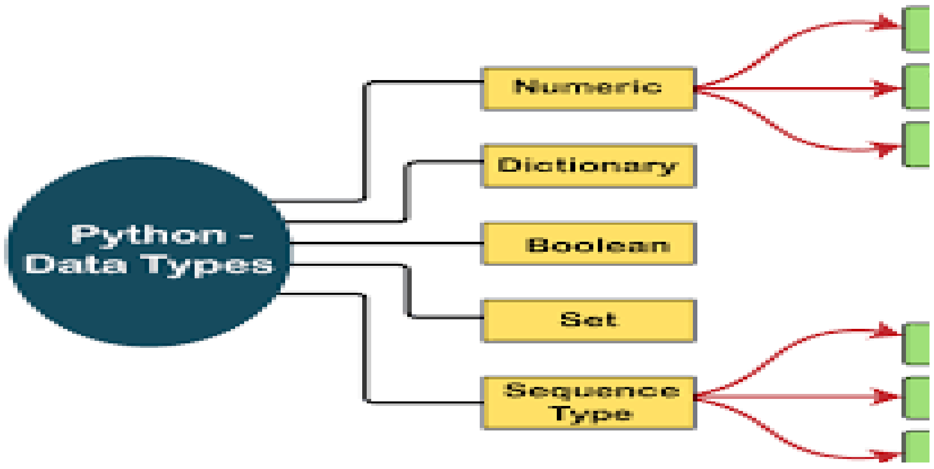
CHAPTER -8 DATA HANDLING

8.1 Introduction	• Data type • Mutable and Immutable Types • Operators • Expression • Python Standard library modules • Debugging
8.2 Data types	Python provides a predefined set of data types for handling the data it uses. Data can be stored in any of these data types .



Examples.

Data-information-RN,age,address,fname,mobile,email id
24,67,1011,240000, 04/02/2004, 67.45,5677.36, ' A, 'B',
'computer' ,indian, 5+i3

Categorized

12,2900,3400	444.45, 56..77	A,B	computer ,indian	04/02/2004

Python offers following built in core data types

1. Numbers 2. String 3. List 4. Tuple 5. Dictionary

8.2.1 Numbers	The numbers in Python have following core data types 1. integers i. integers ii. Booleans 2. Floating point number 3. Complex numbers
8.2.1 A Integers	Integer-Ex-12,2900,3400 Boolean-True,False
8.2.1 B Floating point numbers	Floating point numbers Ex-444.45,256.77,233.45,45 5.3 0.53*10^{1} 0.53E01 0.53 Mantissa and E-Exponent EX-3.5E01= 3.5*10^{1}

8.2.1 C Complex numbers	Complex Number
	C=2.5+2.5j
	
	Examples
	a=1.2+4.5j
	print(a)
	print(a.real)
	print(a.imag)
	print(type(a))

	Q1.What will be the data types of the following two variables in python?
	A=223322323
	B=bool(1)
	C=A+1

4.2.2 String	String is a sequence of characters EX- “computer”, ‘343343’, “A” Index of each characters in String S=”python” S[0]=’p’ S[1]=’y’ S[2]=’t’ S[3]=’h’ S[4]=’o’

	S[5]='n'
	
	Examples-Predict the output of the following code
	s="python"
	print(s)
	print(s[0])
	print(s[1])
	print(s[5])
	print(s[-1])
	print(s[-2])
	
	Output
	python
	p
	y
	n
	n
	o

	Examples-Predict the output of the following code
	
	a=10
	b=25.12
	print(a)
	print(b)
	
	Examples-Predict the output of the following code
	c='A'
	d="python"
	k=False
	print(k)
	print(bool(0))
	print(bool(1))
	print(type(a))
	print(type(b))
	print(type(c))
	print(type(a))
	print(type(d))

	<u>OUTPUT</u>
	`10`
	`25.12`
	`False`
	`False`
	`True`
	`<class 'int'>`
	`<class 'float'>`
	`<class 'str'>`
	`<class 'int'>`
	`<class 'str'>`

4.2.3 List and tuple	**The list and tuples are Python compound data types. List can be changed/ modified(mutable) but tuples cannot be changed or modified (immutable)** **Ex Make a list of marks** **Make a list name** ● **To represents group of information**

List	**A list in Python represent a list of comma separated values of any data types between square brackets, following are some list examples**
	
	List=[12,34,45,56]
	print(List)
	Examples 1
	temp=['one' ,24.5,34.2,45.5,42.4]
	
	Examples 2
	List=[12,34,45,56,23.6]
	print(List)
	print(List[0])
	print(List[1])
	print(List[2])
	print(List[4])
	<u>**OUTPUT**</u>
	`[12, 34, 45, 56, 23.6]`
	`12`
	`34`
	`45`

	`23.6`
	Q.Create a list Total Marks which store the marks of 05 students.
	
	ANS TotalMarks=[209,210,450,345,500]
	List can be changed and modified,mutable
	Ex. List=[12,45,56,27]
	print(List)
	List[1]=90
	print(List[0])
	print(List[1])
	print(List)
	
	OUTPUT
	`12`
	`90`
	
	List=[12,34,45,56,23.6]
	print(List)
	print(List[0])
	print(List[1])
	print(List[2])
	print(List[4])
	
	Output
	`[12, 34, 45, 56, 23.6]`
	`12`
	`34`
	`45`
	`23.6`
	
	Ex-
	
	`List=[12,45,56,27]`
	`print(List[0])`
	`print(List[2]*4)`

Tuple	**Tuples are represented as group of comma separated values of any data type within parenthesis ,following are some couples example**

	Tuples values are immutable i.e not changeable
	
	T2=(20,45,67,89)
	print(T2)
	
	T2=(20,45,67,89)
	print(T2)
	
	print(T2[1])
	
	T3=('a','b','c')
	print(T3[0])

	Updation is not allowed in Tuple
	t2=(11,12,13,14,15)
	print(t2)
	t2[2]=90
	print(t2)

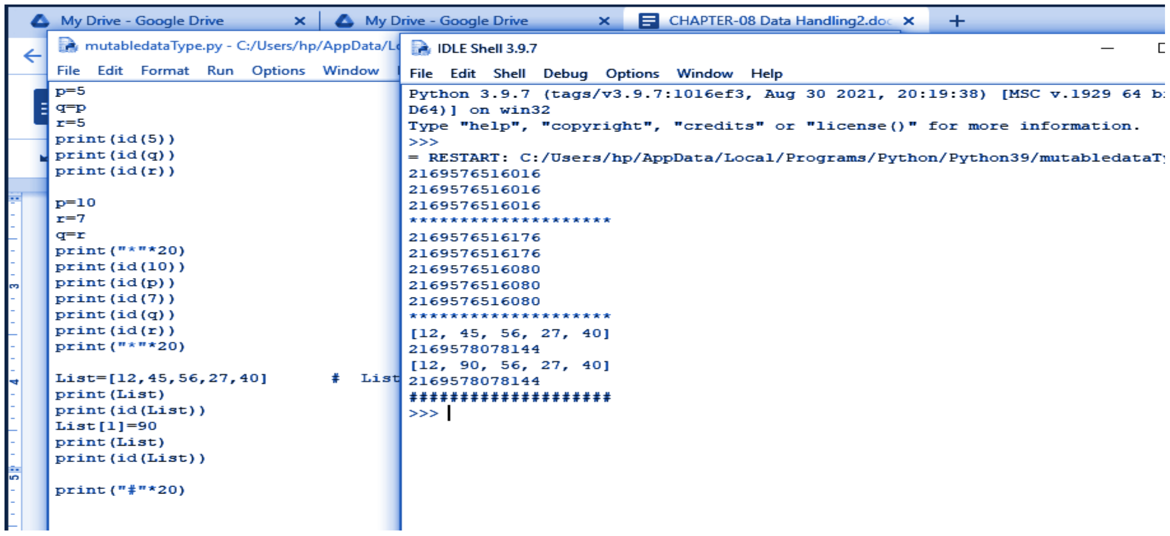
Dictionary	dic={1 : ”india”, 2 : ”USA”, 3: ”England”}
	print(dic[1])
	
	The dictionary is an unordered set of, separated keys value pairs within { } curly braces with the requirement that within a dictionary .No two key can be the same.
	 for instance following are some dictionary example

4.3 Mutable and Immutable data type	● Immutable data types are those that can never change their values in place ,the Following types of mutable : integer,float,Boolean,string,tuple Values are not changing “ in place”-immutable
P=10 p-Variable name or Memory location name 10-Value	5,6,7,8,9,10-Values

5	6	7	8	9	10
1021	1022	1023	1024	1025	1026

P-10122-memory address
P=20
10122-20
	Memory address-1011,1022,1023……1026
	p=5
	q=p
	r=5
	
	p=10
	r=7
	q=r

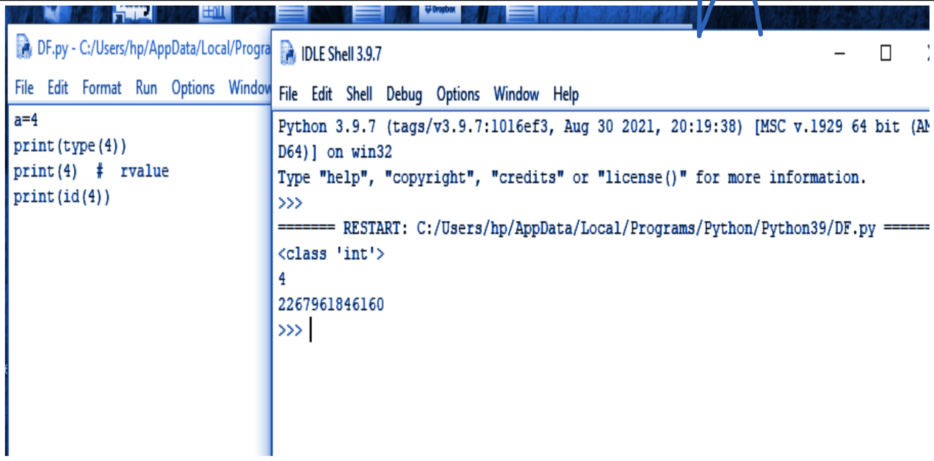
Mutable data type	● List,dictionary
	
	List can be changed and modified,mutable
	List=[12,45,56,27,40] # List[0]=12, List[1]=45,List[2]=56 ……………..
	print(List)
	List[1]=90
	print(List[0])
	print(List[1])
	print(List)

	
Variable Internals	An object is an entity that has certain properties and that exhibit a certain type of behaviour i.e integer variables are object they holds whole number only and they have infinite precision(properties)they support all arithmetic operations (Behaviour) Ex Student (Object)- RN(Numeric) ,Name(Characters) ,age(Numeric)-Data RN,Name,age-properties ,Display Result ,FeeShow

```
p=5
q=p
r=5
print(id(5))
print(id(q))
print(id(r))

p=10
r=7
q=r
print("*"*20)
print(id(10))
print(id(p))
print(id(7))
print(id(q))
print(id(r))
print("*"*20)

List=[12,45,56,27,40]        #  List
print(List)
print(id(List))
List[1]=90
print(List)
print(id(List))

print("#"*20)
```

```
Python 3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021, 20:19:38) [MSC v.1929 64 b
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/hp/AppData/Local/Programs/Python/Python39/mutabledataT
2169576516016
2169576516016
2169576516016
********************
2169576516176
2169576516176
2169576516080
2169576516080
2169576516080
********************
[12, 45, 56, 27, 40]
2169578078144
[12, 90, 56, 27, 40]
2169578078144
####################
>>> 
```

	Every python object has 03 key attributes ● **Data type** ● **Value -real value** ● **Memory address-location value** **A=10** **print(A)** **10**

	
	DF.py - C:/Users/hp/AppData/Local/Progra
	`a=4`
	`print(type(4))`
	`print(4)  # rvalue`
	`print(id(4))`
	
	IDLE Shell 3.9.7
	`Python 3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AM`
	`D64)] on win32`
	`Type "help", "copyright", "credits" or "license()" for more information.`
	`>>>`
	`======= RESTART: C:/Users/hp/AppData/Local/Programs/Python/Python39/DF.py ======`
	`<class 'int'>`
	`4`
	`2267961846160`
	`>>>`

8.4Operator	**Operator are the symbol through which we can perform any operation** **Ex=+ ,- ,*,/ ,%** **Arithmetic Operator**
Arithmetic Operator	**Unary- +a, -a, ~a** **Binary- +,- , * , / , // , %(Modulus operator)** **a+b, a-b, a/c, a%b** **Exponentiation operator -** 2**7**
Assignment operator	**= (Assignment operator)** **Ex. a=80** **p=40** **n=90** **a+=10 #a=a+10** **p -=5 #p=p-5** **n *=10 # n=n*10**

	print(a)
	print(p)
	print(n)
	OUTPUT
	
	`90`
	`35`
	`900`

Relational operator	,<, >=,<= ,**==** ,<> A >b a< b , a==b A==B A< >B !=

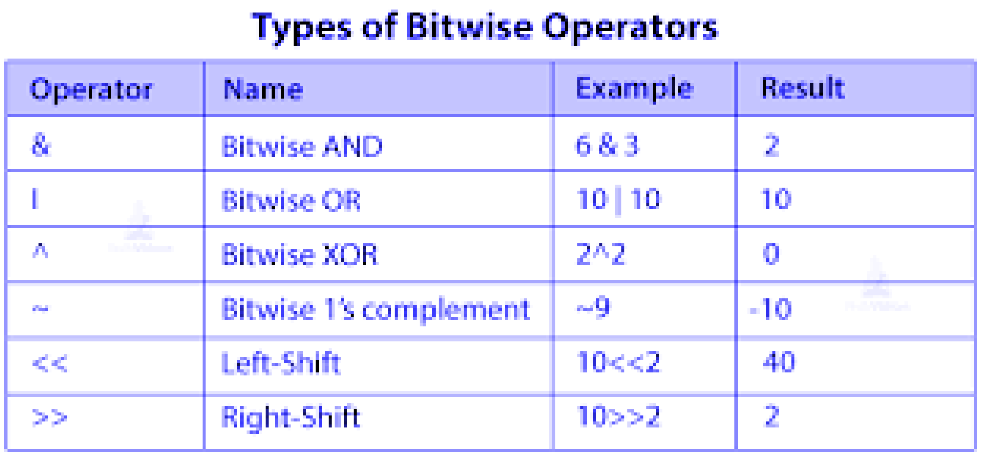
Logical operator	**logical operators refers to the way these relationship can be connected. Python provides three logical operator to combine existing expression .**
	
	These are or ,and, not.
	Ex
	
	a=100
	b=200
	c=5
	print(a>b and a>c)
	print(a>b or a>c)
	
	OUTPUT
	False
	True
	Q Write the output of the following code
	a=10
	b=20
	c=30
	print(a<b and a<c)

	# True
	print(a<b<c) # True
	print(10<40>20) # True
	
	# write the output of the following code
	 a=125 b=240
	if a>b:
	   print(a)
	else :
	    print(b)

Identity operators
(is , is not
	Ex
	a=90
	b=90
	c=100
	print(a is b) # True
	print(a is not c) # True
	
	a=190
	b=90
	c=100
	print(a is b)
	print(a is not c)

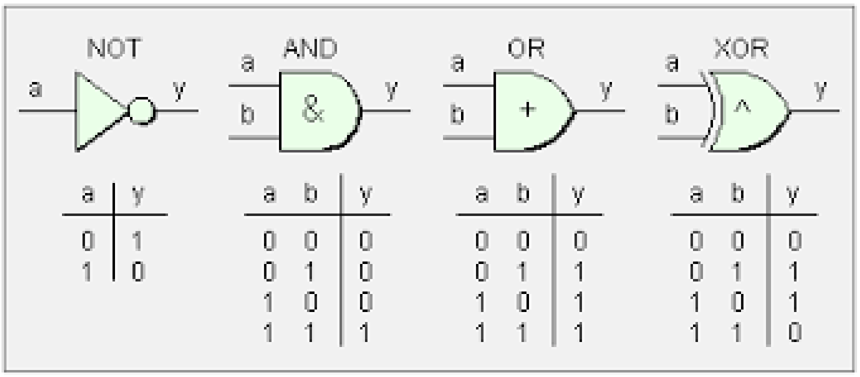
Logical operator	AND .OR NOT
Bitwise Operator	**In Python, bitwise operators are used to performing bitwise calculations on integers. The integers are first converted into binary and then operations are performed on bit by bit, hence the name bitwise operators. Then the result is returned in decimal format.** **Note: Python bitwise operators work only on integers.**

Bitwise operator &,|,^,~



Types of Bitwise Operators

Operator	Name	Example	Result
&	Bitwise AND	6 & 3	2
|	Bitwise OR	10 | 10	10
^	Bitwise XOR	2^2	0
~	Bitwise 1's complement	~9	-10
<<	Left-Shift	10<<2	40
>>	Right-Shift	10>>2	2



a	y
0	1
1	0

a	b	y
0	0	0
0	1	0
1	0	0
1	1	1

a	b	y
0	0	0
0	1	1
1	0	1
1	1	1

a	b	y
0	0	0
0	1	1
1	0	1
1	1	0

Bitwise AND operator: Returns 1 if both the bits are 1 else 0.

Example:

```
a = 10 = 1010 (Binary)
b = 4 =  0100 (Binary)

a & b = 1010
         &
        0100
      = 0000
      = 0 (Decimal)
```

In Python, `in` and `not in` are membership operators used to test for the presence or absence of a value within a sequence or collection. These operators return a boolean value (`True` or `False`).

1. The `in` Operator:

- The `in` operator checks if a specified value is present within a sequence (such as a string, list, tuple, set, or the keys of a dictionary).
- It returns `True` if the value is found in the sequence, and `False` otherwise.

```
my_list = [10, 20, 30, 40]
print(20 in my_list)   # Output: True
print(50 in my_list)   # Output: False


my_string = "hello world"
print("world" in my_string) # Output: True
print("python" in my_string) # Output: False
```

Bitwise or operator: Returns 1 if either of the bit is 1 else 0.

Example:

```
a = 10 = 1010 (Binary)
b = 4 =  0100 (Binary)

a | b = 1010
         |
        0100
      = 1110
      = 14 (Decimal)
```

Bitwise not operator: Returns one's complement of the number.

Example:

```
a = 10 = 1010 (Binary)

~a = ~1010
   = -(1010 + 1)
   = -(1011)
   = -11 (Decimal)
```

Bitwise xor operator: Returns 1 if one of the bits is 1 and the other is 0 else returns false.

Example:

```
a = 10 = 1010 (Binary)
b = 4 =  0100 (Binary)

a & b = 1010
         ^
        0100
      = 1110
      = 14 (Decimal)
```

Operator precedence

```
# Multiplication has higher precedence
# than subtraction
>>> 10 - 4 * 2
2

# Parentheses () has higher precedence
>>> (10 - 4) * 2
12
```

The operator precedence in Python is listed in the following table. It is in descending order (upper group has higher precedence than the lower ones).

Operators	Meaning
`()`	**Parentheses**
`**`	**Exponent**

	`+x, -x, ~x`	Unary plus, Unary minus, Bitwise NOT
	`*, /, //, %`	Multiplication, Division, Floor division, Modulus
	`+, -`	Addition, Subtraction
	`<<, >>`	Bitwise shift operators
	`&`	Bitwise AND
	`^`	Bitwise XOR
	`|`	Bitwise OR
	`==, !=, >, >=, <, <=, is, is not, in, not in`	Comparisons, Identity, Membership operators
	`not`	Logical NOT
	`and`	Logical AND
	`or`	Logical

Operator Associativity	**We can see in the above table that more than one operator exists in the same group. These operators have the same precedence.** **When two operators have the same precedence, associativity helps to determine the order of operations.** **Associativity is the order in which an expression is evaluated that has multiple operators of the same precedence. Almost all the operators have left-to-right associativity.**

	For example, multiplication and floor division have the same precedence. Hence, if both of them are present in an expression, the left one is evaluated first.
	
	`# Left-right associativity`
	`# Output: 3`
	`print(5 * 2 // 3)`
	
	`# Shows left-right associativity`
	`# Output: 0`
	`print(5 * (2 // 3))`

	Note: Exponent operator `**` has right-to-left associativity in Python.
	`# Shows the right-left associativity of **`
	`# Output: 512, Since 2**(3**2) = 2**9`
	`print(2 ** 3 ** 2)`
	
	`# If 2 needs to be exponated fisrt, need to use ()`
	`# Output: 64`
	`print((2 ** 3) ** 2)`

	print(7 * 8/5 // 2) 7*8/5//2 =56/5//2

	=11.2//2
	=5.0
	print((7*8) / 5) // 2)
	=(56/5)//2
	=(11.2)//2
	=5.0
	print(2**3**2)
	2^9
	=512

4.5 Expression	**An expression is a combination of operators and operands that is interpreted to produce some other value. In any programming language, an expression is evaluated as per the precedence of its operators. So that if there is more than one operator in an expression, their precedence decides which operation will be performed first. We have many different types of expressions in Python**
-	**X=4*5+Y** **X,Y- Variable** **4,5-integer literals ,** **+,*,= - operator**
Types of Expression-	1.**Arithmetic Expression-** An arithmetic expression is a combination of numeric values, operators, and sometimes parenthesis. The result of this type of expression is also a numeric value. The operators used in these expressions are arithmetic operators like addition, subtraction, etc. Here are some arithmetic operators in Python: Ex 5+3 , a+b , 67.7+34.2 2.**Relational Expression**- a>b, a>=b, a<b<c In these types of expressions, arithmetic expressions are written on both sides of relational operator (> , < , >= , <=). Those arithmetic expressions are evaluated first, and then compared as per relational operator and produce a boolean output in the end. a>b ,a>=b

3.Logical expression- a>b and a>c, a>b or a>c

These are kinds of expressions that result in either *True* or *False.* It basically specifies one or more conditions. For example, (10 == 9) is a condition if 10 is equal to 9.

4.String Expression = “python” * 3 , ”Computer ” +” Basic” ,” 12 ”+”3”

```
print("Python"  *5)
p="india" * 4
print(p)
print("compter"+"application")
print("12"+"13")
```

```
PythonPythonPythonPythonPython
indiaindiaindiaindia
compterapplication
1213
```

4.5.1 Evaluating expression

1. Evaluating Arithmetic expression- use implicit conversion or type promotion or Coercion

- In Implicit type conversion, Python automatically converts one data type to another data type. This process doesn't need any user involvement.
- Implicit Type Conversion is automatically performed by the Python interpreter.
- Python avoids the loss of data in Implicit Type Conversion.
- Int-int-int
- Int-float-float
- Float-float-float

```
ch=5
i=2
f=4
db=5.0
A=(ch+i)/db     #  Expression
B=f/db*ch/2      # Expression
print(A)
print(B)
```

A= (int +int)/float

A= int/float   7/5.0 =1.4 -- float

Int/float *int/2

Ex. Q.What will be the resultant data type of the following expression

```
a=10
b=2
c=2.5
d=a*b+c
print(d)
print(type(d))
```

Ans

22.5

Float

Q.

```
a=10.5
b=2
c=2.5
d=a*b+a/c
print(d)
print(type(d))
```

Ans -25.2

Float

Note- In Python ,if the operator is division(/) ,the result will always be a floating point number even if both operand are integer type

Ex.

```
a,b=3,6
c=b/a
print(type(c))
print(c)
d=b//a
print(type(d))
```

2.Evaluating Relational expression- True and false

- All relational expression return True and False Value

Ex print(10<15)

Ex print(10<12<15)

Q
print(12<15<5)
Ans False

3.Evaluating logical expression-

- All logical expression return True and False Value

Ex print((3<5) or (5<2)) -----True
Ex print((5<3) or (5<2)) ……… False

```
a,b,c=10,200,30
print(a<b and a<c)
print(a<b or a<c)
```

Ans-True True

```
a,b,c=10,20,30
print(a<b and a<c)
```

Ans-True

Q1.What is the difference between implicit and explicit conversion?

Explicit Type Conversion (type casting)

- In Explicit Type Conversion, users convert the data type of an object to required data type.
- We use the predefined functions like int(), float(), str(), etc to perform explicit type conversion.
- This type of conversion is also called type casting because the user casts (changes) the data type of the objects.

Syntax :

```
<required_datatype>(expression)
```

- Typecasting can be done by assigning the required data type function to the expression.

	● Explicit Type Conversion is also called Type Casting, the data types of objects are converted using predefined functions by the user. ● In Type Casting, loss of data may occur as we enforce the object to a specific data type.

4.6 Working with math Module of Python	**A Python module is a file containing Python definitions and statements.**
	
	A module can define functions, classes, and variables.
	
	A module can also include runnable code. Grouping related code into a module makes the code easier to understand and use. It also makes the code logically organized.
	
	We can import the functions, classes defined in a module to another module using the import statement in some other Python source file.
	Syntax:
	
	Creation a module
	`# A simple module, calc.py`
	
	`def add(x, y):`
	`    return (x+y)`
	
	`def subtract(x, y):`
	`    return (x-y)`
	`==========================================================`
	`import module`
	`import calc`
	`print(calc.add(10, 2))`
	
	`==========================================================`
	import math
	a=23

```
print(math.sqrt(a))
print(math.log(a))
print(math.pow(9,2))
print(math.sin(90))
```

random module

This modules provides random-numbers generators.
random-numbers generators functions in random modules are
1.random()
2.randint(a,b)
3.randrang(start.stop,step)

Example

```
import random
print(random.random())  # generate the random no between 0.0<=n and <1.0
print(random.random()*(30-20)+20)
print(random.randint(15,35))
print(random.randint(3,10)-3)
```

OUTPUT

```
0.8459129190851102
10.681772703080899
35
3
```

Expression for mathematical expression

Mathematical Expression	Python Expression
Ab	a*b
a÷b	a/b
√ $a^2+b^2+c^2$	math.sqrt(a*a+b*b+c*c)
$2-ye^{2y}+4y$	2-y*math.exp(2*y)+4*y
P+ Q/$(r+s)^4$	P+Q/math.pow((r+s),4)
| e^2-x |	math.abs(math.exp(2)-x)
□r^2	3.14*r*r or math.pi
sin(x)+cos(x)	Math.sin(x)+math.cos(x)
$(a+b)^4$	Math.pow((a+b),4)
$(a+b)^4+2y+3y^2$	

4.7 Debugging	Debugging refers to the process of locating the place of error ,cause of error and ,correcting the code accordingly

4.7.1 Errors in program	• Compile time error
	
	1.Syntax Error 2.Semantics Error
	
	• Logical Error
	
	• Run time Errors(Exceptions)
	
	1. Syntax error
	
	Print(a)
	Input, itn “
	
	Semantics errors
	a=2+3
	b=5
	a+b=c
	print(c)
	
	2.Logical error
	
	C=a-b
	C=a+b
	
	3.Run time error
	
	a=5
	b=0
	c=a/b
	print(c)
	
	`ZeroDivisionError Traceback (most recent call last)`

```
<ipython-input-30-58b8ea25038d> in <module>
      1 a=5
      2 b=0
----> 3 c=a/b
      4 print(c)

ZeroDivisionError: division by zero
```

```
l=[12,24,56]
print(l[3])
```

```
IndexError                                Traceback
(most recent call last)
<ipython-input-31-29b8f950866a> in <module>
      1 l=[12,24,56]
----> 2 print(l[3])

IndexError: list index out of range
```

```
import math
x = int(input('Please enter a positive number:\n'))

try:
    print( math.sqrt(x))
except ValueError as ve:
    print("Please enter positive number.")
```

Exercise Questions

	Q1.What will be the data types of following two variable in python?
	A=223322323
	B=bool(1)
	
	A = int B = true Boolean

	Q.Write the output of the following code . List=[12,45,56,27] print(List[0]) print(List[2]*4) Ans - 12 224
	Q..Write the output of the following code p=5 q=p r=5 p=10 r=7 q=r print(p,q,r) Ans 10 7 7
	What is mutable and immutable data type? Ans-Mutable: Data Which we can modify and can change its location Immutable: Data Which we cannot modify and cannot be change its location

	Q. Write the output of the following code
	     a=10
	      b=3
	      c=a+b
	       d=a-b
	       e=a/b

	f=a//b
	k=a%b
	print(c)
	print(d)
	print(e)
	print(f)
	print(k)
	
	ans- 13
	7
	3.3
	3

	Write the output of the following code 1.17%5 output-2 2. 17*5.0 3.(17%5)==(17%5)
	Q.Write the python expression for Area=√ s*(s-a)(s-b)(s-c)
	Write a code to repeat the string “Good morning” n times .Here n is input by the user.
	Write a programme to find compound interest
	Write the output of the following code 1.17%5 2. 17*5.0 3.(17%5)==(17%5) 2.17%5.0 2 2.0 2.0