AKTU

B.Tech IV-SEM All Branches

GATEWAY CLASSES

ONE SHOT REVISION

PYTHON PROGRAMMING

By Pragya Ma'am

Unit-1

Introduction to Python

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.

Write a program of swapping of two number

using third variable

main.py

```
x=int(input("enter first number\n"))
y=int(input("enter second number\n"))
print("the value of x=",x)
print("the value of y=",y)
z=x
x=y
y=z
print("the value of x after swapping", x)
print("the value of y after swapping", y)
```

```
enter first number
5
enter second number
7
the value of x= 5
the value of y= 7
the value of x after swapping 7
the value of y after swapping 5
```

without using third variable

main.py

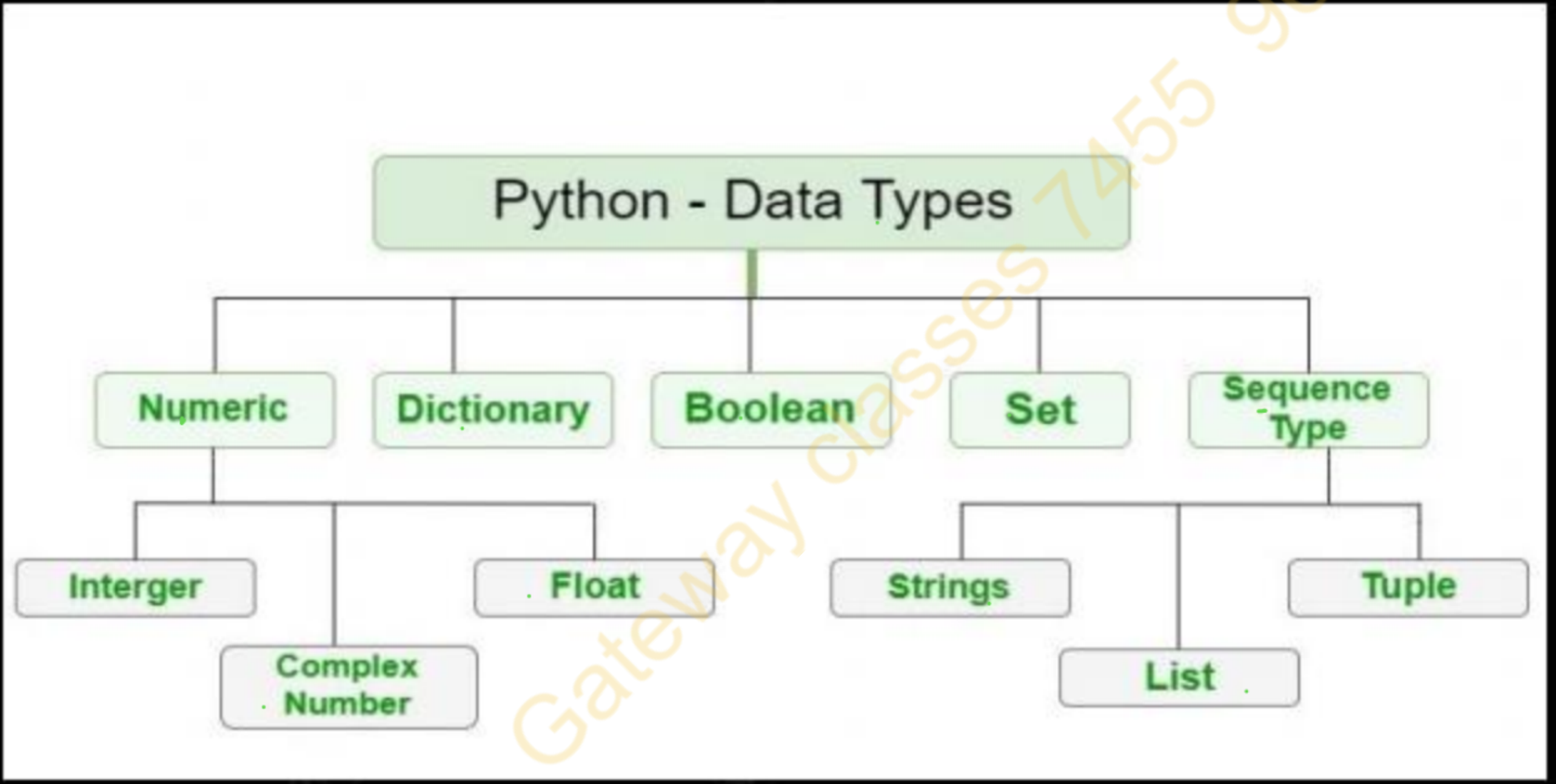
```
x=int(input("enter first number\n"))
y=int(input("enter second number\n"))
print("the value of x=",x)
print("the value of y=",y)
x=x + y
y=x - y
x=x - y
print("the value of x after swapping", x)
print("the value of y after swapping", y)
```

```
enter first number
6
enter second number
7
the value of x= 6
the value of y= 7
the value of x after swapping 7
the value of y after swapping 6


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

DATA TYPE

- Python Data types are the classification or categorization of data items. It represents the kind of value that tells what operations can be performed on a particular data



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.

main.py

```
num1=5
print(num1,"is type",type(num1))
num2=2.0
print(num2,"is of type",type(num2))
num3=1+2j
print(num3,'is of type',type(num3))
```

```
5 is type <class 'int'>
2.0 is of type <class 'float'>
(1+2j) is of type <class 'complex'>
```

SEQUENCING TYPE String

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

main.py

```
str1="hi"
str2='bye'
str3='''hello'''
print(str1,"is of type",type(str1))
print(str2,"is of type ",type(str2))
print(str3,"is of type",type(str3))
```

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

main.py

```
1  str1="hello gatewayclasses"
2  str2="welcome"
3  print(str1[0:2])
4  print(str2[1:3])
5  print(str1[4])
6  print(str2[5])
7  print(str1*2)
8  print(str2*4)
9  print(str1+str2)
10 print(str2+str1)
```

```
he
el
o
m
hello gatewayclasseshello gatewayclasses
welcomewelcomewelcomewelcome
hello gatewayclasseswelcome
welcomehello gatewayclasses
```

LIST

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

main.py

```
1 language=["python","c","java"]#SAME TYPE OF ELEMENT
2 print(language) # PRINTLIST
3 print(language[0])# ACCESS THE ELEMENT
4 print(language[1])
5 print(type(language))
6 language1=["python",1,"c","java",3.5]
7 print(language1)# DIFFRENT TYPE OF ELEMENT
```

```
['python', 'c', 'java']
python
c
<class 'list'>
['python', 1, 'c', 'java', 3.5]
```

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.

main.py

```
1  product=('microsoft','xbox',499.99)
2  print(product)
3  print(type(product))
4  print(product[1])
5  tup = ("hi", "Python", 2)
6  print (tup[1:])#SLICING
7  print (tup[0:1])
8  print (tup + tup) #CONCATENATION
9  print (tup * 3) # REPETITION
10 tup[2] = "hi" # It will throw an error.
11
```

```
('microsoft', 'xbox', 499.99)
<class 'tuple'>
xbox
('Python', 2)
('hi',)
('hi', 'Python', 2, 'hi', 'Python', 2)
('hi', 'Python', 2, 'hi', 'Python', 2, 'hi', 'Python', 2)
Traceback (most recent call last):
  File "/home/main.py", line 10, in <module>
    tup[2] = "hi" # It will throw an error.
TypeError: 'tuple' object does not support item assignment
```

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

main.py

```
capital_city = {'Nepal': 'Kathmandu', 'Italy': 'Rome', 'England': 'London'}
print(capital_city)
print(capital_city['Nepal'])
print(capital_city.keys())
print(capital_city.values())
print(type(capital_city))
```

input

```
{'Nepal': 'Kathmandu', 'Italy': 'Rome', 'England': 'London'}
Kathmandu
dict_keys(['Nepal', 'Italy', 'England'])
dict_values(['Kathmandu', 'Rome', 'London'])
<class 'dict'>
```

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 type.

```
1 a = True
2 print(a)
3 print(type(a))
4
```

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

SETS

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

```
1 a = {"apple", "banana", "cherry", "apple"}
2 print(a)
3 print(type(a))
4
```

```
{'apple', 'cherry', 'banana'}
<class 'set'>
```

Type conversion in python

AKTU PYQs

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

- **Implicit Type Conversion** (Automatic)
- the Python interpreter automatically converts one data type to another without any user involvement → No data loss

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

```
30.5
```

Explicit Type 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.

main.py

```
1  s="10010"
2  c=int(s,2)
3  print(c)
4  e=float(c)
5  print(e)
6
```

```
18
18.0
```

	Global Variable	Local Variable
Definition	➢ declared outside the functions	➢ declared within the functions
Data Sharing	➢ Offers Data Sharing	➢ It doesn't offers Data Sharing
Scope	➢ Can be access throughout the code	➢ Can access only inside the function
Lifetime	➢ They are created 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

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

```
1 a= int(input("enter the number"))
2
```

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
- Python is known for its simplicity and readability, making it an excellent choice for both beginners and experienced programmers.

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
```

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
- Membership operators

- Special operators

1. Identity operators
2. Membership operators

1Arithmetic 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) 7/3 = 2.3333	It divides two numbers and rounds down to the nearest integer. result = 7 // 3 print(result) # Output: 2

2 Comparison operators (relational operator)

main.py

```
1  x=10
2  y=12
3  print(x==y)#equal to
4  print(x>y) #greater than
5  print(x<y)#less than
6  print(x>=y)#greater than equal to
7  print(x<=y)#less than equal to
8  print(x!=y)#not equal to
9
```

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

Assignment operators

main.py

```
1  a=10# assignment
2  print(a)
3  a+=10#a=a+10
4  print("a=",a)
5  a*=10
6  print("a=",a)
7  b=5
8  b/=2
9  print("b=",b)
10 b//=2
11 print("b=",b)
12 b%=2
13 print("b=",b)
14 z=6
15 z**=2
16 print("z=",z)
17
```

```
10
a= 20
a= 200
b= 2.5
b= 1.0
b= 1.0
z= 36
```

logical Operators

- It is used to make decision based on condition

- And - Logical AND
- Or - Logical OR
- Not - Logical not

main.py

```
1 x = 10>2
2 y = 20<4
3 print('x and y is',x and y)
4 print('x or y is',x or y)
5 print('not(x and y) is',not(x and y))
6
```

```
x and y is False
x or y is True
not(x and y) is True
```

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

x	y	x & y	x | y	X^Y	~(x)	~y)
0(false)	0	0	0	0	1	1
0	1	0	1	1	1	0
1(true)	0	0	1	1	0	1
1	1	1	1	0	0	0

	2^7 128	2^6 64	2^5 32	2^4 16	2^3 8	2^2 4	2^1 2	2^0 1
10	0	0	0	0	1	0	1	0
4	0	0	0	0	0	1	0	0
10&4	0	0	0	0	0	0	0	0

	2^7 128	2^6 64	2^5 32	2^4 16	2^3 8	2^2 4	2^1 2	2^0 1
10	0	0	0	0	1	0	1	0
4	0	0	0	0	0	1	0	0
10|4	0	0	0	0	1	1	1	0

	2^7 128	2^6 64	2^5 32	2^4 16	2^3 8	2^2 4	2^1 2	2^0 1
10	0	0	0	0	1	0	1	0
6	0	0	0	0	0	1	1	0
10^6	0	0	0	0	1	1	0	0

	2^7 128	2^6 64	2^5 32	2^4 16	2^3 8	2^2 4	2^1 2	2^0 1
10	0	0	0	0	1	0	1	0
~10	1	1	1	1	0	1	0	1

10	0	0	0	0	1	0	1	0
10<<1	0	0	0	1	0	1	0	0(extra bit added)

Lsr

$X >> n$

$X * 2^n$ — (No of bit you want to Shift)

$10 << 3$

$10 * 2^3$

$10 * 8 = 80$

	2^7 128	2^6 64	2^5 32	2^4 16	2^3 8	2^2 4	2^1 2	2^0 1
10	0	0	0	0	1	0	1	0
10>>1	0(extra bit added)	0	0	0	0	1	0	1

Lost

$X / 2^n$

$10 / 2^1$

5

membership Operators

- 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 list=[1,2,3,4,5]
2 b=2
3 if b in list:
4     print("present")
5 else:
6     print("not present")
7
8
```

```
present
```

```
main.py
1 list=[1,2,3,4,5]
2 b=2
3 if b not in list:
4     print("present")
5 else:
6     print("not present")
7
8
```

```
not present
```

membership Operators

- 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 list=[1,2,3,4,5]
2 b=2
3 if b in list:
4     print("present")
5 else:
6     print("not present")
7
8
```

```
present
```

```
main.py
1 list=[1,2,3,4,5]
2 b=2
3 if b not in list:
4     print("present")
5 else:
6     print("[redacted] present")
7
8
```

```
[redacted] present
```

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. (is, is not)

```
1 x=5
2 y=5
3 print("x id",id(x))
4 print("y id",id(y))
5 print("x type",type(x))
6 print("y type",type(y))
7 print(x is y)
```

```
x id 130592050970992
y id 130592050970992
x type <class 'int'>
y type <class 'int'>
True
```

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

20 * 3 + 5
60 + 5
65

20 * 3 * 2
60 * 2
120

Identity operator

- num1 = 5
  num2 = 5
  list1 = [1,2,3]
  list2 = [1,2,3]
  list3 = list1
  print(num1 is num2) — true
  print(list1 is list2) — False
  print(num1 is not num2) False
  print(list1 is not list2) — true

is operator	Return true if both objects refers to same memory location else return false
is not Operator	Return false if both refers to same memory location else return false.

Python Operator Precedence

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

(2>3) and (4>3)
F and T
F

Operators	Description	Associativity
()	Parentheses	Left to right
x[index], x[index : index]	Subscription, slicing	Left to right
await x	Await expression	N/A
**	Exponentiation	Right to left
+x, -x, ~x	Positive, negative, bitwise NOT	Right to left
*, @, /, //, %	Multiplication, matrix, division, floor division, remainder	Left to right
+, –	Addition and subtraction	Left to right

Operators	Description	Associativity
<<, >>	Shifts	Left to right
&	Bitwise AND	Left to right
^	Bitwise XOR	Left to right
|	Bitwise OR	Left to right
in, not in, is, is not, <, <=, >, >=, !=, ==	Comparisons, membership tests, identity tests	Left to Right
not x	Boolean NOT	Right to left
and	Boolean AND	Left to right

Operators	Description	Associativity
or	Boolean OR	Left to right
if-else	Conditional expression	Right to left
lambda	Lambda expression	N/A
:=	Assignment expression (walrus operator)	Right to left

AKTU PYQs

- Solve the following question step by step
- 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 5*2^10=5*1024=5120
- 3&5120^5
- 0^5 =5 (note a^0=a a^1= a')

AKTU PYQs

- Solve the following question step by step
- 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>>25^100
- 0^1000
- 1000

Thank You