list-set-dict

July 19, 2024

1 List

A sequential collection of elements that is mutable , ordered and allows heterogeneous data types.

1.1 Creating

[1]:
```
print(list((1,2,3,4)))      # Create a list using  list(sequence)
print(list())               # Create a empty list using list()
print(list('Hello'))        # Create a list of string split up
print([])                   # Create a list using '[]'
```

```
[1, 2, 3, 4]
[]
['H', 'e', 'l', 'l', 'o']
[]
```

1.2 Manipulation

consider list a =[9,8,3,4,2] as a list used to manipulate

[2]:
```
a=[9,8,3,4,2]
```

[3]:
```
a.append(3)          # Add item 3 at the end
```

[4]:
```
print(a)
```

```
[9, 8, 3, 4, 2, 3]
```

[5]:
```
a.extend([7,8,9])        # Add a list at the end
print(a)
```

```
[9, 8, 3, 4, 2, 3, 7, 8, 9]
```

[6]:
```
a.insert(4,0)            #Insert a item 0 at index 4
print(a)
```

```
[9, 8, 3, 4, 0, 2, 3, 7, 8, 9]
```

[7]:
```
a.remove(0)              # Removes the item 0 if exists else it throws error
```

[8]:
```
a.pop()                  # Removes and returns the item at the specified index␣
 ↪or by default last
```

[8]: 9

[9]:
```
a.sort()                 # Sorts the list and dont give output
print(a)
```

```
[2, 3, 3, 4, 7, 8, 8, 9]
```

[10]:
```
a.reverse()              # Reverse the list
print(a)                 # Reverse dont give output
```

```
[9, 8, 8, 7, 4, 3, 3, 2]
```

[11]:
```
a.clear()                # Removes all the elements in the list
print(a)
```

```
[]
```

[ ]:
```

```

1.3 Accessing and slicing

consider a=[1,2,3,4,5]

[12]:
```
a=[1,2,3,4,5]
```

[13]:
```
print(a[3])          # Accesing the item at index i
print(a[1:4:2])      # Slicing the item by start index , stop index, step
print(a[-2])         # Access the element by negative index
```

```
4
[2, 4]
4
```

1.4 Concatenation and repetition

consider a=[1,2,3]

[14]:
```
a=[1,2,3]
```

[15]:
```
a+[5,6,7]    # Concat element using + operator
```

[15]: [1, 2, 3, 5, 6, 7]

[16]:
```
a*3          # Repetion of elements
```

[16]: [1, 2, 3, 1, 2, 3, 1, 2, 3]

1.5 Additional Info

consider a=[1,2,3,4,5]

[17]:
```
a=[1,2,3,4,5]
```

[18]:
```
print(len(a))            # Print length of a
print(a.count(3))        # Print the no of items repeated in a
print(a.index(4))        # Print the index of the first occurance of the item
 ↪also can be manipulated by start index and stop index
```

```
5
1
3
```

[19]:
```
b=a.copy()               # Create a shallow copy of the list
c=a                     # This creates a direct copy of list into c if c gets
 ↪manipulated a get manipulated too
```

[20]:
```
print('id(a)',id(a),'\nid(b)',id(b),'\nid(c)',id(c))
```

```
id(a) 2214200248768
id(b) 2214200272704
id(c) 2214200248768
```

[21]:
```
c.append('If you dont use copy this will be added to original list')
print(f'a={a}\nb={b}\nc={c}')
```

```
a=[1, 2, 3, 4, 5, 'If you dont use copy this will be added to original list']
b=[1, 2, 3, 4, 5]
c=[1, 2, 3, 4, 5, 'If you dont use copy this will be added to original list']
```

[22]:
```
print(4 in a)        # Return true if 4 in the list else false
print(4 not in a)    # Return true if 4 in the list else false
```

```
True
False
```

[23]:
```
a=[5,4,3,2,1]
print(list(enumerate(a)))       # Returns a enumerate object which creates
 ↪indices to the list
print(sorted(a))                # Returns a list of sorted list
print(''.join(list(map(str,a))))              # Joins all the items in the list
 ↪into string with a delimiter
```

```
[(0, 5), (1, 4), (2, 3), (3, 2), (4, 1)]
[1, 2, 3, 4, 5]
54321
```

[24]:
```
print(all(a))                    # Prints true if all elements are true
print(any(a))                    # Print true if any elements in the list is true
```

```
True
True
```

[25]:
```
a=[i for i in range(5,0,-1)]         # Create a list using list comprehension
a
```

[25]:
```
[5, 4, 3, 2, 1]
```

[26]:
```
a=[[ 1 if i==j else 0 for i in range(0,5)] for j in range(0,5)]      # Create a␣
 ↪2 list using List comprehension
a
```

[26]:
```
[[1, 0, 0, 0, 0],
 [0, 1, 0, 0, 0],
 [0, 0, 1, 0, 0],
 [0, 0, 0, 1, 0],
 [0, 0, 0, 0, 1]]
```

2 Set

Set is a unordered collection of unique items. It is defines by placing comma-separated values inside curly braces '{}'

2.1 Creating

[27]:
```
a=[1,2,3,4]
print(set())         # Create a empty set
print({})            # Create a set with specified items
print(set())
```

```
set()
{}
set()
```

2.2 Manipulation

consider a={1,2,3,4,5}

[28]:
```
a={1,2,3,4,5}
```

[29]:
```
a.add(6)                # Adds an item to the set
print(a)
a.remove(6)             # Removes the item in the set if it exists ,else␣
 ↪returns error
print(a)
a.discard(6)            # Removes the item in the set if it exists ,else␣
 ↪returns none
print(a)
print(a.pop())          # Removes and returns arbitrary item from the set
a.update({6,7,8,9})     # Update a set with the union of itself and others.
print(a)
a.clear()               # Removes all the elements in the set
print(a)
```

```
{1, 2, 3, 4, 5, 6}
{1, 2, 3, 4, 5}
{1, 2, 3, 4, 5}
1
{2, 3, 4, 5, 6, 7, 8, 9}
set()
```

[30]:
```
a={1,2,3,4,5}
b=a.copy()
c=a
```

[31]:
```
print(f'id(a) : {id(a)} \nid(b) : {id(b)} \nid(c) : {id(c)} \nId a and c are␣
 ↪same')
```

```
id(a) : 2214200198528
id(b) : 2214200198976
id(c) : 2214200198528
Id a and c are same
```

[32]:
```
a.add(7)
print(f'a = {a}\nb = {b} \nc = {c}')    # If a get changed c also get changed␣
 ↪but not b
```

```
a = {1, 2, 3, 4, 5, 7}
b = {1, 2, 3, 4, 5}
c = {1, 2, 3, 4, 5, 7}
```

2.3 Set operations

[33]:
```
a={1,2,3,4,5}
b={4,5,6,7,8}
```

[34]:
```
# Union operation
print(a.union(b))       # Returns set with elements from both sets
print(a|b)              # Same as union
```

```
{1, 2, 3, 4, 5, 6, 7, 8}
{1, 2, 3, 4, 5, 6, 7, 8}
```

[35]:
```
# Intersection
print(a.intersection(b))        # Returns common elements in the set
print(a&b)                      # Same as intersection
```

```
{4, 5}
{4, 5}
```

[36]:
```
# Difference
print(a.difference(b))          # Returns elements in a not in b
print(a-b)                      # Same as difference
```

```
{1, 2, 3}
{1, 2, 3}
```

[37]:
```
#Symmetric Difference
print(a.symmetric_difference(b))        # Returns a set that are not in both set
print(a^b)                              # Same as Symmetric difference
```

```
{1, 2, 3, 6, 7, 8}
{1, 2, 3, 6, 7, 8}
```

[38]:
```
# Check subset and superset
print(a.issubset({1,2,3}))          # Returns True if a is subset
print(a.issuperset({1,2,3}))        # Returns True if a is superset
```

```
False
True
```

2.4 Membership tests

[39]:
```
print(4 in a)           # Prints true if 4 in a
print(4 not in a)       # Prints true if 4 not in a
```

```
True
False
```

2.5 Iterations

[40]:
```
# For Iteration
for i in a:
    print(i)
```

```
1
2
3
4
5
```

[41]:
```
# Enumerate()
set(enumerate(a))   # Returns a enumerate object , yeilding pairs of indexs and␣
 ↪an element from the set
```

[41]:
```
{(0, 1), (1, 2), (2, 3), (3, 4), (4, 5)}
```

[42]:
```
b=iter(a)   # Creates a iterator object
```

[43]:
```
b.__next__()       # Can be used to return next element of the sequence
```

[43]:
```
1
```

[ ]:
```

```

3 Dictionary

Dictionary is a collection of key value pairs, where key must be unique and is used to access its corresponding value. defined by '{}'

3.1 Creating

[44]:
```
print(dict())                                  # Create a empty dict using dict␣
 ↪function
print({1:[1,2,3],2:[2,3,4]})                   # Create a dict using curly braces
print(dict(one=[1,2,3],two=[4,5,6]))           # Create a dict using dict function
print(dict([('a', 1), ('b', 2), ('c', 3)])) # Create a dict using list of tuples
```

```
{}
{1: [1, 2, 3], 2: [2, 3, 4]}
{'one': [1, 2, 3], 'two': [4, 5, 6]}
{'a': 1, 'b': 2, 'c': 3}
```

3.2 Accessing Elements

consider a={1: [1,2,3], 2: [4,5,6], 3: [7,8,9]}

[45]:
```
a={1: [1,2,3], 2: [4,5,6], 3: [7,8,9]}  # Returns the value associated with key
print(a[1])
```

```
[1, 2, 3]
```

[46]:
```
print((a.get(1)))   # Returns the Value associated with the dict else it will␣
 ↪returns a default value(none) or we can return default value
print(a.get(4))     # key 4 doesnt exist so it give None as output
print(a.get(4,'This is the Output'))    # Default output can be changed
```

```
[1, 2, 3]
None
This is the Output
```

3.3 Manipulations

consider a={1: [1,2,3], 2: [4,5,6], 3: [7,8,9]}

[47]:
```
a={1: [1,2,3], 2: [4,5,6], 3: [7,8,9]}
```

[48]:
```
a[4]=[11,12,13]     # Sets the Value with the can and can be the existing key
print(a)
a.update({5:[14,15,16],6:[17,18,19]}) # Update the dictionary with key-value␣
 ↪pairs
print(a)
```

```
{1: [1, 2, 3], 2: [4, 5, 6], 3: [7, 8, 9], 4: [11, 12, 13]}
{1: [1, 2, 3], 2: [4, 5, 6], 3: [7, 8, 9], 4: [11, 12, 13], 5: [14, 15, 16], 6:
[17, 18, 19]}
```

[49]:
```
print(a.pop(6,'No output')) # Pops the specified key - value else like get it␣
 ↪will return default
print(a)
```

```
[17, 18, 19]
{1: [1, 2, 3], 2: [4, 5, 6], 3: [7, 8, 9], 4: [11, 12, 13], 5: [14, 15, 16]}
```

[50]:
```
print(a.popitem()) # removes and returns an arbititary key-value pair as tuple␣
 ↪if dict is empty then key error
```

```
(5, [14, 15, 16])
```

[51]:
```
print(a.setdefault(5,[14,15,16]))  # Checks if the key exists else creates a␣
 ↪key and adds the default value
```

```
[14, 15, 16]
```

[52]:
```
print(dict.fromkeys([1,2],[17,18,19]))   # This method creates a new dictionary␣
↪with keys from a given iterable (like a list or tuple).
```

```
{1: [17, 18, 19], 2: [17, 18, 19]}
```

[53]:
```
b=a.fromkeys([7,8,9],[17,18,19])
b
```

[53]:
```
{7: [17, 18, 19], 8: [17, 18, 19], 9: [17, 18, 19]}
```

[54]:
```
b=a.copy()                    # Creates a shallow copy to b
```

[55]:
```
a.clear()                     # Empties the dictionary
```

3.4 Iterations

consider a={1: [1,2,3], 2: [4,5,6], 3: [7,8,9]}

[56]:
```
a={1: [1,2,3], 2: [4,5,6], 3: [7,8,9]}
```

[57]:
```
for i,j in a.items():
    print(i,j)
```

```
1 [1, 2, 3]
2 [4, 5, 6]
3 [7, 8, 9]
```

[58]:
```
print([i for i in a])           # Iterate over the keys in dict
print([i for i in a.keys()])    # Iterate over the keys in dict
print([i for i in a.values()])  # Iterate over the values in dict
```

```
[1, 2, 3]
[1, 2, 3]
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

[59]:
```
print(a.items())    # Returns view object of the dict's key-value pair
print(a.keys())     # Returns view object of the dict's keys
print(a.values())     # Returns view object of the dict's values
```

```
dict_items([(1, [1, 2, 3]), (2, [4, 5, 6]), (3, [7, 8, 9])])
dict_keys([1, 2, 3])
dict_values([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
```

3.5 Membership Test

[60]:
```
print(3 in a )       # Returns True if 3 in a
print(3 not in a)    # Returns True if 3 not in a
print([1,2,3] in a.values())    # Can be modified to get memberships of the dict
```

```
True
False
True
```

I hope you found this information helpful! Feel free to save this post for future reference. Let's continue to learn and grow together!

Rajendra Prasad