Basic Python

data containers

Basic data container

list (*) A=[1,2,3,4]

Tuple (a special case of list, immutable list)

dictionary D={‘A’:0,’C’:1,’G’:2,’T’:3}

Set a={1, 2, 3, 4}

List

List is almost the most important Data Type in Python.

- List is an ordered collection of objects.
- List can contain variables of any data type.
- Same List can contain variables of different data types.
- A=[4.0,6.0,7.0,9.7,1.2]

Common list functions

- A[x], get element in list A at position x
- A.append(x), append x to A list
- A.insert(p,x), insert x to A at position p
- A. index(x), get the first position of x in A, error it not found
- A.count(x), get the occurrence of x in A
- A.sort(reverse=True/False), sort list A. True/False set the sorting order.
- A[x:y] , get the subset of list A from position x to position y.

List operations

```
mylist = []
mylist.append(1)
mylist.append(2)
mylist.append(3)
print(mylist[0]) # prints 1
print(mylist[1]) # prints 2
print(mylist[2]) # prints 3
# prints out 1,2,3
for x in mylist:
print(x)
```

```
a = [23, 24, 40, 34, 25, 24]
print(f"Length of a :{len(a)}")

print(f'Before: {a}')
a.append(50)
print(f"After append 50 to the list :{a}")

 a.pop()
 print(f"After pop last element from the
 list :{a}")

 a.insert(2, 15)
 print(f"After add 15 to 3rd position of the list :{a}")

 a.pop(2)
 print(f"After pop 3rd element in the list :{a}")
```

List operations

```
# List concatenation
list_a, list_b = [1, 2, 3, 4, 5], [11, 12, 13, 14, 15, 16, 17]
list_c = list_a + list_b
print(f'list c:{list_c}')

# Reverse list
list_a = [1, 2, 3, 4, 5]
print(f'One way to reverse a list: {list_a[::-1]}')
list_a.reverse()
print(f'Another way to reverse a list: {list_a}')
```

```
# Sort list
list_a = [1, 5, 2, 3, 4]
print(f'before: {list_a}')
list_a.sort()
print(f'after(ascending): {list_a}')

list_a.sort(reverse = True)
print(f'after(descending): {list_a}')
```

```
# Another way to sort list
list_a = [1, 5, 2, 3, 4]
print(f'before: {list_a}')
list_a = sorted(list_a)
print(f'after(ascending): {list_a}')
list_a = sorted(list_a, reverse=True)
print(f'after(descending): {list_a}')
```

List comprehension

A succinct way to create a list from another list

```
sentence = "the quick brown fox jumps over the lazy dog“

words = sentence.split()

word_lengths = []

for word in words:
  if word != "the":
    word_lengths.append(len(word))

print(words)

print(word_lengths)
```

```
sentence = "the quick brown fox jumps over the lazy dog“

words = sentence.split()

word_lengths = [len(word) for word in words if word != "the"]

print(words)

print(word_lengths)
```

Tuple

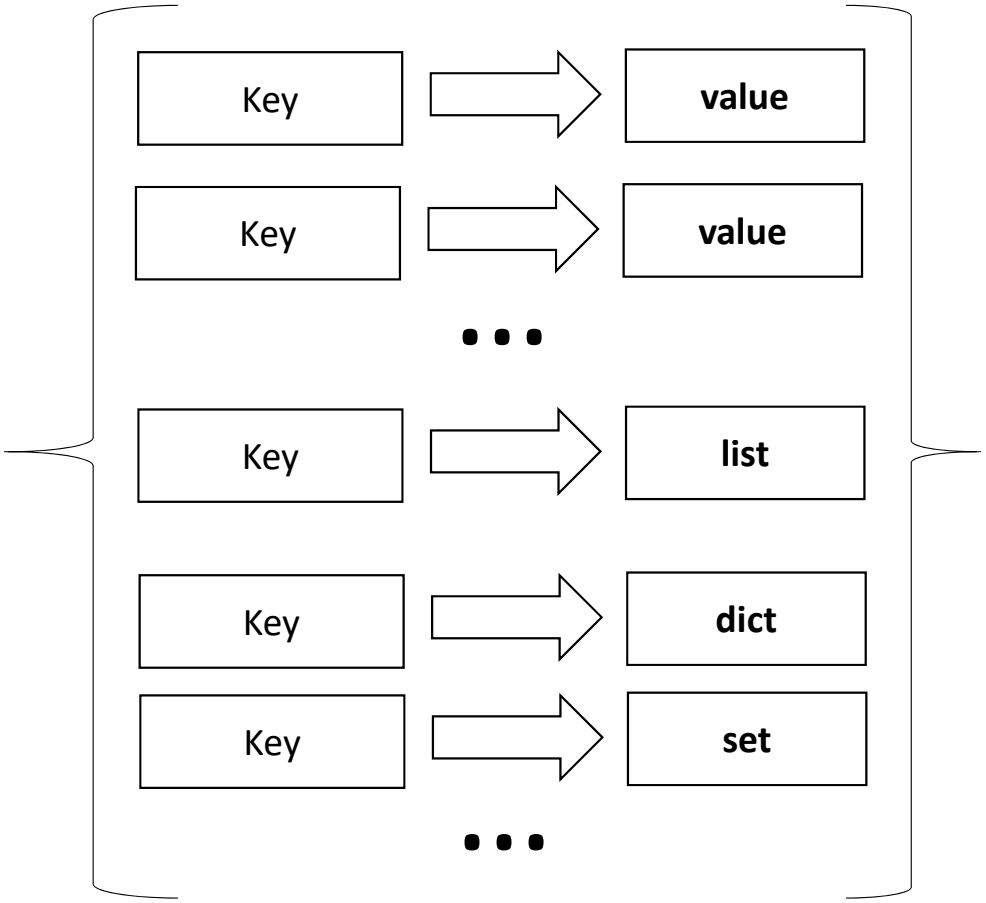
- Tuple is a like a list in every way, except it cannot be changed.
- To change a tuple, it is converted into a list first, then the list is changed and converted back to the tuple.
- A list can not be used as key of a dictionary, while tuple can.

Tuple operations

```
  a = ('Name', 'Age', 'Height')

  print(a[0])
  print(a[1:2])
  print(a[1:3])

  b = list(a)
  b[0] = 'Mike'
  a = tuple(b)
  print(a)

# The following will result in error
a.pop()
a.append('new_element')
a[0]=1
```

Dictionary

- An unordered data structure that stores data as a form of {key: value} pairs.
- Helps to access or change a value quickly by its key.
- Key must be number, string or tuple.
- Value can be any data type



Dictionary

```
a = {"Name":"Jason", "Age": 23, "Height":167.5}
# look up value in dictionary
print(f'a["Name"]:{a["Name"]}')
print(f'a["Age"]:{a["Age"]}')
print(f'a["Height"]:{a["Height"]}')
print(a["Heightaaqaa"])

print(a.keys())
print(a.values())
print(a.items())

b = {}
b["Name"] = "Jason"
b["Age"] = 23
b["Height"] = 167.5

print(b.keys())
print(b.values())
print(b.items())
print(b)
```

Dictionary Comprehension

A succinct way to create a dictionary from a list

```
a = [["Name", "Jason"], ["Age", 23],
["Height",167.5]]

b = {}

for item in a:
  key, value = item
  b[key] = value

print(b)

c = {key:value for key, value in a}
print(c)
```

Set

- Set is an unordered collection of unique elements.
- Set data structure is used to perform the set operations e.g., join, intersection, subtract, check if an element exists, check if a set is subset of another set etc.
- Tuple and list can have duplicate values but set and dictionary cannot.

```
a = {1, 2, 3, 4}
print('Set:', a)
print(f"Lenght of set: {len(a)}")

a.add(5)
a.remove(3)
a.add(5)

print(f"Set after update:{a}")
```

Set Operations

```
# Set operation
a = {1, 2, 3, 5}
b = {1, 2, 9}

print(f"Join:{a.union(b)}")
print(f"Join(another way):{a|b}")

print(f"Intersect:{a.intersection(b)}")
print(f"Intersect(another way):{a&b}")

print(f"Subtract:{a.difference(b)}")
print(f"Subtract(another way)):{a-b}")

print(f"Is 5 in a:{5 in a}")
print(f"Is b subset of a:{b.issubset(a)}")
print(f"Is b subset of a(another way):{b <= a}")
print(f"Is b proper subset of a in a:{b < a}")
```