Python Programming

Presented by

Mariena A. A.

Asst. Professor and Head, Department of Computer Science

Little Flower College, Guruvayoor.

Numeric data types

Numeric value can be **integer, floating number or even complex numbers**. These values are defined as int, float and complex class in Python.

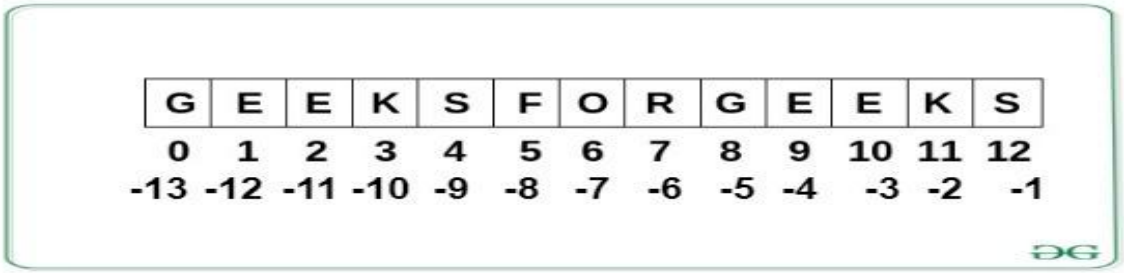
Integers :contain positive and negative numbers without any decimal point. Integer can be represented as decimal, binary, octal and hexadecimal.

Decimal allowed values are 0-9.
Binary allowed values are 0,1(0B/0b1111)
Octal allowed values are 0-7(0O23/0o23)
Hexa allowed values are 0-9,a-f(0X10/0x10)
Float: real number with floating point
Complex: real part+imaginery part
ex: x=4+8j// 4 is real part and 8j is imaginery part(j=√-1)
Print(x. real)/(x.imag)
Real part can be in any of the binary,decimal,octal,hexa form
Boolean : takes two built in values True or False. It is denoted by bool.

String

String: Array of bytes represents Unicode characters.

Collection of characters put in a single quote, double quote or triple quote. There is no char data type single character is a string having length one. Representation of string in python.



Part of the string is slice.

Ex:- s=‘abcdefghij’

Print(s[3:9])// it prints the characters from 3rd index to 8 th index characters

If you want to hold a group of values together then you have to go for collection related data types:

+ operator for concatenation and * operator for repeatation(str1=‘hello’/str1*3=hellohellohello)

List

List:-

List is a comma separated values having various types enclosed in a square bracket.

It is represented using [ ]

Lists are **mutable**, and hence, they can be altered even after their creation.

Order preserved.

Duplicate objects are allowed.

Heterogeneous types are allowed.

The elements in a list are indexed according to a definite sequence and the **indexing** of a list is done with 0 being the first index.

L=[10,20,30,40]

l. append(10)//growable

l. remove(30)

l[0]=777// we can modify the existing values// mutable

Tuple

Tuple is an ordered collection of Python objects much like a list.

The sequence of values stored in a tuple can be of any type, and they are indexed by integers.

Ex:- my_tuple=(‘p’,’q’,’r’,’s’,’t’,’u’)

Print(my_tuple[1:4])

Print(my_tuple[:-3])

Print(my_tuple[3:])

Print(my_tuple[:]

The important difference between a list and a tuple is that tuples are **immutable.** // This means that elements of a tuple cannot be changed once they have been assigned.

We could not append and remove any element from a tuple list.

Tuple with Single value should ends with comma otherwise it will treated as int.

Ex:- t(10,) Ex: print((1,2,3)+(4,5,6))

+ operator is used to concatenate two tuples and * operator is used to repeat the elements in a number of times.

Set and Frozen set

Set is an **unordered** collection of data type that is iterable, **mutable** and has **no duplicate** elements.

The order of elements in a set is undefined though it may consist of various elements.

S1={1,2,3}

S1.add(4)// no guarantee where it is stored

S1.remove(3) // No indexing and slicing performed on set.

S={ } empty curly braces always create dictionary not set.

Frozen set

It is exactly same as set but we cannot change.

Immutable(No addition and removal)

No duplicates are allowed

No indexing, No slicing, Order is not preserved

Ex:-

s={10,20,30}

fr=frozenset(s)

Dictionary

Dictionary

If we want to represent data as key-value pairs then we have to use dictionary.

Each key :value pair is separated by comma.

It can be created by using dict() function.

Syntax: d={k1:v1,k2:v2,k3:v3}

Duplicate keys are not allowed and values can be duplicated//d={name: ‘durga’, age:20}

We can change the value using assignment operator.

d[age]=26// age will replace with 26

To get the values we can use get() method

Print(d. get(name))

Mutable and growable so no indexing and slicing

Range

To represent sequence of numbers

r=range(0,10)

Print(r)

o/p—range(0,10)

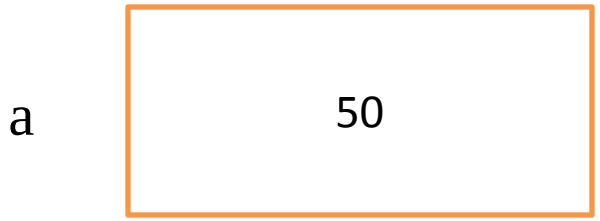
To get the values we have to use for loop

Four General format:

range(n)

range(begin to end-1)// r=range(1,10)o/p= 1-9 numbers

range(begin,end-1,increment/decrement)



If we want remove the pointer then 50 will be removed by garbage collector.
It is not pointed by any variable.
If function not return any value then the output will be **none.**

Conclusion

Properties	List	Tuple	Set	Dict	Frozen set
Representation	[ ]	( )	{ }	{}	{ }
mutable	Yes	No	Yes	Yes	No
Order	Prserved	yes	No	No	Not
Duplicate	Allowed	yes	Not	Only values	Not
Insexing/Slicing	Allowed	allowed	Not	Not	Not

Precedence and Associativity

- There can be more than one operator in an expression.
- To evaluate these types of expressions there is a rule of precedence in Python. It guides the order in which these operations are carried out.
- Ex:- 10-4*2
- We can change the priority by using parenthesis ( )
- Ex:- (10-4)*2

- Python precedence rule is given in the next slide it is in descending order (upper group has higher precedence than the lower ones).

Precedance

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 OR

Associativity

Associativity of Python Operators

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 In left associativity.

```
5*2//3
Print(5*2//3) # left to right associativity output 3
Print(5*2//3) # right to left associativity output is 0
```


Thank you!