Python programming

Features of python

- Easy to learn
- Easy to read
- High level language
- Multithreading, multiprocess
- Garbage collector
- Interpreter-→runs the program line by line. It coverts every line into machine language, and executes in the memory
- GUI -→Graphical user interface→tkinter, QtPy
- Networking -→socket programming
- If you know python then you may design --→ single user, web application(Django,Flask), ML,AI(Scikit-learn)
- Interactive—allows to accept i/p from user
- Python dynamically typed.
  s=34
  s="Kishori"
  s=56.7

  Variable
- Variable name should start with alphabet, can use combination of alphabet and digits
- It cannot contain special character
- Do no use keywords as variable

Structure of python program

1. Semicolon is not mandatory
2. No {} , () begin and end curly braces for loops, functions are not needed , () for condtion declaration is not needed.

Download python from python.org

To check python version

Open cmd prompt

C:\system32>python –version

To open python shell or REPL (Read evaluate print loop>

C:\system32>python

Operators in python

Arithmetic operators

+, -, *, /, %,  //(integer division),  :=(walrus)

a=4

b=3

C=a//b

Ternary operator in python

C=a>b?a:b  #java

C=a if a>b else b  #pythonic

There is no ++ and - - operator

Logical operator

and , or, not

relational operators

>, < , >=, <=, ==, !=

Bitwise opearators

& , | , >>,  <<, ^

Loops in python

2 loops in python

1. For loop
    a. If you know number of iterations in advance
2. While loop
    a. If the number of iterations are not known, then we use while loop

For----else

While----else

Strings in python

Internally the string is list of characters

| T | H | I | S | | I | S | | S | T | R | I | N | G |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| -14 | -13 | -12 | -11 | -10 | -9 | -8 | -7 | -6 | -5 | -4 | -3 | -2 | -1 |

| | | | | | |
|---|---|---|---|---|---|
| To find last character | S[-1] | G | | | |
| To display 2 to 7 characters | S[2:9] | IS IS S | | | |
| To display string from 2 nd index till end | S[2:] | IS IS STRING | S | | I |
| To display string from THE BEGINNING TILL 8 index POSITION | S[:9] | | | | |
| To display all characters at even index position | S[::2] | | | | |
| To display all characters at even index position | S[1::2] | | | | |
| To display string in reverse order | S[::-1] | | | | |

| | |
|---|---|
| String.find(substr,[start,end]) | The position of the first occurrence of the substr, and returns -1 if not found.<br>Start and end are optional, if you specify then searching will be done only in that portion , otherwise it searches the whole string |
| String.rfind(substr, ,[start,end]) | The position of the last occurrence of the substr, and returns -1 if not found<br>Start and end are optional, if you specify then searching will be done only in that portion, otherwise it searches the whole string |
| String.index(substr) | The position of the first occurrence of the substr, and throws exception if not found<br>Start and end are optional, if you specify then searching will be done only in that portion , otherwise it searches the whole string |
| String.rindex(substr) | The position of the last occurrence of the substr, and throws exception if not found<br>Start and end are optional, if you specify then searching will be done only in that portion , otherwise it searches the whole string |
| String.upper() | Convert string in uppercase |
| String.lower() | Convert the string into lowercase |
| String.startswith(“xxx”) | Returns true if string starts with xxx, else it returns false |
| String.endswith(“xxx”) | Returns true if string ends with xxx, else it returns false |
| String.split(“:”) | “abc:pqr:xyz”.split(“:”) , it will break the string into 3 parts, and store it in the list |
| “delimiter”.join(lst) | All the values of lst will be concatenated by delimiter and generate a string |
| String.replace(oldstr,newstr,[count]) | If count is not given the all occurrences of oldstr will be replaced by new str |
| String.count(substr) | It prints the number of occurrence of the substr in the string |
| String.strip(list of characters) | This function will delete all occurrence of characters from left and right side of the string |
| String.lstrip(list of characters) | This function will delete all occurrence of characters from left side of the string |

| | |
|---|---|
| String.rstrip(list of characters) | This function will delete all occurrence of characters from right side of the string |

Basic data types are:
Number
String
Boolean

All basic data types are immutable.

Data structure
1. List
2. Tuple
3. Set
4. Frozenset
5. Dictionary

List:

1. It is dynamically growable and shrinkable

2. heterogeneous data can be stored

3. ordered list, it maintains the order of insertion, it means that we can retrieve the data randomly by using index

1,4,23,10

4. it allows duplicate values.

5. to represent lists we use []

6. lists are mutable

| | |
|---|---|
| List.index(val,[start,end]) | Returns index position of the first occurrence of the number if found, else it throws exception if not found<br>If start and end values are given then it searches in the given range, otherwise search in entire list |
| List.append(data) | It adds one element at the end of the list |
| List.extend([d1,d2,d3]) | It adds multiple elements at the end of the list |
| List.insert(pos,data) | It add data at the given position |
| List.pop([num]) | If num is not given, then by default pop will delete the values from the end of the list, but id=f num is given, then it deletes from the num position |
| List.remove(data) | If deletes the first occurrence of data from the list if exists, otherwise it will generate exception |
| List.reverse() | It reverse the list, it changes the original list |
| List.sort() | If the list contains homogeneous values then only it will sort in ascending order, to arrange it in descending order<br>Lst.sort(reverse=True), it changes the original list |
| List.count(val) | It returns number of occurrences of the given values |
| List.clear() | It removes all the values from the list |

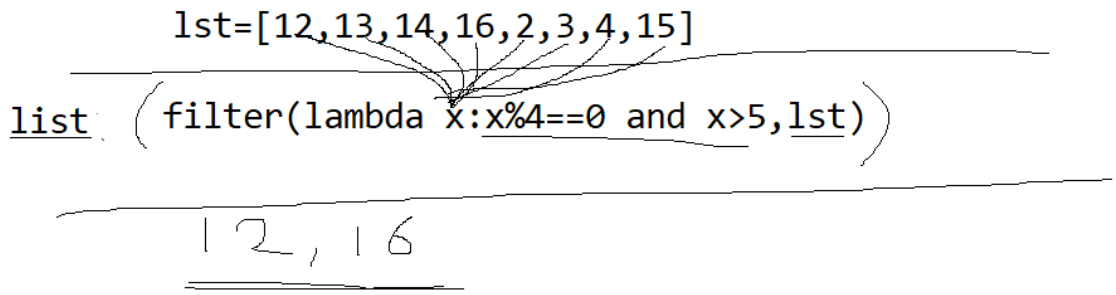
| | |
|---|---|
| List.copy() | It will create a shallow copy of the list |

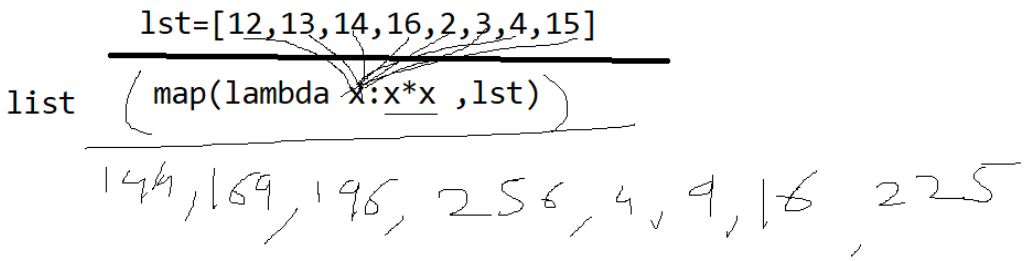
Tuple:

1. Tuples are immutable.
2. Tuples are represented by using ()
3. It is an ordered collection, so data can be retrieved randomly by using index.
4. It allows duplicate values.
5. It allows to store heterogeneous values.
6. Tuples are mostly useful for returning multiple values from a function.
7. It is also used to pass variable number of parameters to the function.

| | |
|---|---|
| Tuple.Count(value) | Returns the number of occurrences of the given value in the tuple |
| Tuple.index(value) | Returns the position of the first occurrence of the value in the given tuple |

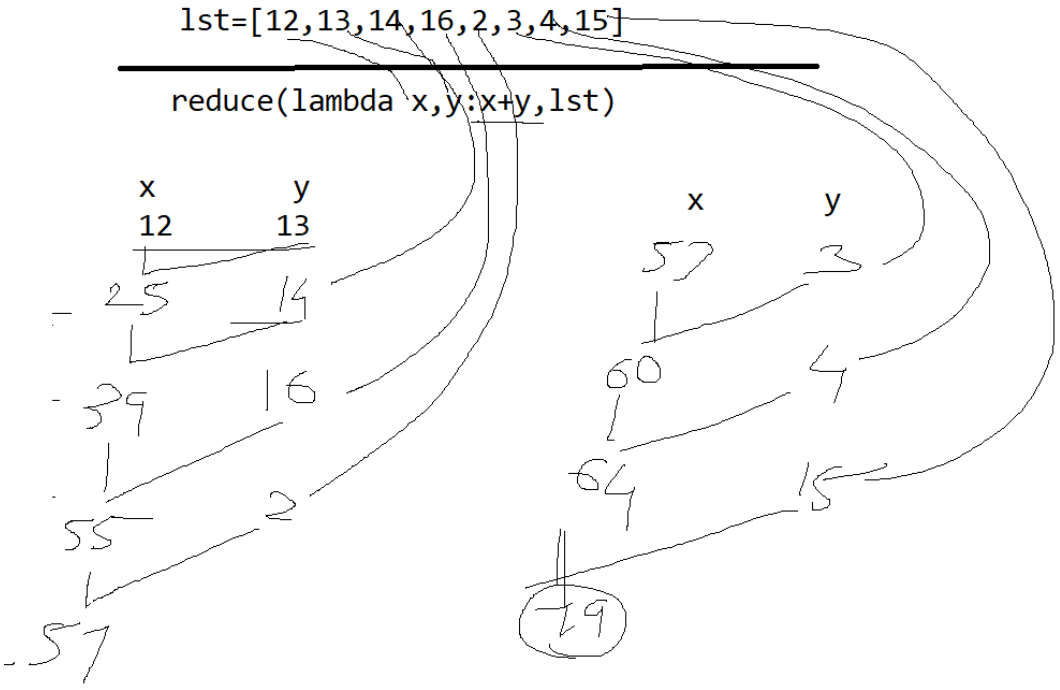
Filter



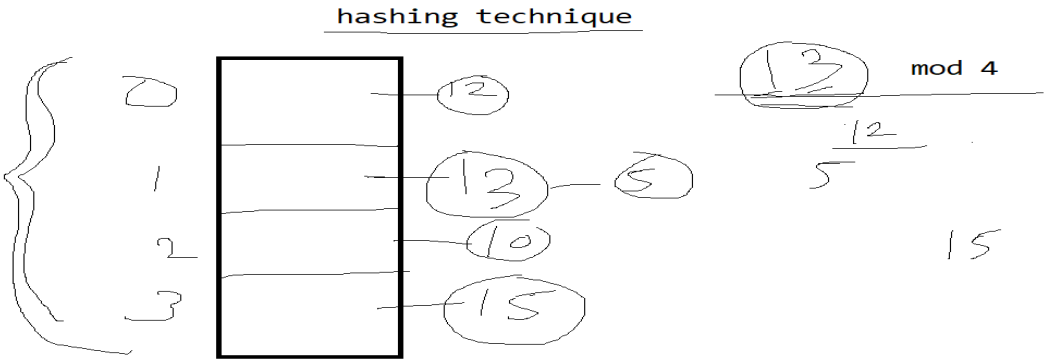

| | |
|---|---|
| Zip(lst1,lst2,…) | To navigate through multiple iterables simulteneously |
| Enumerate(lst,[start]) | It numbers the list, numbering starts with 0, hence can be considered as index of the list, if the value for start is given then the numbering will start from the given numb er |
| sorted(lst,[reverse=True] | It will send the data to for loop in the sorted order, but the original list will not be changed, if reverse=true is given, then the list will be sorted in descending order. |
| reversed(lst) | It sends the data in the for loop in the reverse order, it doesnot change the original list. |
| filter(function,lst) | It will apply given function on each value of the list and it will accumulate only values for which function returns true |
| Map(function,lst) | apply given function on each value of the list and it will accumulate all the values returned from the function, (one for each i.p value) |

| | |
|---|---|
| reduce(function,lst,[initial value]) | apply given function on each value of the list, and reduce the values in the list to a single value |

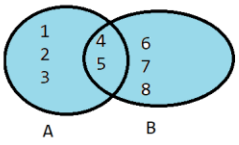



Sets

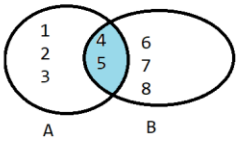
1. It allows to store unique.
2. Sets are represented by using {}.
3. Empty set cannot be created by using {}, but you may create it as
   S=set()
4. It is unordered list of values; it will not be able to access randomly by using index.
5. It is mutable data structure.
6. It allows to store only immutable objects.



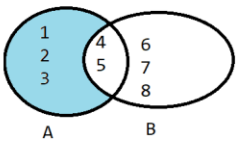
| | |
|---|---|
| Set.add(val) | It adds the value into set, based on hash function |
| Set.update(iterable) | To add multiple values in the set |
| Set.pop() | To delete the value randomly |
| Set.remove(val) | To delete the given value if exists, otherwise, it throws exception |
| Set.discard(val) | To delete the given value if exists, otherwise, ignores |
| S1.union(s2) s1\|s2 | To find all the values from s1,s2 and common values |
| S1.intersection(s2)<br>S1&s2 | To find all common values from the sets |
| S1.difference(s2)<br>S1-s2 | To find only values which are in s1 but not in s2 |
| S1.symmetric_difference(s2)<br>S1^s2 | To find only values which are in s1 and only values which are in s2, common values will be ignored |
| S1.difference_update(s2)<br>S1=s1-s2 | To find only values which are in s1 but not in s2, but it will overwrite s1. |
| S1.symmetric_difference_update(s2)<br>S1=s1^s2 | To find only values which are in s1 and only values which are in s2, common values will be ignored |
| S1.issuperset(s2) | If s1 has all value of s2, plus something extra then it returns true |
| S1.issubset(s2) | If s2 has all value of s1, plus something extra then it returns true |
| S1.isdisjoint(s2) | If no values are common in s1 and s2 then it returns true |



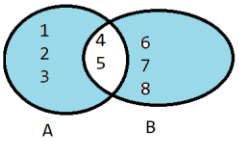

A.union(B)
A|B={1,2,3,4,5,6,7,8}



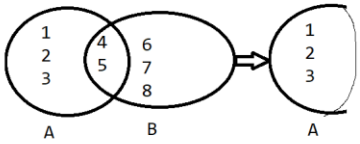

A.intersection(B)
A&B={4,5}



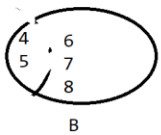

A.difference(B)
A-B={1,2,3}

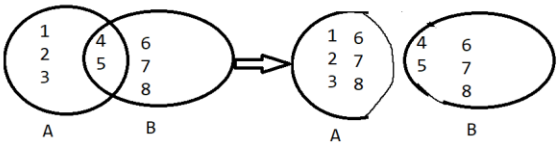



A.symmetric_difference(B)
A^B={1,2,3,6,7,8}

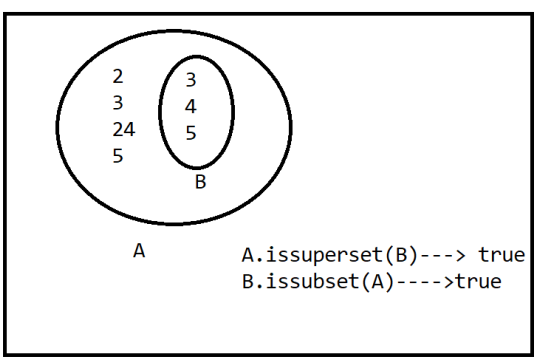


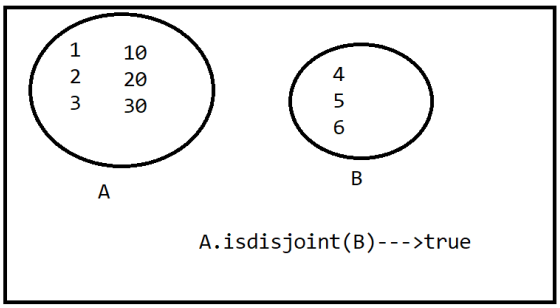

A.difference_update(B)
A=A-B




A.sysmmetric_difference_update(B)
A=A^B



Dictionary

1. It allows to store key→ value pair
2. It is ordered collection.
3. It is represented by using {}.
4. Key should be unique.

Whenever you want to store the data in key→value pair format then use dictionary
D={“python”:250,”java”:500,”c++”:450}
#to print value of key “python”, if key not found then it will throw exception
Print(D[“python”}

D[“.net”]=456 #if key is not there it will ad a new key
D[“python”]=356 #it will overwrite the value

D1={“a”:23,”b”:10}
D2={“b”:30,”c”:56”}
D1.update{D2)
D1=={“a”:23,”b”:30,”c”:56}

| | |
|---|---|
| D1.update(d2) | It will add all keys of d2 in d1, if keys are common then values of d1 will be overwritten , else new keys will be added |
| D1.keys() | It retrieves all the keys |
| D1.values() | It retrieves all the values from the dictionary |
| D1.items() | It retrieves a key-value pair as a tuple |
| D1.pop(key,[defaultvalue]) | It deletes the key-value pair, if key exists,<br>If key is not there and if default value is not given then it will throw exception, otherwise it will return default value |
| D1.popitem() | Delete the last key value pair |
| D1.get(key,[defaultvalue]) | This will return the value of key if key exists, otherwise. If default value is not given then it will return None, else it will return default value |
| D1.setdefault(key,defaultvalue) | This will return the value of key if key exists, otherwise. It will add key->default value pair in the dictionary and returns default value |
| D1.fromkeys([lst],[defaultvalue]) | It will add all the values from the lst as keys, and value for all keys will be set to None if default value is not given, otherwise all the values will be default value. |
| D1.copy() | To create a shallow copy of dictionary |
| D1.clear() | To clear all key-value pairs |

Regular expression

| | |
|---|---|
| * | 0 or more occurrences of preceding pattern |
| + | 1 or more occurrences of preceding pattern |
| ? | 0 or 1 occurrences of preceding pattern |
| $ | Check the pattern at the end of line |
| ^ | Check the pattern at the end of line |
| . | Any single character |
| \d | Any one digit [0-9] |
| \D | Any one character other than digit [^0-9] |
| \s | One space character |
| \S | Any one character other than space |
| \w | Any one word character [A-Za-z0-9_] |
| \W | Any one non word character [^A-Za-z0-9_] |
| \b | Boundary character |
| \B | Non boundary character |
| {m} | Exactly m occurrences |
| {m,n} | Minimum m occurrences and maximum n occurrences |
| {m,} | Minimum m occurrences and maximum any occurrences |

Mobile
[0-9]{10} -→ exactly 10 digits

| | [Oo]r | ^[O0]r | [Oo]r$ | ^[Oo]r$ | \b[Oo]r\b |
|---|---|---|---|---|---|
| Origami is good | y | y | n | N | n |
| There is a tailor | y | n | y | N | n |
| This is normal | y | n | n | N | n |
| This or that | y | n | n | N | y |
| or | y | y | y | y | y |
| There is a cat | n | n | n | n | n |

This is home
\w+\s\w+\s\w+

This is home sweet home

^\w+\s\w+\s\w+$

Something is there somewhere
[Ss].*?e

| | |
|---|---|
| Re.search(pattern,string,flags) | Find the first occurrence of the given pattern anywhere in the string and returns a match object, if pattern not found, then it returns None |
| Re.match(pattern,string,flags) | Find the first occurrence of the given pattern only at the beginning of the string and returns a match object.<br>if pattern not found, then it returns None |
| Re.findall(pattern,string,flags) | Find all occurrence of the given pattern anywhere in the string and returns a list of strings.<br>if pattern not found, then it returns None |

| | |
|---|---|
| Re.finditer(pattern,string,flags) | Find all occurrence of the given pattern anywhere in the string and returns a list of match objects.<br>if pattern not found, then it returns None |
| Re.sub(pattern,,newstr,string,flags) | Find and replace all occurrence of the given pattern in the string with newstr and returns a newly generated string<br>if pattern not found, then it returns None |
| Re.compile(pattern,flags) | Generates a regular expression object, can be used with all functions in re module |

Exception handling

Exception

- ValueError
- KeyError
- IndexError
- ZeroDivisionError

Users

1. Novice
2. Expert
3. Hackers