Python Data Structures: List, Tuple, Set, Dictionary

LIST Functions

[append]

Explanation: Adds an element at the end of the list.

Syntax: list.append(element)

```
numbers = [1, 2, 3]
numbers.append(4)
print(numbers)  # [1, 2, 3, 4]
```

[insert]

Explanation: Inserts an element at a specific index.

Syntax: list.insert(index, element)

```
numbers = [1, 2, 3]
numbers.insert(1, 10)
print(numbers)  # [1, 10, 2, 3]
```

[remove]

Explanation: Removes the first occurrence of a value.

Syntax: list.remove(value)

```
numbers = [1, 2, 3]
numbers.remove(2)
print(numbers)  # [1, 3]
```

[pop]

Explanation: Removes and returns the last element.

Syntax: list.pop([index])

```
numbers = [1, 2, 3]
numbers.pop()
print(numbers)  # [1, 2]
```

[sort]

Explanation: Sorts the list in ascending order.

Syntax: list.sort()

```
numbers = [3, 1, 2]
numbers.sort()
print(numbers)  # [1, 2, 3]
```

[reverse]

Explanation: Reverses the list.

Syntax: list.reverse()

```
numbers = [1, 2, 3]
numbers.reverse()
print(numbers)  # [3, 2, 1]
```

[count]

Explanation: Returns the number of times a value occurs.

Syntax: list.count(value)

```
numbers = [1, 2, 2, 3]
print(numbers.count(2))  # 2
```

[index]

Explanation: Returns the index of the first matching element.

Syntax: list.index(value)

```
numbers = [1, 2, 3]
print(numbers.index(2))  # 1
```

[clear]

Explanation: Removes all elements from the list.

Syntax: list.clear()

```
numbers = [1, 2, 3]
numbers.clear()
print(numbers)  # []
```

[extend]

Explanation: Adds all elements from another list.

Syntax: list.extend(iterable)

```
a = [1, 2]
b = [3, 4]
a.extend(b)
print(a)  # [1, 2, 3, 4]
```

TUPLE Functions

Python Data Structures: List, Tuple, Set, Dictionary

[count]

Explanation: Returns the number of times a value appears.

Syntax: tuple.count(value)

```
t = (1, 2, 2, 3)
print(t.count(2))  # Output: 2
```

[index]

Explanation: Returns the index of the first occurrence of a value.

Syntax: tuple.index(value)

```
t = (10, 20, 30, 20)
print(t.index(20))  # Output: 1
```

SET Functions

[add]

Explanation: Adds an element to the set.

Syntax: set.add(element)

```
s = {1, 2}
s.add(3)
print(s)  # {1, 2, 3}
```

[remove]

Explanation: Removes the specified element.

Syntax: set.remove(element)

```
s = {1, 2, 3}
s.remove(2)
print(s)  # {1, 3}
```

[discard]

Explanation: Removes the element if it exists.

Syntax: set.discard(element)

```
s = {1, 2}
s.discard(5)
print(s)  # {1, 2}
```

[pop]

Explanation: Removes and returns a random element.

Syntax: set.pop()

```
s = {10, 20, 30}
s.pop()
print(s)
```

[clear]

Explanation: Removes all elements from the set.

Syntax: set.clear()

```
s = {1, 2, 3}
s.clear()
print(s)  # set()
```

[union]

Explanation: Returns a new set with all items from both sets.

Syntax: set1.union(set2)

```
a = {1, 2}
b = {3, 4}
print(a.union(b))  # {1, 2, 3, 4}
```

[intersection]

Explanation: Returns a set with items common to both sets.

Syntax: set1.intersection(set2)

```
a = {1, 2, 3}
b = {2, 3, 4}
print(a.intersection(b))  # {2, 3}
```

[difference]

Explanation: Returns items in set1 but not in set2.

Syntax: set1.difference(set2)

```
a = {1, 2, 3}
b = {2, 4}
print(a.difference(b))  # {1, 3}
```

[update]

Explanation: Adds all items from another set into the current set.

Syntax: set1.update(set2)

```
a = {1}
b = {2, 3}
a.update(b)
print(a)  # {1, 2, 3}
```

[issubset]

Explanation: Checks if one set is a subset of another.

Syntax: set1.issubset(set2)

```
a = {1, 2}
b = {1, 2, 3}
print(a.issubset(b))  # True
```

[issuperset]

Explanation: Checks if one set is a superset of another.

Syntax: set1.issuperset(set2)

```
a = {1, 2, 3}
b = {1, 2}
print(a.issuperset(b))  # True
```

[symmetric_difference]

Explanation: Returns elements in either set but not in both.

Syntax: set1.symmetric_difference(set2)

```
a = {1, 2, 3}
b = {3, 4}
print(a.symmetric_difference(b))  # {1, 2, 4}
```

DICTIONARY Functions

[get]

Explanation: Returns the value of a key.

Syntax: dict.get(key)

```
student = {'name': 'Amit', 'age': 20}
print(student.get('name'))  # Amit
```

[keys]

Explanation: Returns a view object containing all keys.

Syntax: dict.keys()

```
student = {'name': 'Amit', 'age': 20}
print(student.keys())
```

[values]

Explanation: Returns a view object containing all values.

Syntax: dict.values()

```
student = {'name': 'Amit', 'age': 20}
print(student.values())
```

[items]

Explanation: Returns all key-value pairs as tuples.

Syntax: dict.items()

```
student = {'name': 'Amit', 'age': 20}
print(student.items())
```

[update]

Explanation: Adds new key-value pairs or updates existing ones.

Syntax: dict.update(other_dict)

```
student = {'name': 'Amit'}
student.update({'age': 20})
print(student)
```

[pop]

Explanation: Removes the specified key and returns its value.

Syntax: dict.pop(key)

```
student = {'name': 'Amit', 'age': 20}
age = student.pop('age')
print(age)
```

[popitem]

Explanation: Removes and returns the last inserted key-value pair.

Syntax: dict.popitem()

```
student = {'name': 'Amit', 'age': 20}
item = student.popitem()
```

```
print(item)
```

[clear]

Explanation: Removes all items from the dictionary.

Syntax: dict.clear()

```
student = {'name': 'Amit', 'age': 20}
student.clear()
print(student)
```

[fromkeys]

Explanation: Creates a dictionary from a list of keys with the same default value.

Syntax: dict.fromkeys(keys, value)

```
keys = ['name', 'age']
d = dict.fromkeys(keys, 0)
print(d)
```

[setdefault]

Explanation: Returns value of key, inserts it if not present.

Syntax: dict.setdefault(key, value)

```
student = {'name': 'Amit'}
student.setdefault('age', 20)
print(student)
```