Report on Built-in Methods in Python

I. str (String) Methods

1. upper()

- Syntax

```
string.upper()
```

- Documentation

  Returns a copy of the string with all characters converted to uppercase.

- Usage

  Used to format strings or prepare for case-insensitive comparison.

- Output

```
text = "hello world"
result = text.upper()
print(result)
# Output: HELLO WORLD
```

2. strip()

- Syntax

```
string.strip([chars])
```

- Documentation

  Removes leading and trailing characters (default is whitespace).

- Usage

  Used to clean up user input or formatted strings.

- Output

```
text = "   Python  "
result = text.strip()
print(result)
# Output: Python
```

3. replace()

- Syntax

  ```
  string.replace(old, new[, count])
  ```

- Documentation

  Replaces all (or some) occurrences of a substring with another string.

- Usage

  Used for text substitution in strings.

- Output

  ```
  text = "I love Java"
  result = text.replace("Java", "Python")
  print(result)
  # Output: I love Python
  ```

4. split()

- Syntax

  ```
  string.split([separator[, maxsplit]])
  ```

- Documentation

  Splits a string into a list using the specified separator.

- Usage

  Used to tokenize or parse text.

- Output

  ```
  text = "apple,banana,cherry"
  result = text.split(",")
  print(result)
  # Output: ['apple', 'banana', 'cherry']
  ```

5. find()

- Syntax

  ```
  string.find(sub[, start[, end]])
  ```

- Documentation

  Finds the first occurrence index of a substring. Returns -1 if not found.

- Usage

    Used for searching within strings.

- Output

```
text = "Welcome to Python"
index = text.find("Python")
print(index)
# Output: 11
```

II. list (List) Methods

1. append()

- Syntax

```
list.append(item)
```

- Documentation

    Appends an item to the end of the list.

- Usage

    Used for building lists dynamically.

- Output

```
fruits = ["apple", "banana"]
fruits.append("cherry")
print(fruits)
# Output: ['apple', 'banana', 'cherry']
```

2. remove()

- Syntax

```
list.remove(item)
```

- Documentation

    Removes the first occurrence of a value. Raises ValueError if not found.

- Usage

    Used to delete specific list items by value.

- Output

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

3. pop()

- Syntax

  list.pop([index])

- Documentation

  Removes and returns the item at the given index. Default is the last item.

- Usage

  Used when removing items and using their values.

- Output

```
colors = ["red", "green", "blue"]
removed = colors.pop(1)
print(removed)
print(colors)
# Output: green
# ['red', 'blue']
```

4. sort()

- Syntax

  list.sort(key=None, reverse=False)

- Documentation

  Sorts the list in-place in ascending order.

- Usage

  Used to organize data in a list.

- Output

```
numbers = [5, 3, 1, 4, 2]
numbers.sort()
print(numbers)
# Output: [1, 2, 3, 4, 5]
```

III. tuple (Tuple) Methods

1. count()

- Syntax

  tuple.count(item)

- Documentation

  Counts how many times an item appears in the tuple.

- Usage

  Used for frequency analysis.

- Output

```
data = (1, 2, 2, 3)
print(datcount(2))
# Output: 2
```

2. index()

- Syntax

  tuple.index(item)

- Documentation

  Returns the index of the first occurrence of the item.

- Usage

  Used to locate items in a tuple.

- Output

```
data = ("a", "b", "c")
print(datindex("b"))
# Output: 1
```

IV. dict (Dictionary) Methods

1. get()

- Syntax

  dict.get(key, default=None)

- Documentation

  Returns the value for the specified key. Returns default if key is not found.

- Usage

  Used to safely access dictionary values.

- Output

```
student = {"name": "Alex", "age": 21}
print(student.get("name"))
print(student.get("grade", "Not Found"))
# Output: Alex
# Not Found
```

2. keys()

- Syntax

  dict.keys()

- Documentation

  Returns a view object of all the dictionary's keys.

- Usage

  Used to iterate through keys.

- Output

```
user = {"id": 1, "name": "Bob"}
print(user.keys())
# Output: dict_keys(['id', 'name'])
```

3. items()

- Syntax

  dict.items()

- Documentation

  Returns a view object of key-value pairs.

- Usage

  Used in loops to get both keys and values.

- Output

```
user = {"id": 1, "name": "Bob"}
for key, value in user.items():
    print(key, "->", value)
# Output: id -> 1
# name -> Bob
```

4. update()

- Syntax

  ```
  dict.update([other])
  ```

- Documentation

  Updates the dictionary with key-value pairs from another dictionary or iterable.

- Usage

  Used for merging or modifying dictionaries.

- Output

  ```
  info = {"name": "Tom"}
  info.update({"age": 25})
  print(info)
  # Output: {'name': 'Tom', 'age': 25}
  ```

V. set (Set) Methods

1. add()

- Syntax

  ```
  set.add(element)
  ```

- Documentation

  Adds an element to the set.

- Usage

  Used to insert elements dynamically into sets.

- Output

  ```
  fruits = {"apple", "banana"}
  fruits.add("cherry")
  print(fruits)
  # Output: {'banana', 'apple', 'cherry'}
  ```

2. remove()

- Syntax

  set.remove(element)

- Documentation

  Removes the specified element. Raises KeyError if not found.

- Usage

  Used to delete elements by value.

- Output

```
numbers = {1, 2, 3}
numbers.remove(2)
print(numbers)
# Output: {1, 3}
```

3. discard()

- Syntax

  set.discard(element)

- Documentation

  Removes the element if present. Does not raise error if not found.

- Usage

  Safe removal of elements from sets.

- Output

```
items = {10, 20, 30}
items.discard(40)
print(items)
# Output: {10, 20, 30}
```

4. union()

- Syntax

  set1.union(set2)

- Documentation

  Returns a set with all elements from both sets.

- Usage

    Used to combine data from sets.

- Output

```
a = {1, 2}
b = {2, 3}
print(union(b))
# Output: {1, 2, 3}
```

5. intersection()

- Syntax

```
set1.intersection(set2)
```

- Documentation

    Returns a set of common elements.

- Usage

  Used for finding shared data.

- Output

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

6. difference()

- Syntax

```
set1.difference(set2)
```

- Documentation

    Returns a set with elements in the first but not the second set.

- Usage

    Used for comparison.

- Output

```
a = {1, 2, 3}
b = {3, 4}
```

```
print(difference(b))
# Output: {1, 2}
```

7. clear()

- Syntax

```
set.clear()
```

- Documentation

  Removes all elements from the set.

- Usage

  Used to empty a set.

- Output

```
a = {5, 6, 7}
clear()
print(a)
# Output: set()
```