Comprehensions_and_Generators

July 18, 2024

1 Comprehensions and Generators

1. Write a list comprehension which returns the square of each number from 0 to 15

[1]:
```
[x**2 for x in range(0,15)]
```

[1]:
```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196]
```

2. The expression `x%4==0` will return `True` if 4 is a factor of `x`. Write a list comprehension which will return numbers from 0 to 20 which have 4 as a factor.

[3]:
```
[x for x in range(0,21) if x%4 == 0]
```

[3]:
```
[0, 4, 8, 12, 16, 20]
```

3. Write a generator function which will return the string 'foo' every fifth call, and 'moo' otherwise.

[7]:
```
def foo_moo():
    i = 0
    while True:
        if i%5==0:
            yield print('foo')
        else:
            yield print('moo')
        i += 1

fm = foo_moo()

for _ in range(11):
    if next(fm) != None:
        print(next(fm))
```

```
foo
moo
moo
moo
moo
foo
```

```
moo
moo
moo
moo
foo
```

4. A comprehension using curly brackets will produce a dictionary. Set the variable 'name' to your name and use it to create a dictionary

[27]:
```
name = 'Tanaphat'
items = zip(list(name), list(range(len(name))))

{x:y for x,y in items}
```

[27]:
```
{'T': 0, 'a': 6, 'n': 2, 'p': 4, 'h': 5, 't': 7}
```

last index position

index position

[ ]:

[ ]: