1. Conditions

Ternary Condition

```python
max_value = numbers[row][column] if numbers[row][column] > max_value else max_value
```

2. Lists

List_Operations1

```python
1   # append: Adding 10 to end of list
2   l1.append(10)
3
4   # insert: Inserting 5 at index 0
5   l1.insert(0, 5)
6
7   # extend: Adding multiple elements  [15, 20, 25] at the end
8   l1.extend([15, 20, 25])
9
10  # max: find the max value in the list
11  max_val = max(l1)
12
13  # min: find the min value in the list
14  min_val = min(l1)
15
16  # sum: Sum all items of the list
17  summed_list = sum(l1)
18
19  # ----------------------------------------------------------------------
20  l2 = [10, 20, 30, 40, 50]
21
22  # slicing
23  l2_new = l2[1:3] # l2 from index 1 to 3 (not included!), or 1 to 2 included. (= [20, 30])
24
25  # remove: Removes the first occurrence of 30
26  l2.remove(30)
27  print("After remove(30):", l2)
28
29  # pop: Removes the element at index 1 (20)
30  popped_val = l2.pop(1)
31
32  # delete: Deletes the first element (10)
33  del l1[0]
34
35  # ----------------------------------------------------------------------
36  fruits = ['orange', 'apple', 'pear', 'banana', 'kiwi', 'apple', 'banana']
37
38  # count: Return the number of occurences of an item
39  fruits.count('apple') # Output: 2
40
41  # index: Find index of next item from index 2 to index 6
42  fruits.index('apple', 3, 6)  # Output: 5
43
44  # reverse: Reverse the order of the list
45  fruits.reverse()
46
47  # sort: Sort the list A-Z or 0-9
48  fruits.sort()
49
50  # sorted: Create a NEW sorted list
51  fruits_new = fruits.sorted()
```

Get item from list

```python
1d_list | num = numbers[i]    2d_list | num = numbers[row][column]
```

List Comprehension

A short and powerful way to construct a new list.

```python
List | new_list = [item for item in list1 if (var satisfies this condition)]
```

The following list comprehension will transpose rows and columns:

```python
Input | # Input: matrix = [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]]
```

```python
Transpose | transposed = [[row[i] for row in matrix] for i in range(4)]
```

```python
Output | # Output: [[1, 5, 9], [2, 6, 10], [3, 7, 11], [4, 8, 12]]
```

Create an Empty List

Completely Empty:

```python
list = []
```

Null values in a certain length:

```python
list = [None] * length
```

Create an empty 2D list (using List Comprehension)

```python
empty_arr = [[None] * columns for i in range(rows)]
```

Useful List-Building Functions

Input_List

```python
def input_list(length = 6): # Create a user-input list
    list = [None] * length
    print(f'Forming a list. Please enter {length} numbers: ')
    for item in range(len(list)):
        list[item] = int(input())
    return list
```

Random_List

```python
def random_list(length = 6, max = 100): # Create a random list
    list = [None] * length
    for item in range(len(list)):
        list[item] = random.randint(0, max)
    return list
```

Random_into_2D_list

```python
# This function overwrites the original list!
def rng_into_2d_arr(two_d_arr): # Insert random values into en empty 2D list
    for row in range(len(arr)):
        for column in range(len(arr[row])):
            arr[row][column] = random.randint(1, 99)
```

View_as_Matrix

```python
def view_as_matrix(two_d_arr): # View a 2D List as a Matrix
    matrix = ''
    for row in range(len(two_d_arr)):
        for column in range(len(two_d_arr[row])):
            matrix += f'{str(two_d_arr[row][column]).rjust(2)} '
        matrix += '\n'
    return matrix
```

3. Tuple

Creating / Packing

```python
opt1 | tuple = num1, num2, num3    opt2 | tuple = (num1, num2)    empty | tuple = ()
```

Unpacking

Tuple_Unpacking

```python
t1 = (10, 20, 30)
num1, num2, num3 = t1
print(num1 + num2 + num3) # Output will be: 60
```

Convert from List to Tuple

```python
List_To_Tuple | tup1 = tuple(list1)
```

Main

```python
if __name__ == '__main__':
```

4. Set

Create a set

🐍 opt1 | set1 = {10, 20, 30} ; 🐍 opt2 | set1 = set(10, 20, 30) ; 🐍 empty | set1 = set()

Create a set from a list / string

🐍 set_from_list | set1 = set([2, 4, 5, 1]) 🐍 set_from_list | set1 = {[2, 4, 5, 1]}

🐍 set_from_str | set1 = set("hello")

Set Comprehension

🐍 opt1 | set1 = {i for i in range(10)} ; 🐍 opt2 | set1 = set(i for i in range(10)) - # Output will be {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

Set Operations

- **Add item:** 🐍 add_to_set | the_set.add("this", 8)
- **Remove / Pop item:** 🐍 remove_from_set | the_set.remove(8, "Yon") ; 🐍 pop_from_set | the_set.pop(index_num)
- **Get length of a set:** 🐍 len_of_set | set1_len = len(set1)
- **Intersect 2 sets:** 🐍 intersect_sets | intersected = set1 & set2
- **Union 2 sets:** 🐍 unioned_sets | unioned = set1 | set2
- **XOR 2 sets:** 🐍 xor_sets | xor_set = set1 ^ set2
- **Diff 2 sets:** 🐍 diff_sets | unioned = set1 - set2
- **Check if subset (contained):** 🐍 is_subset | print(set1 <= set2) # True/False

5. Dictionary

Create a dict

🐍 option1 | dict1 = dict(book_id = var_id, title='AOT', votes = 0)

🐍 option2 | dict1 = {"book_id":var_id, "title":"AOT", "votes":0}

🐍 option3 | dict1 = dict([("book_id", var_id), ("title","AOT"), ("votes",0)])

Get item in location

🐍 example1 | if book['genre'] == the_genre: 🐍 example2 | book['votes'] += 1

Dict's keys

🐍 get_keys | keys1 = dict1.keys()

🐍 loop_on_keys | for key in dict1.keys():

Dict's values

🐍 get_values | values1 = dict1.values()

🐍 loop_on_values | for value in dict1.values():

Dict's pairs / items

🐍 get_pairs | items1 = dict1.items()

VERY USEFUL: 🐍 loop_on_pairs | for key, value in dict1.items(): 🐍 example | for album, songs_list in LinkinPark.items():

🐍 convert_dict_to_list_of_pairs | pairs_list = list(dict1.items)

🐍 **Dictionary_Operations**

```
1   # Define a dict
2   tel = {'Sagi': 4098, 'Amit': 4139}
3
4   # Create / add an item
5   tel['Ilay'] = 4127
6
7   # Get value of a key
8   tel['Sagi'] # Output: 4098
9
10  # Delete an item
11  del tel['Amit'] # opt1
12  tel.pop('Amit') # opt2
13
14  # View the dictionary's keys
15  list(tel) # Output: ['Sagi', 'Amit', 'Ilay']
16
17  # View the dictionary's keys, sorted
18  sorted(tel) # Output: ['Amit', 'Ilay', 'Sagi']
19
20  # Check if a key exists in the dict
21  'Sagi' in tel # Output: True
22  'Amit' not in tel # Output: False
```

6. Integer

absolute

```python
to_user = (abs(user_floor - elevator_floor))
```

random

first we need to import the library:

```python
import random
bingo = random.randint(1,100)
```

7. String

String Operations

For some of these, we need to import the library:

```python
import string
```

Search

- **var.find() / var.rfind():** Searches the string for a specified value and returns the position of where it was found
- **var.index():** Searches the string for a specified value and returns the position of where it was found
- **var.count():** Returns the number of times a specified value occurs in a string

Format / Split / Replace

- **var.rjust():** `syntax | num = num.rjust(width, 'fillchar')` `example | num = num.rjust(2, '0')`
- **var.join():** `var += ''.join('Enter text here')`
- **var.partition(): / var.rpartition()** Returns a tuple where the string is parted into three parts
- **var.split() / var.rsplit():** Splits the string at the specified separator, and returns a list
- **var.splitlines():** Splits the string at line breaks and returns a list
- **var.rstrip() / var.lstrip():** Returns a right/left trim version of the string
- **var.replace:** Returns a string where a specified value is replaced with a specified value
- **remove:** `remove_ALL_OCCURENCES_of_'a' | st = st.replace('a', '')` ; `remove_2_OCCURENCES_of_'d' | st = st.replace('d', '', 2)`

Lowercase / Uppercase Conversion

- **var.upper():** Convert a string to uppercase
- **var.lower():** Convert a string to lowercase
- **var.capitalize():** Capitalizes the string. First letter is CAPITAL, rest are small letters
- **var.swapcase():** Swaps cases, lower case becomes upper case and vice versa
- **var.title():** Converts the first character of each word to upper case
- **var.casefold():** Converts string into lower case

Boolean Checks

- **var.startswith():** Returns true if the string starts with the specified value
- **var.endswith():** Returns true if the string ends with the specified value
- **var.istitle():** Returns True if the string follows the rules of a title
- **var.isalnum():** Returns True if all characters in the string are alphanumeric
- **var.isalpha():** Returns True if all characters in the string are in the alphabet
- **var.isascii():** Returns True if all characters in the string are ascii characters
- **var.isdigit():** Returns True if all characters in the string are digits
- **var.isnumeric():** Returns True if all characters in the string are numeric
- **var.isspace():** Returns True if all characters in the string are whitespaces
- **'text' in var:** Check if a letter/symbol exists in a string. returns True/False `check_sym = '@' in address`

Slicing

Slicing

```python
b = "Hello, World!"

# Get the characters from position 2 to position 5 (not included):
print(b[2:5]) # Output: "ell"

# Get the characters from position -5 to position -2 (not included):
print(b[-5:-2]) # Output: "orl"
```