List Comprehension Cheat Sheet

1. Definition

- A quicker way to create lists from any iterable (lists, range, strings)

In [1]:
```
numbers = [1,2,3,4,5]

# Make a new list which contains all of the item of numbers +1
new_nums = [nums +1 for nums in numbers]
print(new_nums)
```

```
[2, 3, 4, 5, 6]
```

In [2]:
```
result = [num for num in range(11)]
print(result)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

In [3]:
```
x = [letter for letter in "hey"]
print(x)
```

```
['h', 'e', 'y']
```

2. Comprehensions as Nested for loops

- Create a 5*5 matrix

In [5]:
```
matrix_1 = [[col for col in range(0,5)] for row in range(0,5)]
display(matrix_1)

# Print the matrix
for row in matrix_1:
    print(row)
```

```
[[0, 1, 2, 3, 4],
 [0, 1, 2, 3, 4],
 [0, 1, 2, 3, 4],
 [0, 1, 2, 3, 4],
 [0, 1, 2, 3, 4]]

[0, 1, 2, 3, 4]
[0, 1, 2, 3, 4]
[0, 1, 2, 3, 4]
[0, 1, 2, 3, 4]
[0, 1, 2, 3, 4]
```

- Typical 5*5 Matrix

In [3]:
```
matrix = []

for num in range(0,5):
    row = []
    for col in range(0,5):
        row.append(col)
    matrix.append(row)

display(matrix)

# Print the matrix
for row in matrix:
    print(row)
```

```
[[0, 1, 2, 3, 4],
 [0, 1, 2, 3, 4],
 [0, 1, 2, 3, 4],
 [0, 1, 2, 3, 4],
 [0, 1, 2, 3, 4]]

[0, 1, 2, 3, 4]
[0, 1, 2, 3, 4]
[0, 1, 2, 3, 4]
[0, 1, 2, 3, 4]
[0, 1, 2, 3, 4]
```

- Comprehension nested loop

In [7]:
```
pairs2 = [(num3, num4) for num3 in range(0,2) for num4 in range(6, 8)]
print(pairs2)
```

```
[(0, 6), (0, 7), (1, 6), (1, 7)]
```

- Typical nested for loop

In [4]:
```
pairs = [ ]

for num3 in range(0,2):
    for num4 in range(6,8):
        pairs.append((num3, num4))

print(pairs)
```

```
[(0, 6), (0, 7), (1, 6), (1, 7)]
```

---

3. Conditionals

In [15]:
```
y = [x ** 2 for x in range(10) if x % 2 == 0]
print(y)
```

```
[0, 4, 16, 36, 64]
```

In [1]:
```
y = [x ** 2 if x % 2 == 0 else 0 for x in range(10)]
print(y)
```

```
[0, 0, 4, 0, 16, 0, 36, 0, 64, 0]
```

In [6]:
```
# Create a list of strings: fellowship
fellowship = ['frodo', 'samwise', 'merry', 'aragorn', 'legolas', 'borc
```

- In the output expression, keep the string as-is if the number of characters is >= 7, else replace it with an empty string

In [7]:
```
# Create list comprehension: new_fellowship with strings with 7 charac
new_fellowship = [member if len(member) >= 7 else '' for member in fel

# Print the new list
print(new_fellowship)
```

```
['', 'samwise', '', 'aragorn', 'legolas', 'boromir', '']
```

In [14]:
```
# Create list comprehension: new_fellowship with strings with 7 charac
new_fellowship = [member for member in fellowship if len(member) >= 7]

# Print the new list
print(new_fellowship)
```

```
['samwise', 'aragorn', 'legolas', 'boromir']
```

---

4. Dictionaries Comprehensions

- Create new dictionaries from iterables

- Curly braces {} instead of []
- Key and value are seperated by a colon in the output expression

In [18]:
```
# Create a list of strings: fellowship
fellowship = ['frodo', 'samwise', 'merry', 'aragorn', 'legolas', 'borc
```

In [19]:
```
# Create dict comprehension: new_fellowship
new_fellowship = {member: len(member) for member in fellowship}

# Print the new dictionary
print(new_fellowship)
```

```
{'frodo': 5, 'samwise': 7, 'merry': 5, 'aragorn': 7, 'legolas': 7, 'b
oromir': 7, 'gimli': 5}
```

In [20]:
```
pos_neg = {num: -num for num in range(9)}
display(pos_neg)
```

```
{0: 0, 1: -1, 2: -2, 3: -3, 4: -4, 5: -5, 6: -6, 7: -7, 8: -8}
```

5. Comprehensions With Zip

In [22]:
```
lists = [(x, y) for x, y in zip(range(1,11), range(11,21))]
print(lists)
```

```
[(1, 11), (2, 12), (3, 13), (4, 14), (5, 15), (6, 16), (7, 17), (8, 1
8), (9, 19), (10, 20)]
```