

Sequences

Contents

- List
- Tuple
- Dictionary

List

- One of the very powerful sequences
- Widely used
- List
 - Similar to an array, but can contain heterogeneous type elements
 - e.g.
 - `stud = [1, 'Bhatt', 'Naman', 'M', 97.0, 95.2, True]`
- `list.ipynb`

Accessing and indexing list elements

- `[]` square brackets.

- `list.ipynb`

Create list using range()

- `range(start, stop, stepsize)`

- `list.ipynb`

Printing list

- `stud = [1, 'Saradar', "Bhagatsingh", 97.0, 95.2, 'M', 18]`
- `for i in stud:`
 `print(i)`

Updating list elements

- `stud = [1, 'azad', "chandrashekhar", 97.0, 95.2, 'M', 21]`
- `stud[0] = 11`
- `stud[1] = "AZAD"`
- `stud[-1] = 77`
- `stud[-2] = 'm'`

Delete list element

- `del stud[-1]` # pass index
- `del stud[5]`
- `stud.remove('AZAD')` #pass value
- `list.ipynb`

Reverse list

- `stud.reverse()`

- `list.ipynb`

Concatenate two lists

- $x = [1, 2, 3]$
- $y = [4, 5, 6]$
- $z = x + y$

Repetition of lists

- `lst = [1,2,3]`
- `lst2 = lst * 2`

Membership in lists

- `'ram' in stud`
- `0 in [1,2,3,4,0]`

Aliasing lists vs. cloning lists

- Assigning another name to existing list
- `x = [1,2,3,4,5]`
- `y = x`

vs.

```
y = x[:]
```

list.ipynb

List methods

- `sum()` function: returns sum of all elements in the list
- `index(x)`: returns the index of first occurrence of `x` in the list
- `append(x)`: appends `x` at the end of the list
- `insert(i,x)`: inserts `x` in the list at the index `i`
- `copy()`: copies all the list elements into a new list and returns it
- `extend(lst1)`: appends list `lst1` to list
- `count(x)`: returns count of occurrences of `x` in the list
- `remove(x)`: removes `x` from the list
- `pop()`: removes the ending element from the list
- `sort()`: sorts the list's elements in ascending order
- `reverse()`: reverses the sequence of elements in the list
- `clear()`: deletes all elements from the list

Tuple

- A python sequence
 - Stores a group of elements or items
- Tuple is immutable
- List is mutable

Create tuple

- `t = (1,2,3)`
- `t = (1,)` # tuple with one element. Observe comma
- `t = ()` # empty tuple
- `t = (1,'one',2,'two')`
- `t = 1,'one',2,'two'`

- `lst = [1,2,3]`
- `tpl = tuple(lst)`

- `tuple.ipynb`

Indexing / slicing tuple elements

- `t = (1,2,3,4,5,6,7,8,9)`
 - `t[0]`
 - `t[-1]`
 - `t[1:4]`
-
- `tuple.ipynb`

Repeat tuple elements

- `t = (100.50,) * 4`

- `tuple.ipynb`

Functions to process tuples

- `len(tpl)` : returns number of elements in tuple
- `min(tpl)` : returns smallest element in tuple
- `max(tpl)` : returns biggest element in tuple
- `sorted(tpl)` : sorts tuple elements in ascending order. Use `reverse=True` for descending order
 - Returns list
 - Original tuple remains intact
- Methods:
 - `tpl.count(x)` : returns how many times 'x' occurs in tpl
 - `tpl.index(x)` : returns index of first occurrence of 'x' in tpl
- `tuple.ipynb`

Dictionary

- A group of elements
 - Arranged in key-value pairs.
- Create dictionary
 - `d1 = {'id':1,'name':'ram','salary':100}`
- `len(d1)`
- Operations
 - Update element
 - Delete element

Dictionary methods

- `d.clear()`: removes all key-value pairs from `d`
- `d2 = d.copy()`: copies all elements from `d` into new dictionary `d2`

Traversing dictionary

- `d.keys()`
- `d.values()`
- `d.items()`
- `d.get(k)`

Creating dictionary from lists

- `countries = ['usa','india','Germany']`
- `cities = ['wahington','delhi','berlin']`
- `z = zip(countries, cities)`
- `d = dict(z)`

dictionary methods

- `d.clear()` : removes all key-values pairs from `d`
- `d1 = d.copy()`: copies all elements from `d` into a new dictionary `d1`
- `d.fromkeys(s[,v])`: Returns a new dictionary with keys from sequence `s` and values all set to `v`
- `d.get(k[,v])`: returns the value associated with key 'k'. If the key 'k' is not found, it returns `v`.
- `d.items()`: returns an object that contains key-value pairs of `d`. The pairs are stored as tuples in the object.
- `d.keys()`: returns a sequence of keys from `d`.
- `d.values()`: returns a sequence of values from dictionary `d`.
- `d.update(x)`: adds all elements from dictionary `x` to `d`.

dictionary methods

- `d.pop(k[,v])`: removes the key `k` and its value from `d` and returns the value. If a key is not found, then the value `'v'` is returned. If key is not found and `'v'` is not mentioned, then `KeyError` is raised.
- `d.setdefault(k[,v])`: if key `k` is found, its value is returned. If key is not found, then the `k, v` pair is stored into the dictionary `d`.