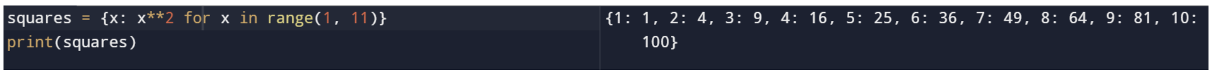
Dictionary Comprehension

1. Squares of Numbers:

Create a dictionary with numbers from 1 to 10 as keys and their squares as values using dictionary comprehension.

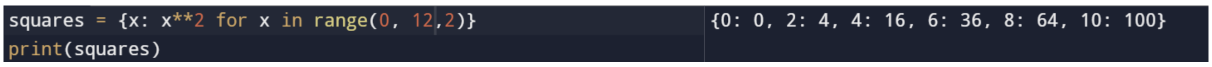


```
squares = {x: x**2 for x in range(1, 11)}
print(squares)
```

```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10:
    100}
```

2. Filter Even Numbers:

Create a dictionary using dictionary comprehension where the keys are numbers from 1 to 10, and the values are their squares, but include only the even numbers.

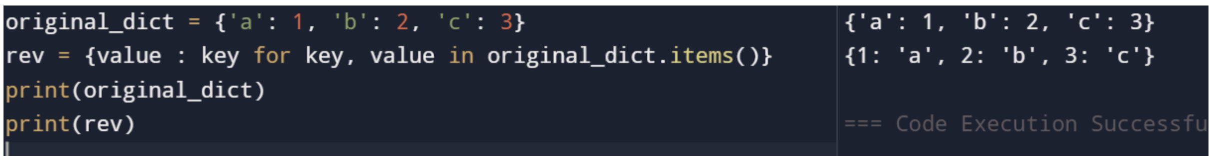


```
squares = {x: x**2 for x in range(0, 12,2)}
print(squares)
```

```
{0: 0, 2: 4, 4: 16, 6: 36, 8: 64, 10: 100}
```

3. Reverse a Dictionary:

Reverse the keys and values of this dictionary using dictionary comprehension:
original_dict = {'a': 1, 'b': 2, 'c': 3}

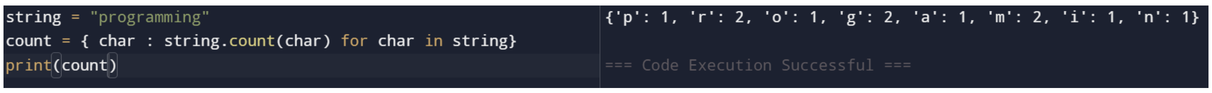


```
original_dict = {'a': 1, 'b': 2, 'c': 3}
rev = {value : key for key, value in original_dict.items()}
print(original_dict)
print(rev)
```

```
{'a': 1, 'b': 2, 'c': 3}
{1: 'a', 2: 'b', 3: 'c'}

=== Code Execution Successfu
```

4. Count Character Frequency:

Write a dictionary comprehension to count the frequency of each character in the string "programming".

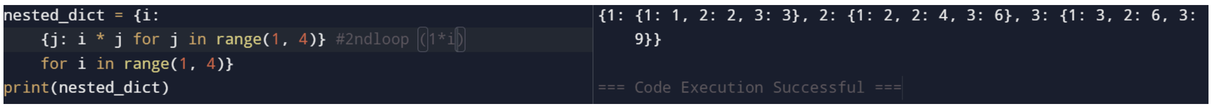


```
string = "programming"
count = { char : string.count(char) for char in string}
print(count)
```

```
{'p': 1, 'r': 2, 'o': 1, 'g': 2, 'a': 1, 'm': 2, 'i': 1, 'n': 1}

=== Code Execution Successful ===
```

5. Nested Dictionary:

Use dictionary comprehension to create a nested dictionary where the keys are numbers from 1 to 3, and the values are dictionaries that map numbers from 1 to 3 to their products.
Example: {1: {1: 1, 2: 2, 3: 3}, 2: {1: 2, 2: 4, 3: 6}, 3: {1: 3, 2: 6, 3: 9}}

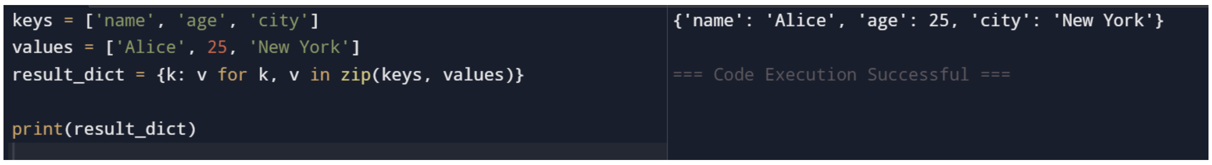


```
nested_dict = {i:
    {j: i * j for j in range(1, 4)} #2ndloop (1*i)
    for i in range(1, 4)}
print(nested_dict)
```

```
{1: {1: 1, 2: 2, 3: 3}, 2: {1: 2, 2: 4, 3: 6}, 3: {1: 3, 2: 6, 3:
    9}}

=== Code Execution Successful ===
```

6. Zip Two Lists into a Dictionary:

Use dictionary comprehension to create a dictionary from these two lists:
keys = ['name', 'age', 'city']
values = ['Alice', 25, 'New York']

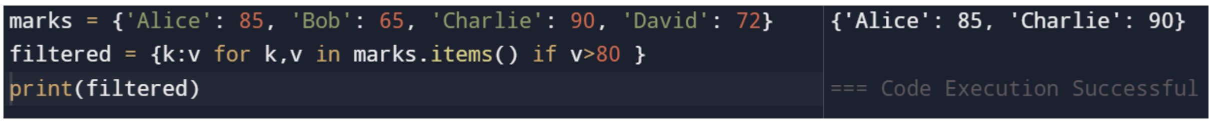


```
keys = ['name', 'age', 'city']
values = ['Alice', 25, 'New York']
result_dict = {k: v for k, v in zip(keys, values)}

print(result_dict)
```

```
{'name': 'Alice', 'age': 25, 'city': 'New York'}

=== Code Execution Successful ===
```

7. Filter Dictionary by Value:

Given a dictionary marks = {'Alice': 85, 'Bob': 65, 'Charlie': 90, 'David': 72}, create a new dictionary containing only students who scored more than 80.

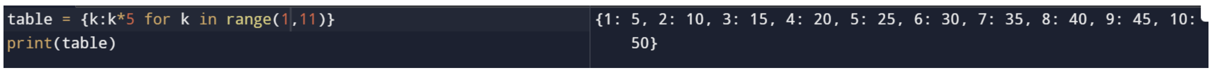


```
marks = {'Alice': 85, 'Bob': 65, 'Charlie': 90, 'David': 72}
filtered = {k:v for k,v in marks.items() if v>80 }
print(filtered)
```

```
{'Alice': 85, 'Charlie': 90}

=== Code Execution Successful
```

8. Multiplication Table:

Create a dictionary comprehension to generate a multiplication table for the number 5 (from 1 to 10).
Example: {1: 5, 2: 10, 3: 15, ..., 10: 50}

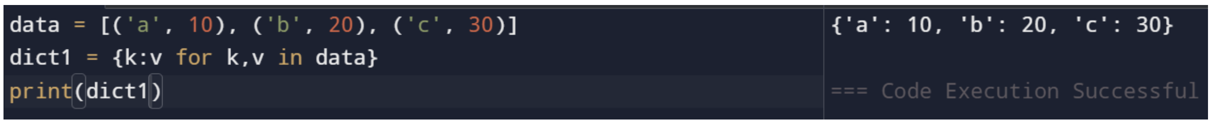


```
table = {k:k*5 for k in range(1,11)}
print(table)
```

```
{1: 5, 2: 10, 3: 15, 4: 20, 5: 25, 6: 30, 7: 35, 8: 40, 9: 45, 10:
    50}
```

9. Convert List to Dictionary:

Given a list of tuples data = [('a', 10), ('b', 20), ('c', 30)], convert it into a dictionary using dictionary comprehension.



```python
data = [('a', 10), ('b', 20), ('c', 30)]
dict1 = {k:v for k,v in data}
print(dict1)
```

```
{'a': 10, 'b': 20, 'c': 30}

=== Code Execution Successful
```