**Assignment Solution Template: Exploring Python List Methods**

**Method 1: append()**

**Description:**

Appends a single item to the end of the list.

**Syntax:**

list.append(x)

**Parameters:**

`x`: The item to be added to the list.

**Return Type:**

None

**Example Usage:**

```python

my\_list = [1, 2, 3]

my\_list.append(4)

print(my\_list) # Output: [1, 2, 3, 4]

```

**Method 2: extend()**

**Description:**

Extends the list by appending elements from the iterable.

**Syntax:**

list.extend(iterable)

**Parameters:**

`iterable`: Any iterable object (list, tuple, etc.) whose elements will be added to the list.

**Return Type:**

None

**Example Usage:**

```python

my\_list = [1, 2, 3]

my\_list.extend([4, 5])

print(my\_list) # Output: [1, 2, 3, 4, 5]

```

**Method 3: insert()**

**Description:**

Inserts an item at a specified position in the list.

**Syntax:**

list.insert(i, x)

**Parameters:**

`i`: Index at which the item should be inserted.

`x`: The item to be inserted.

**Return Type:**

None

**Example Usage:**

```python

my\_list = [1, 2, 4]

my\_list.insert(2, 3)

print(my\_list) # Output: [1, 2, 3, 4]

```

**Method 4: remove()**

**Description:**

Removes the first occurrence of the specified item from the list.

**Syntax:**

`list.remove(x)`

**Parameters:**

`x`: The item to be removed.

**Return Type:**

None

**Example Usage:**

```python

my\_list = [1, 2, 3, 2]

my\_list.remove(2)

print(my\_list) # Output: [1, 3, 2]

```

**Method 5: pop()**

**Description:**

Removes and returns the item at the specified position in the list. If no index is specified, it removes and returns the last item in the list.

**Syntax:**

`list.pop([i])`

**Parameters:**

`i`: (Optional) Index of the item to be removed and returned.

**Return Type:**

The item removed from the list.

**Example Usage:**

```python

my\_list = [1, 2, 3]

item = my\_list.pop(1)

print(item) # Output: 2

print(my\_list) # Output: [1, 3]

```

**Method 6: clear()**

**Description:**

Removes all items from the list.

**Syntax:**

`list.clear()`

**Parameters:**

None

**Return Type:**

None

**Example Usage:**

```python

my\_list = [1, 2, 3]

my\_list.clear()

print(my\_list) # Output: []

```

**Method 7: index()**

**Description:**

Returns the index of the first occurrence of the specified item.

**Syntax:**

`list.index(x[, start[, end]])`

**Parameters:**

`x`: The item to search for.

`start`: (Optional) Starting index to search.

`end`: (Optional) Ending index to search.

**Return Type:**

int

**Example Usage:**

```python

my\_list = [1, 2, 3, 2]

index = my\_list.index(2)

print(index) # Output: 1

```

**Method 8: count()**

**Description:**

Returns the number of times the specified item appears in the list.

**Syntax:**

`list.count(x)`

**Parameters:**

`x`: The item to count.

**Return Type:**

int

**Example Usage:**

```python

my\_list = [1, 2, 2, 3]

count = my\_list.count(2)

print(count) # Output: 2

```

**Method 9: sort()**

**Description:**

Sorts the items of the list in place.

**Syntax:**

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

**Parameters:**

- `key`: (Optional) A function to execute to decide the order.

- `reverse`: (Optional) If `True`, the list is sorted in descending order.

\*\*Return Type:\*\*

`None`

\*\*Example Usage:\*\*

```python

my\_list = [3, 1, 2]

my\_list.sort()

print(my\_list) # Output: [1, 2, 3]

```

---

### Method 10: `reverse()`

\*\*Description:\*\*

Reverses the elements of the list in place.

\*\*Syntax:\*\*

`list.reverse()`

\*\*Parameters:\*\*

None

\*\*Return Type:\*\*

`None`

\*\*Example Usage:\*\*

```python

my\_list = [1, 2, 3]

my\_list.reverse()

print(my\_list) # Output: [3, 2, 1]

```

---

### Method 11: `copy()`

\*\*Description:\*\*

Returns a shallow copy of the list.

\*\*Syntax:\*\*

`list.copy()`

\*\*Parameters:\*\*

None

\*\*Return Type:\*\*

A new list.

\*\*Example Usage:\*\*

```python

my\_list = [1, 2, 3]

copy\_list = my\_list.copy()

print(copy\_list) # Output: [1, 2, 3]

Continuing with the remaining list methods:

---

### Method 12: `len()`

\*\*Description:\*\*

Returns the number of items in the list.

\*\*Syntax:\*\*

`len(list)`

\*\*Parameters:\*\*

None

\*\*Return Type:\*\*

`int`

\*\*Example Usage:\*\*

```python

my\_list = [1, 2, 3]

length = len(my\_list)

print(length) # Output: 3

```

---

### Method 13: `sorted()`

\*\*Description:\*\*

Returns a new list containing all items from the iterable in ascending order.

\*\*Syntax:\*\*

`sorted(list)`

\*\*Parameters:\*\*

- `list`: The list to be sorted.

\*\*Return Type:\*\*

A new sorted list.

\*\*Example Usage:\*\*

```python

my\_list = [3, 1, 2]

sorted\_list = sorted(my\_list)

print(sorted\_list) # Output: [1, 2, 3]

```

---

### Method 14: `sum()`

\*\*Description:\*\*

Sums start and the items of an iterable from left to right and returns the total.

\*\*Syntax:\*\*

`sum(iterable, start=0)`

\*\*Parameters:\*\*

- `iterable`: The iterable to sum.

- `start`: (Optional) The starting value to add to the sum.

\*\*Return Type:\*\*

`int` or `float`

\*\*Example Usage:\*\*

```python

my\_list = [1, 2, 3]

total = sum(my\_list)

print(total) # Output: 6

```

---

### Method 15: `min()`

\*\*Description:\*\*

Returns the smallest item in an iterable.

\*\*Syntax:\*\*

`min(iterable)`

\*\*Parameters:\*\*

- `iterable`: The iterable to find the minimum value from.

\*\*Return Type:\*\*

The smallest element in the list.

\*\*Example Usage:\*\*

```python

my\_list = [1, 2, 3]

minimum = min(my\_list)

print(minimum) # Output: 1

```

---

### Method 16: `max()`

\*\*Description:\*\*

Returns the largest item in an iterable.

\*\*Syntax:\*\*

`max(iterable)`

\*\*Parameters:\*\*

- `iterable`: The iterable to find the maximum value from.

\*\*Return Type:\*\*

The largest element in the list.

\*\*Example Usage:\*\*

```python

my\_list = [1, 2, 3]

maximum = max(my\_list)

print(maximum) # Output: 3

```

---

### Method 17: `list()` (Constructor)

\*\*Description:\*\*

Creates a new list.

\*\*Syntax:\*\*

`list([iterable])`

\*\*Parameters:\*\*

- `iterable`: (Optional) An iterable object whose items will be added to the list.

\*\*Return Type:\*\*

A new list.

\*\*Example Usage:\*\*

```python

new\_list = list([1, 2, 3])

print(new\_list) # Output: [1, 2, 3]

```

---

### Method 18: `list comprehension` (Not a method but a syntax)

\*\*Description:\*\*

Provides a concise way to create lists.

\*\*Syntax:\*\*

`[expression for item in iterable if condition]`

\*\*Parameters:\*\*

- `expression`: The expression used to generate each list element.

- `item`: The variable representing each element from the iterable.

- `iterable`: The iterable to loop over.

- `condition`: (Optional) A condition to filter elements.

\*\*Return Type:\*\*

A new list.

\*\*Example Usage:\*\*

```python

squares = [x\*\*2 for x in range(5)]

print(squares) # Output: [0, 1, 4, 9, 16]

```