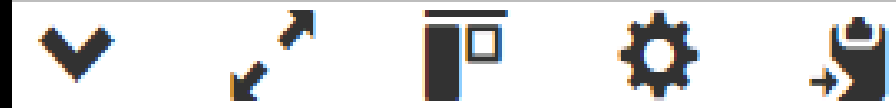
input

```
{'mango', 'grapes', 'banana', 'papita'}
<class 'set'>
{2.5, 'hello', '1'}
Traceback (most recent call last):
  File "/home/main.py", line 6, in <module>
    s[1]="bye"
TypeError: 'set' object does not support item assignment
```

Gateway classes

GATEWAY CLASSES

```
1  s1={"hi","hello","jjj","hi" }
2  print(s1)
```



```
{'hello', 'hi', 'jjj'}
```

Gateway classes

PYTHON PROGRAMMING

UNIT -1 : Introduction to Python

Lec-3

Today's Target

- Operators
- Some basic program
- AKTU PYQs

Pragya Rajvanshi

B.tech,M.tech

Gateway classes

Write a program to find out the area and perimeter of Square.

```
a=float(input("enter the side of sqaure\n"))
area=a*a
perimeter=4*a
print("area of sqaure",area)
print("perimeter of sqaure",perimeter)
```

```
enter the side of sqaure
3.4
area of sqaure 11.559999999999999
perimeter of sqaure 13.6
```

GATEWAY CLASSES
Gateway Classes

Q. **How do you read an input from a user in python to be used as an integer in the rest of the program ? Explain with an example. (AKTU)**

```
a=int(input("enter the number\n"))
```

```
enter the number
3
```

Benefits of python(interpreted , high level, OOP)

- Python is interpreted, which allows for easier debugging and code development.
- Python is portable across operating systems and interactive, allowing real-time code execution and testing.
- Python is dynamically typed, meaning you don't need to declare data types explicitly.
- Python is a high-level language that abstracts low-level details, making it more user-friendly.
- Python is open source.

GATEWAY CLASSES
Gateway classes

Ques. In some language every statement ends with semi-colon (;) what happens if you put a semicolon at the end of python statement

- A semicolon in Python signifies separation rather than termination.
- No error produce.

```
1  print("hi");print("bye");print("hello")
2
```

```
hi
bye
hello
```

GATEWAY CLASSES
Gateway Classes

OPERATOR

- These are the special symbols in Python and are used to execute an Arithmetic or Logical computation
- An Operand is a value that the operator needs to complete a task.
- Types of Operators in Python-
- Arithmetic operators.
- Comparison operators .
- Assignment operators.
- Logical operators .
- Bitwise operators.

Special operators

- Identity operators
- Membership operators
- **1 Arithmetic operators**

operator	example
+,-,*	3+3=6 5-3=2 5*3=15
/,%(modulus) **(exponent)	50/2=25, 21%2=1 10**3=1000
//(floor division)	It divides two numbers and rounds down to the nearest integer. result = 7 // 3 print(result) # Output: 2

Gateway classes

2. Comparison operator

```
1  x=10
2  y=3
3  print(x+y)
4  print(x-y)
5  print(x*y)
6  print(x/y)
7  print(x%y)
8  print(x//y)
```

```
13
7
30
3.3333333333333335
1
3
```

```
print(2**3)#exponent
```

```
8
```

```
x=10
y=3
print(x<y)#less than
print(x<=y)#less than equal to
print(x>y)#greater than
print(x>=y)#greater than equal to
print(x==y)#equal to
print(x!=y)#not equal to
```

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

GATEWAY CLASSES
Gateway Classes

3. logical Operators

- It is used to make decision based on condition.
- And - Logical AND
- Or - Logical OR
- Not - Logical not

```
x=10>3
y=3<9
print("x and y",x and y)
print("x or y",x or y)
print("not x",not(x))
```

```
x and y True
x or y True
not x False
```

4. Bitwise operator

- Bitwise operators are used in Python to perform the operations on binary numerals or bit patterns.
- It operates bit by bit.
- & Binary AND
- | Binary OR
- ^ Binary XOR
- ~ Binary Ones Complement
- << Binary Left Shift
- >> Binary Right Shift

GATEWAY CLASSES
Gateway classes

```
print(2&5)
print(2|5)
print(2^5)
print(~5)
print(5<<1)
print(5>>1)
```

```
0
7
7
-6
10
2
```

Bitwise and

	128	64	32	16	8	4	2	1
2	0	0	0	0	0	0	1	0
5	0	0	0	0	0	1	0	1
2&5	0	0	0	0	0	0	0	0

Bitwise or

	128	64	32	16	8	4	2	1
2	0	0	0	0	0	0	1	0
5	0	0	0	0	0	1	0	1
2|5	0	0	0	0	0	1	1	1

Gateway classes

GATEWAY CLASSES

Bitwise ones' complement

	128	64	32	16	8	4	2	1
5	0	0	0	0	0	1	0	1
~5	1	1	1	1	1	0	1	0

Bitwise xor

	128	64	32	16	8	4	2	1
2	0	0	0	0	0	0	1	0
5	0	0	0	0	0	1	0	1
2^5	0	0	0	0	0	1	1	1

Gateway Classes

GATEWAY CLASSES

GATEWAY CLASSES

	128	64	32	16	8	4	2	1
5	0	0	0	0	0	1	0	1
5<<1 Bitwise left shift	0	0	0	0	1	0	1	0

	128	64	32	16	8	4	2	1
5	0	0	0	0	0	1	0	1
5>>1 Bitwise right shift	0	0	0	0	0	0	1	0

Gateway classes

Membership operator

- Membership Operators are mainly used in Python to find whether a **value or variable is present in a sequence** (string, tuple, list, set, and directory).
- **"In" and "Not In"** are the two membership operators available in Python.

main.py

```
1  l1=[1,2,3,4]
2  b=2
3  if b in l1:
4      print("hello")
5  else:
6      print("bye")
7
```

```
hello
```

GATEWAY CLASSES
Gateway classes

GATEWAY CLASSES

Membership Operator	Description	Syntax
in Operator	Returns True if the value exists in a sequence, else returns False	Value/variable in sequence
not in Operator	Returns False if the value exists in a sequence, else returns True	Value/variable not in sequence

Gateway classes

main.py

```
1  l1=[1,2,3,4]
2  b=2
3  if b  not in l1:
4      print("hello")
5  else:
6      print("bye")
```

bye

main.py

```
1  l1=(1,2,3,4)
2  b=2
3  if b  not in l1:
4      print("hello")
5  else:
6      print("bye")
```

bye

```
1  l1=(1,2,3,4)
2  b=2
3  if b in l1:
4      print("hello")
5  else:
6      print("bye")
```

hello

GW GATEWAY CLASSES
Gateway classes

```python
l1="pythonprogramming"
b="l"
if b in l1:
    print("hello")
else:
    print("bye")
```

bye

```python
s={"classA":3,"CLASSB":2,"CLASSC":7}
b="classA"
if b in s:
    print("bbbb")
else:
    print("ccccc")
```

input

bbbb

```python
l1="pythonprogramming"
b="l"
if b not in l1:
    print("hello")
else:
    print("bye")
```

hello

```python
s={"classA":3,"CLASSB":2,"CLASSC":7}
b="classA"
if b not in s:
    print("bbbb")
else:
    print("ccccc")
```

input

ccccc

Gateway classes

Identity Operators

Identity operators are used to compare the objects if both the objects are actually of the same data type and share the same memory location.

Identity Operator	Description	Syntax
is Operator	Returns True if both objects refers to same memory location, else returns False	obj1 is obj2
is not Operator	Returns False if both object refers to same memory location, else returns True	obj1 is not obj2

GATEWAY CLASSES
Gateway classes

Variable

```
x=2
y=5
print(type(x))
print(type(y))
print(id(x))
print(id(y))
print(x is y)
print(x is not y)
```

```
<class 'int'>
<class 'int'>
1271795638602 40
1271795638603 36
False
True
```

Variable

```
x=2
y=2
print(type(x))
print(type(y))
print(id(x))
print(id(y))
print(x is y)
print(x is not y)
```

```
<class 'int'>
<class 'int'>
133606456836368
133606456836368
True
False
```

GATEWAY CLASSES
Gateway Classes

Tuple

```
x=(1,2,3,4)
y=(1,2,3,4)
print(type(x))
print(type(y))
print(id(x))
print(id(y))
print(x is y)
print(x is not y)
```

```
<class 'tuple'>
<class 'tuple'>
134520851275968
134520851275968
True
False
```

string

```
x="hello"
y="hello"
print(type(x))
print(type(y))
print(id(x))
print(id(y))
print(x is y)
print(x is not y)
```

```
<class 'str'>
<class 'str'>
129451994184752
129451994184752
True
False
```

GATEWAY CLASSES
Gateway classes

List

```
x=[1,2,3,4]
y=[1,2,3,4]
print(type(x))
print(type(y))
print(id(x))
print(id(y))
print(x is y)
print(x is not y)
```

```
<class 'list'>
<class 'list'>
134489793223744
134489793300672
False
True
```

GATEWAY CLASSES
Gateway classes

PYTHON PROGRAMMING

UNIT -1 :Introduction to Python

Lec-4

Today's Target

- Global and local variable
- Operator Precedence and Associativity
- Single line block and multi line block
- AKTU PYQs

Gateway classes

Difference between-

	GLOBAL VARAIBLE	LOCAL VARAIBLE
Definition	Declared outside the functions.	Declared within the functions.
Data Sharing	It Offers Data Sharing.	It doesn't offers Data Sharing.
Scope	It can be access throughout the code.	It can access only inside the function.
Lifetime	They are created when the execution of the program begins and are lost when the program is ended.	They are created when the function starts its execution and are lost when the function ends.
Value	Once the value changes it is reflected throughout the code.	once changed the variable don't affect other functions of the program.

GATEWAY CLASSES
Gateway classes

Local variable

```
1  def a():
2      y = 5
3      print(y)
4
5  a()
6  print(y)
```

input

```
5
Traceback (most recent call last):
  File "/home/main.py", line 6, in <module>
    print(y)
NameError: name 'y' is not defined
```

Global Variable

```
1  x = 10
2  def a():
3      print(x)
4
5  a()
```

```
10
```

GATEWAY CLASSES
Gateway classes

Operator precedence and associativity

- In Python, operators have different levels of precedence, which determine the order in which they are evaluated.
- When multiple operators are present in an expression, the ones with higher precedence are evaluated first.
- In the case of operators with the same precedence, their associativity comes into play, determining the order of evaluation

Precedence	Operator Sign	Operator Name
Highest	**	Exponentiation
	+x, -x, ~x	Unary positive, unary negative, bitwise negation
	*, /, //, %	Multiplication, division, floor, division, modulus
	+, -	Addition, subtraction
	<<, >>	Left-shift, right-shift
	&	Bitwise AND
	^	Bitwise XOR
	|	Bitwise OR
	==, !=, <, <=, >, >=, is, is not	Comparison, identity
	not	Boolean NOT
	and	Boolean AND
Lowest	or	Boolean OR

Gateway Classes

Precedence	Operators	Description	Associativity
1	( )	Parentheses	Left to right
2	x[index], x[index:index]	Subscription, slicing	Left to right
3	await	Await expression	N/A
4	**	Exponentiation	Right to left
5	+x, -x, ~x	Positive, negative, bitwise NOT	Right to left
6	*, @, /, //, %	Multiplication, matrix, division, floor division, remainder	Left to right

GATEWAY CLASSES
Gateway classes

Precedence	Operators	Description	Associativity
7	+, –	Addition and subtraction	Left to right
8	<<, >>	Shifts	Left to right
9	&	Bitwise AND	Left to right
10	^	Bitwise XOR	Left to right
11	|	Bitwise OR	Left to right
12	in, not in, is, is not, <, <=, >, >=, !=, ==	Comparisons, membership tests, identity tests	Left to Right

GATEWAY CLASSES
Gateway classes

Precedence	Operators	Description	Associativity
13	not x	Logical/Boolean NOT	Right to left
14	and	Logical/Boolean AND	Left to right
15	or	Logical/Boolean OR	Left to right
16	if-else	Conditional expression	Right to left
17	lambda	Lambda expression	N/A
18	:=	Assignment expression (walrus operator)	Right to left

GATEWAY CLASSES
Gateway classes

GATEWAY CLASSES

GATEWAY CLASSES

```
print(2**3**2)# right to left associativity
print(2<<3<<2)#left to right associativity
print(2<3<3>7)#left to right
print(3==3==2)#left to right associative
print(2 or 3)
print(2 or 3 or 5)#left to right
print(2 and 5 and 6)#left to right
print(2 and 5 and 2)
print(2 and 0 and 5)
print(not(not(2)))
```

```
512
64
False
False
2
2
6
2
0
True
```

Gateway classes

Solve the following question

a & b << 2 // 5 ** 2 + c ^ b

a=3,b=5,c=10

3&5<<2//5**2+10^5

((1) ** (2) // (3) + (4) << (5) & (6)^ precedence)

3& 5<<2//25+10^5

3&5<<0+10^5

3&5<<10^5

3&5120^5

0^5=5

(Note a ^0=a and a^1=a')

Solve the following question

a**2<<2>>b**2^c**3

a=3, b=5,c=10

Precedence 1)**(right to left)

2)<<,>>(left to right)

3)^(left to right)

3**2<<2>>5**2^10**3

3**2<<2>>5**2^1000

3**2<<2>>25^1000

9<<2>>25^1000

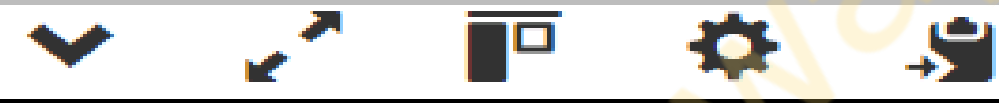
36>>25^1000

0^1000=1000

Gateway classes

What will the output of the program (AKTU2020-21)

```
1  a=0
2  b=2
3  c=3
4  x=c or a
5  print(x)
```



```
3
```

GATEWAY CLASSES
Gateway classes

Questions

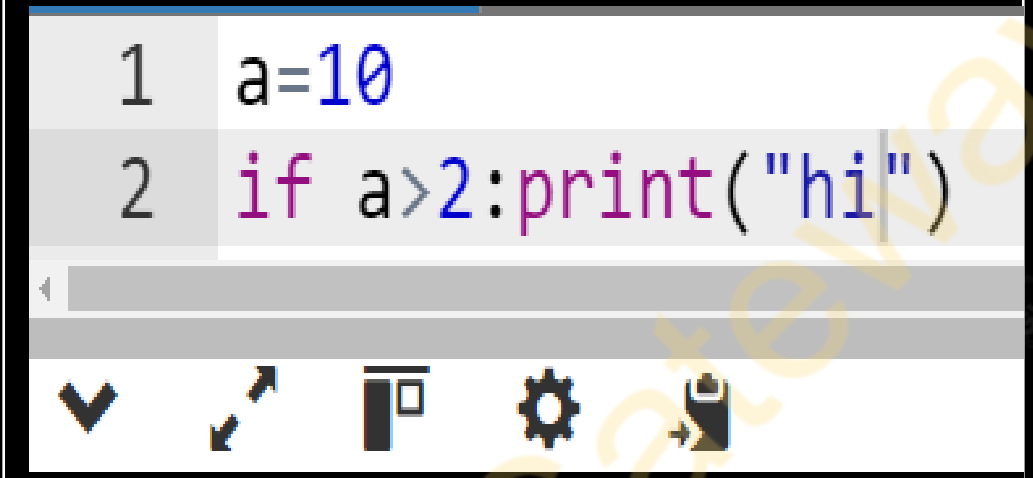
```
l1=[1,2,3,4,5]
print(l1[0:3])
print(l1[3:0])
print(l1[2:-1])
print(l1[-1:-2])
print(l1[0:4:2])
print(l1[1:4:-1])
print(l1[-4:-1:2])
print(l1[-4])
```

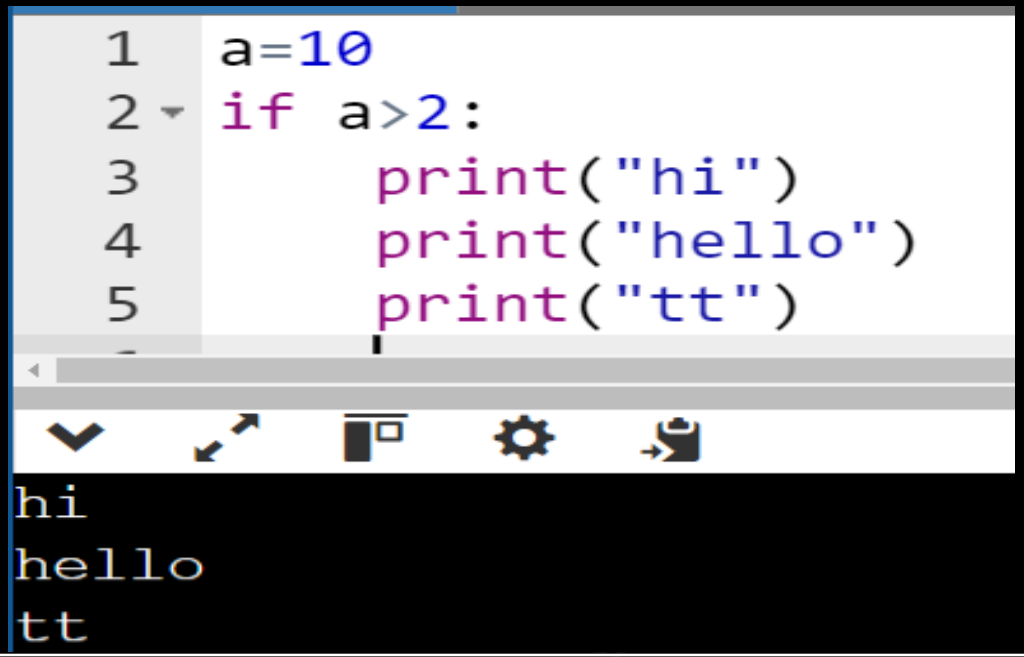
```
[1, 2, 3]
[]
[3, 4]
[]
[1, 3]
[]
[2, 4]
```

GATEWAY CLASSES
Gateway Classes

Difference between-

Single-line block	Multi-line block
Code is written in a single line.	Code spans multiple lines with proper indentation.
Might be more challenging to identifying and fix error within a single line	Easier to identify and correct error due to the structured layout





GATEWAY CLASSES
Gateway classes



Gateway Classes