9. Comprehension

October 18, 2022

1 Comprehension

- A comprehension is a compact way of creating a Python data structure from one or more iterators.

1.1 List Comprehension

- Write a program to create list of squares of numbers from 1 to 10

[1]:
```
# program to generate list of squares of first 10 natural numbers

squares = []
for i in range(1,11):
    squares.append(i**2)
```

[2]:
```
squares
```

[2]:
```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

[4]:
```
# program to generate list of squares of first 10 natural numbers
# using list comprehension
squares_2 = [i**2 for i in range(1,11)]
```

[5]:
```
squares_2
```

[5]:
```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

- Write a program to create list of squares of **even** numbers from 1 to 10

[6]:
```
sq_even = []
for i in range(1,11):
    if i%2 == 0:
        sq_even.append(i**2)
```

[7]:
```
sq_even
```

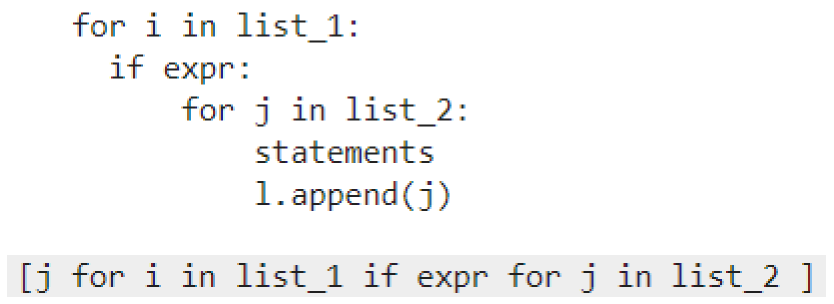
[7]:
```
[4, 16, 36, 64, 100]
```

[9]:
```
sq_even_C = [i**2 for i in range(1,11) if i%2 == 0]
```

[10]:
```
sq_even_C
```

[10]:
```
[4, 16, 36, 64, 100]
```

- Comprension for nested loops



```
for i in list_1:
  if expr:
      for j in list_2:
          statements
          l.append(j)

[j for i in list_1 if expr for j in list_2 ]
```

1.2 Tuple Comprehension

[11]:
```
squaresT = tuple(i**2 for i in range(1,11))
```

[12]:
```
squaresT
```

[12]:
```
(1, 4, 9, 16, 25, 36, 49, 64, 81, 100)
```

1.3 Dictionary Comprehension

- Generate a dictionary of squares of first 10 natural numbers, where key is the natural number and value is its square

[13]:
```
squares = {x : x*x for x in range(1,11)}
```

[14]:
```
squares
```

[14]:
```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100}
```

1.4 Set comprehension

[15]:
```
input_list = [1, 2, 3, 4, 4, 5, 6, 6, 6, 7, 7]
```

[16]:
```
# create set of even numbers from given list of numbers
set_using_comp = {var for var in input_list if var % 2 == 0}
```

[17]:
```
set_using_comp
```

[17]:
```
{2, 4, 6}
```

2 enumerate

- A lot of times when dealing with iterators, we also get a need to keep a count of iterations.
- Python eases the programmers' task by providing a built-in function enumerate() for this task.
- `enumerate()` method adds a counter to an iterable and returns it in a form of enumerate object.
- This enumerate object can then be used directly in for loops.

[18]:
```
list_colors = ["red", "green",'blue','pink']
```

[20]:
```
for i in enumerate(list_colors):
    print(i)
    print("----")
```

```
(0, 'red')
----
(1, 'green')
----
(2, 'blue')
----
(3, 'pink')
----
```