# Methods & Functions

## Numbers

- `abs(x)` – Returns the absolute value of `x`.
- `round(x, n)` – Rounds `x` to `n` decimal places.
- `pow(x, y)` – Returns `x` raised to the power `y`.
- `divmod(x, y)` – Returns a tuple `(x // y, x % y)`.
- `bin(x)` – Converts `x` to a binary string.
- `oct(x)` – Converts `x` to an octal string.
- `hex(x)` – Converts `x` to a hexadecimal string.
- `int(x, base)` – Converts `x` to an integer (default base 10).
- `float(x)` – Converts `x` to a floating-point number.
- `complex(x, y)` – Returns a complex number `x + yj`

### Mathematical Functions

- `abs(x)` – Returns the absolute value of x.
- `pow(x, y, mod)` – Returns x raised to the power y, optionally modulo `mod`.
- `round(x, n)` – Rounds x to n decimal places.
- `divmod(x, y)` – Returns `(x // y, x % y)` as a tuple.

### Type Conversion Functions

- `int(x, base)` – Converts `x` to an integer (default base 10).
- `float(x)` – Converts `x` to a floating-point number.
- `complex(x, y)` – Returns a complex number `x + yj`.

### Number System Conversion Functions

- `bin(x)` – Converts `x` to a binary string.
- `oct(x)` – Converts `x` to an octal string.
- `hex(x)` – Converts `x` to a hexadecimal string.

## Boolean Methods

- **`bit_length()`** → Returns the number of bits needed to represent the boolean value.
- **`conjugate()`** → Returns the complex conjugate (same value for `bool`).
- **`to_bytes(length, byteorder)`** → Converts `bool` to bytes.
- **`from_bytes(bytes, byteorder)`** → Converts bytes to an integer (can be used with `bool`).

### General Boolean Functions

- **`bool(x)`** – Converts `x` to a Boolean (`True` or `False`).
- **`all(iterable)`** – Returns `True` if **all** elements in the iterable are `True`.
- **`any(iterable)`** – Returns `True` if **at least one** element in the iterable is `True`.

## List Methods

- **`append(x)`** – Adds an item `x` to the end of the list.
- **`extend(iterable)`** – Extends the list by appending elements from an iterable.
- **`insert(i, x)`** – Inserts item `x` at index `i`.
- **`remove(x)`** – Removes the first occurrence of `x` in the list.
- **`pop([i])`** – Removes and returns the item at index `i` (last item if index is not provided).
- **`clear()`** – Removes all items from the list.
- **`index(x, [start], [end])`** – Returns the index of the first occurrence of `x`.
- **`count(x)`** – Returns the number of times `x` appears in the list.
- **`sort(key=None, reverse=False)`** – Sorts the list in ascending order (or descending if `reverse=True`).
- **`reverse()`** – Reverses the list in place.
- **`copy()`** – Returns a shallow copy of the list.

**Only Functions**

- `list(iterable)` – Creates a list from an iterable (e.g., tuple, string, set).

## String Methods

**Case Conversion & Formatting**

- `capitalize()` – Converts the first character to uppercase, rest lowercase.
- `title()` – Converts the first character of each word to uppercase.
- `upper()` – Converts all characters to uppercase.
- `lower()` – Converts all characters to lowercase.
- `swapcase()` – Swaps uppercase characters to lowercase and vice versa.
- `casefold()` – Converts string to lowercase (more aggressive than `lower()`).
- `zfill(width)` – Pads the string with zeros on the left to make it `width` characters long.

**Checking String Properties**

- `isalpha()` – Returns `True` if all characters are alphabets.
- `isdigit()` – Returns `True` if all characters are digits.
- `isalnum()` – Returns `True` if all characters are alphanumeric (letters & numbers).
- `isspace()` – Returns `True` if all characters are whitespace.
- `islower()` – Returns `True` if all characters are lowercase.
- `isupper()` – Returns `True` if all characters are uppercase.
- `istitle()` – Returns `True` if string is titlecased (each word starts with uppercase).

**Searching & Finding Substrings**

- **`find(sub, start=0, end=len(string))`** – Returns index of first occurrence of `sub` (-1 if not found).
- **`rfind(sub, start=0, end=len(string))`** – Returns the last occurrence index of `sub` (-1 if not found).
- **`index(sub, start=0, end=len(string))`** – Like `find()`, but raises an error if not found.
- **`rindex(sub, start=0, end=len(string))`** – Like `rfind()`, but raises an error if not found.
- **`count(sub, start=0, end=len(string))`** – Counts occurrences of `sub` in string.
- **`startswith(prefix, start=0, end=len(string))`** – Checks if string starts with `prefix`.
- **`endswith(suffix, start=0, end=len(string))`** – Checks if string ends with `suffix`.

**String Modification**

- **`replace(old, new, count=-1)`** – Replaces occurrences of `old` with `new`.
- **`trip(chars=None)`** – Removes leading and trailing `chars` (whitespace by default).
- **`lstrip(chars=None)`** – Removes leading `chars`.
- **`rstrip(chars=None)`** – Removes trailing `chars`.

**Splitting & Joining Strings**

- **`split(sep=None, maxsplit=-1)`** – Splits string into a list using `sep` (default: whitespace).
- **`rsplit(sep=None, maxsplit=-1)`** – Splits from the right.
- **`splitlines(keepends=False)`** – Splits string at line breaks.
- **`partition(sep)`** – Splits string into three parts: before `sep`, `sep`, and after `sep`.
- **`rpartition(sep)`** – Like `partition()`, but starts from the right.

- `join(iterable)` – Joins iterable items into a string, using the string as a separator.

**Encoding & Justification**

- `encode(encoding='utf-8', errors='strict')` – Encodes string into bytes.
- `ljust(width, fillchar=' ')` – Left-aligns string in a field of width `width`.
- `rjust(width, fillchar=' ')` – Right-aligns string in a field of width `width`.
- `center(width, fillchar=' ')` – Centers string in a field of width `width`.

**String Encoding & Decoding Functions**

- `ord(char)` – Returns the Unicode code point of a character.
- `chr(int)` – Returns the character corresponding to a Unicode code point.
- `ascii(object)` – Returns a string with escape sequences for non-ASCII characters.
- `repr(object)` – Returns a string representation of an object.
- `format(value, format_spec)` – Formats a value according to `format_spec`.

- `str(object)` – Converts an object to a string.

## Tuple Methods

- `count(value)` – Counts occurrences of a value in the tuple.
- `index(value, start=0, end=len(tuple))` – Finds the first occurrence index of a value.

**Other Useful Operations on Tuples**

- Concatenation: `(1, 2) + (3, 4) → (1, 2, 3, 4)`
- Repetition: `('a',) * 3 → ('a', 'a', 'a')`
- Membership Test: `3 in (1, 2, 3) → True`
- Iteration: `for x in (1, 2, 3): print(x)`
- Length: `len((1, 2, 3)) → 3`
- Min/Max: `min((3, 1, 2)) → 1`, `max((3, 1, 2)) → 3`
- Conversion: `tuple([1, 2, 3]) → (1, 2, 3)`

- `tuple(iterable)` – Creates a tuple from an iterable (e.g., list, string, set).

## Dictionary Methods

- `clear()` – Removes all items from the dictionary.
- `copy()` – Returns a shallow copy of the dictionary.
- `fromkeys()` – Creates a new dictionary from keys with a default value.
- `get()` – Returns the value for a key, or a default if the key is missing.
- `items()` – Returns a view of key-value pairs.
- `keys()` – Returns a view of dictionary keys.
- `values()` – Returns a view of dictionary values.
- `pop()` – Removes and returns the value of the given key.
- `popitem()` – Removes and returns the last key-value pair.
- `setdefault()` – Returns the value of a key; sets it if missing.

- `dict(iterable)` – Creates a dictionary from an iterable (e.g., a list of key-value pairs or keyword arguments).

## Sets Methods

- `update()` – Merges another dictionary into the current one.
- `add()` – Adds an element to the set.
- `clear()` – Removes all elements from the set.
- `copy()` – Returns a shallow copy of the set.
- `difference()` – Returns the difference between sets.
- `difference_update()` – Removes elements found in another set.
- `discard()` – Removes an element if present, without error.
- `intersection()` – Returns common elements between sets.
- `intersection_update()` – Updates the set with common elements.
- `isdisjoint()` – Checks if two sets have no common elements.
- `issubset()` – Checks if the set is a subset of another.
- `issuperset()` – Checks if the set is a superset of another.
- `pop()` – Removes and returns an arbitrary element.
- `remove()` – Removes a specific element, raises error if missing.
- `symmetric_difference()` – Returns elements in either set, not both.
- `symmetric_difference_update()` – Updates with elements in either set, not both.
- `union()` – Returns the union of multiple sets.
- `update()` – Adds elements from another set.

- `set(iterable)` – Creates a set from an iterable (e.g., list, tuple, string).

## Array Methods

- `append(x)` – Adds an element `x` to the end of the array.
- `extend(iterable)` – Extends the array by appending elements from an iterable.
- `insert(i, x)` – Inserts an element `x` at index `i`.
- `remove(x)` – Removes the first occurrence of `x` in the array.

- `pop(i)` – Removes and returns the element at index `i` (default is the last element).
- `index(x, start, end)` – Returns the index of the first occurrence of `x` between `start` and `end`.
- `count(x)` – Returns the number of occurrences of `x` in the array.
- `reverse()` – Reverses the order of elements in the array.
- `sort()` – Sorts the array in ascending order.
- `buffer_info()` – Returns a tuple containing memory address and the number of elements.
- `byteswap()` – Swaps the byte order of array elements.
- `fromlist(list)` – Extends the array with elements from a list.
- `tolist()` – Converts the array to a list.
- `frombytes(s)` – Appends items from a bytes object.
- `tobytes()` – Converts the array to a bytes object.
- `fromunicode(s)` – Extends the array with a Unicode string.
- `tounicode()` – Converts the array to a Unicode string.

- `array.array(typecode, iterable)` – Creates an array with elements of a specific `typecode`.