Python One-Liners

By:
Sandeep (GU-2021-4204)
Shubham (GU-2021-4208)

Introduction

Why use one-liners and what are they?

Python is a widely-used general-purpose, high-level programming language. Python programs generally are smaller than other programming languages like Java. Programmers have to type relatively less and indentation requirements of the language makes them readable all the time. However, Python programs can be made more concise using some one-liner codes. These can save time by having less code to type

Topics to be covered:

- Value Swapping
- Comparing Elements
- Lambda Functions(Anonymous Functions)
- Filter & Map Function
- List Comprehension
- BONUS

Value Swapping

Traditional way to swap value of two or more variables was to re-assign them individually, But there’s a easy way to do same:

```
a = 1

b = 2

a , b = b , a      or      a , b = 2 , 1
```

We can also assign more than two at once

```
a, b, c, d =1, 3, 5, 7
```

Comparing Elements

Comparing if m is 1 or 2 or 3

```
if m == 1 or m == 2 or m == 3:
    pass
if m in [1, 2, 3]:
    pass
```

Lambda Function

Lambda functions are python one liner functions and are often used when an expression is to be evaluated.

For example, let’s suppose we want to create a function that returns the square of the number passed as argument. The normal way of doing this v/s doing the same with lambda function:

```
def sqr(x):
   return x * x

print(sqr(5))
```

```
sqr = lambda x: x * x

print(sqr(5))
```

Filter & Map Function

The filter() function extracts elements from an iterable (list, tuple etc.) for which a function returns True.

- Returns which function returns True
- Only iterate through one iterable.

Eg. return even numbers from list from a list.

```
en =[x for x in range(11) if x % 2 == 0]
print(en)
```

The map() function applies a given function to each item of an iterable (list, tuple etc.) and returns an iterator.

- Returns all values.
- Can iterate more than one iterable.

Eg. return square/cube of numbers from a list

```
sqr = lambda x: x * x
print(sqr(5))
```

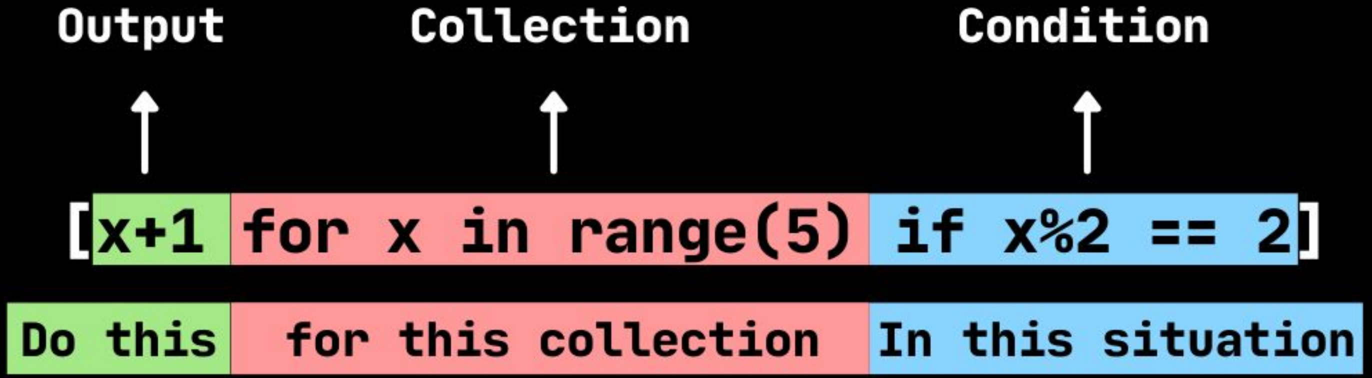
List Comprehension

List comprehension in Python is an easy and compact syntax for creating a list from a string or another list.

It is a very concise way to create a new list by performing an operation on each item in the existing list.

List comprehension is considerably faster than processing a list using the for loop.

LIST COMPREHENSION



Output

Collection

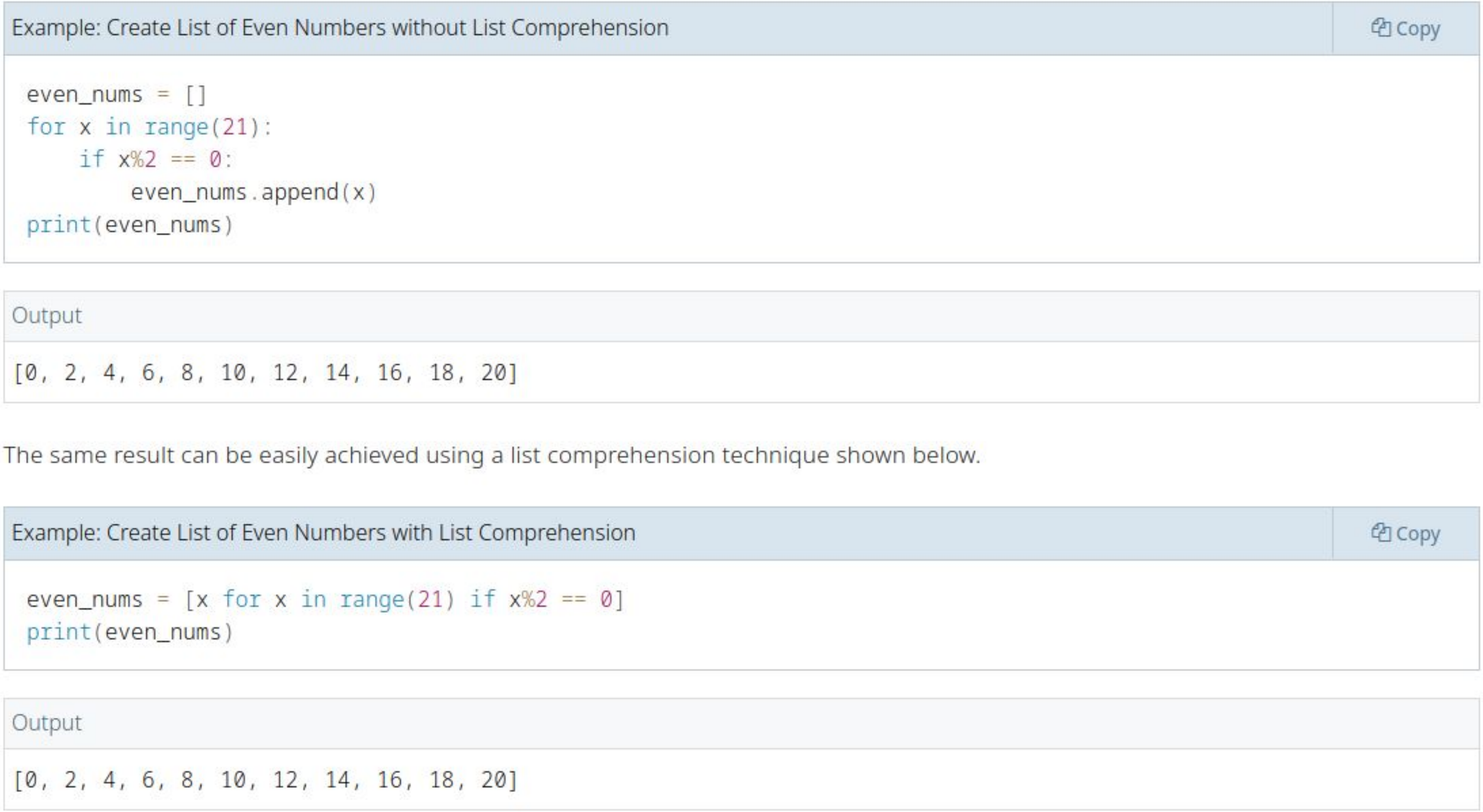
Condition

```
[x+1 for x in range(5) if x%2 == 2]
```

Do this

for this collection

In this situation



Example: Create List of Even Numbers without List Comprehension

```
even_nums = []
for x in range(21):
    if x%2 == 0:
        even_nums.append(x)
print(even_nums)
```

Output

```
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

The same result can be easily achieved using a list comprehension technique shown below.

Example: Create List of Even Numbers with List Comprehension

```
even_nums = [x for x in range(21) if x%2 == 0]
print(even_nums)
```

Output

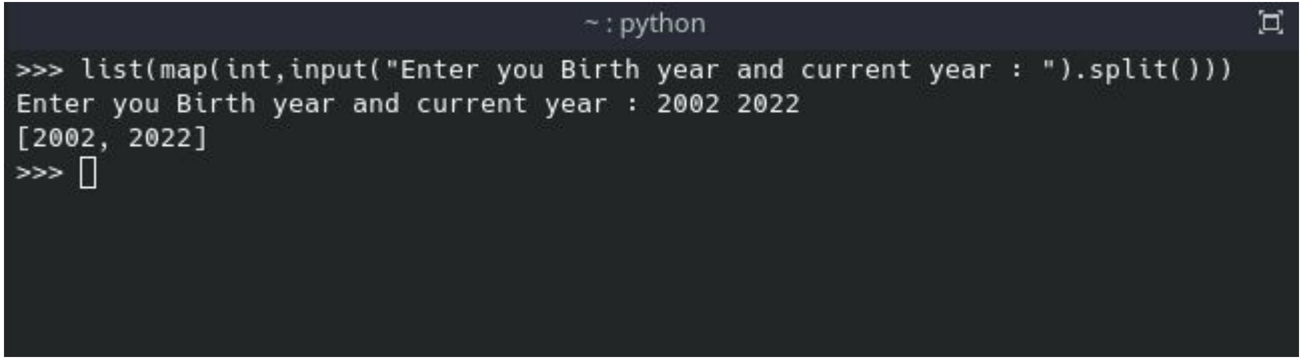
```
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

BONUS

Take multiple input at once and store it in the form of list. Later access/use them using indexing(but you need to know prior datatype of input)

```
Variable = list ( map ( <datatype> , input().split() ) )
```

```
Input = list ( map ( int , input().split() ) )
```



```
>>> list(map(int,input("Enter you Birth year and current year : ").split()))
Enter you Birth year and current year : 2002 2022
[2002, 2022]
>>> 
```

TANK YOU