



Python Quick Tips

Loops (for/while)



Python Quick Tips #8

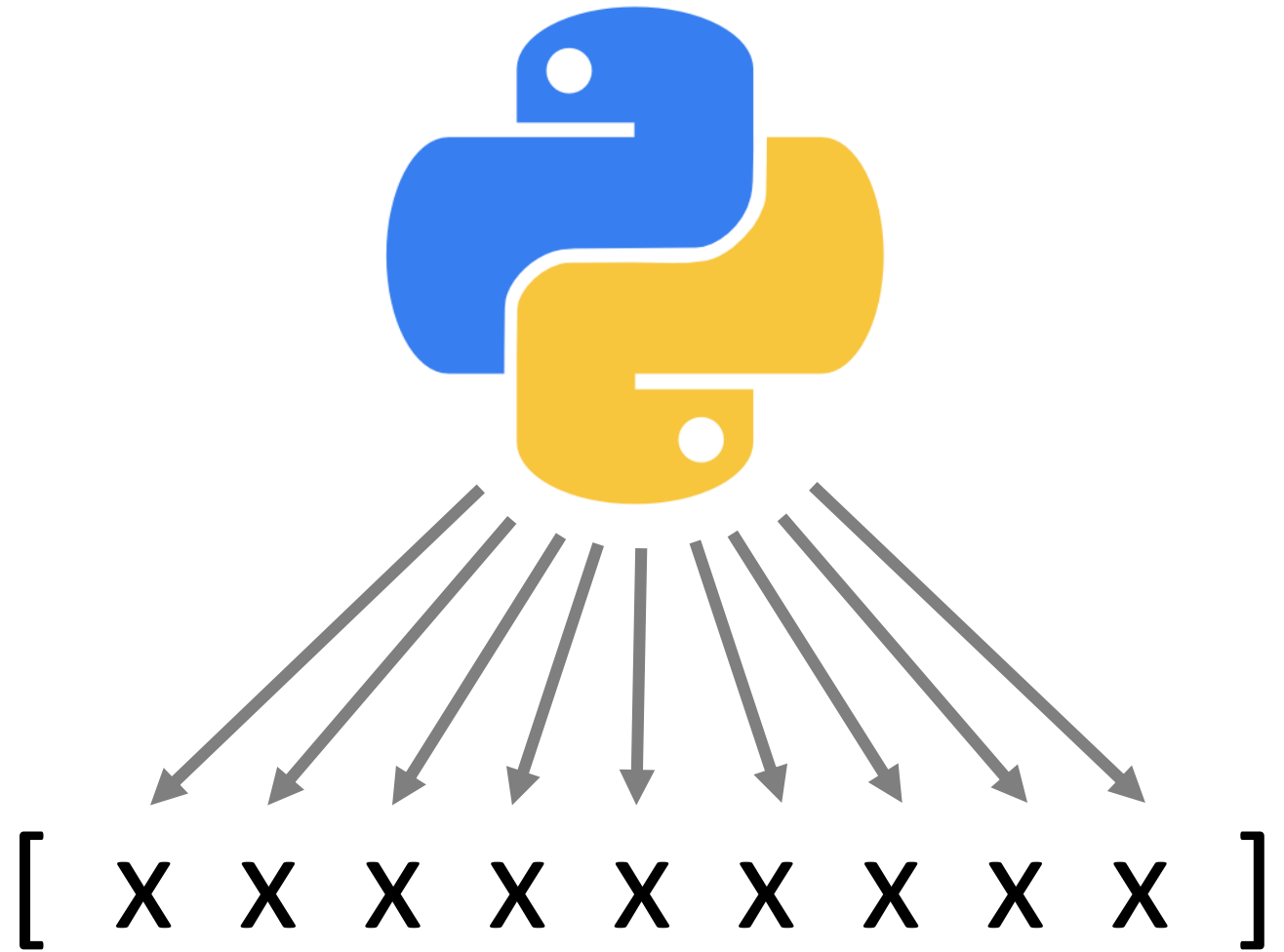
Loops (for/while)

Iteration



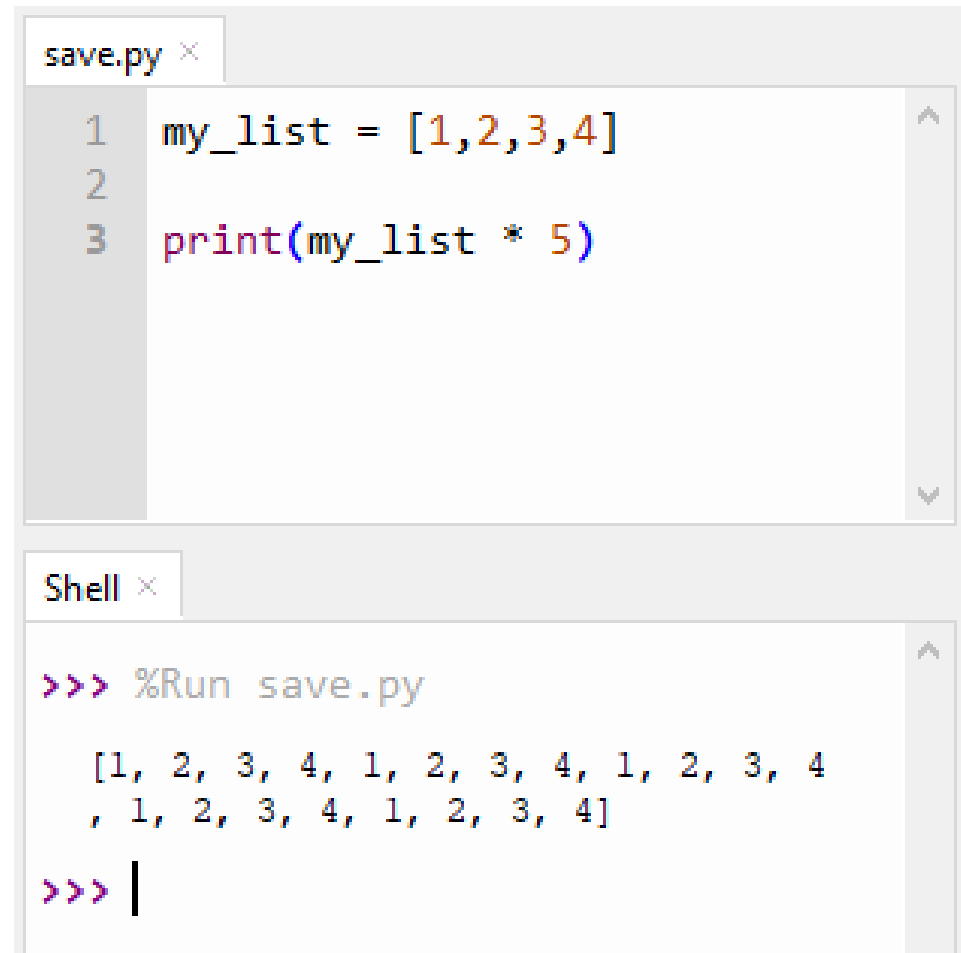
The **repetition** of a
process to generate an
outcome

Loops



Why use Loops

Usually, lists



The screenshot shows a Python IDE with two panels. The top panel, titled 'save.py', contains three lines of code: `1 my_list = [1,2,3,4]`, `2` (blank), and `3 print(my_list * 5)`. The bottom panel, titled 'Shell', shows the execution of the script with the command `>>> %Run save.py`. The output is a list repeated five times: `[1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4]`. The prompt `>>> |` is visible at the bottom of the shell.

```
save.py ×  
1 my_list = [1,2,3,4]  
2  
3 print(my_list * 5)  
  
Shell ×  
  
>>> %Run save.py  
  
[1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4]  
  
>>> |
```

For (Loop)

Syntax

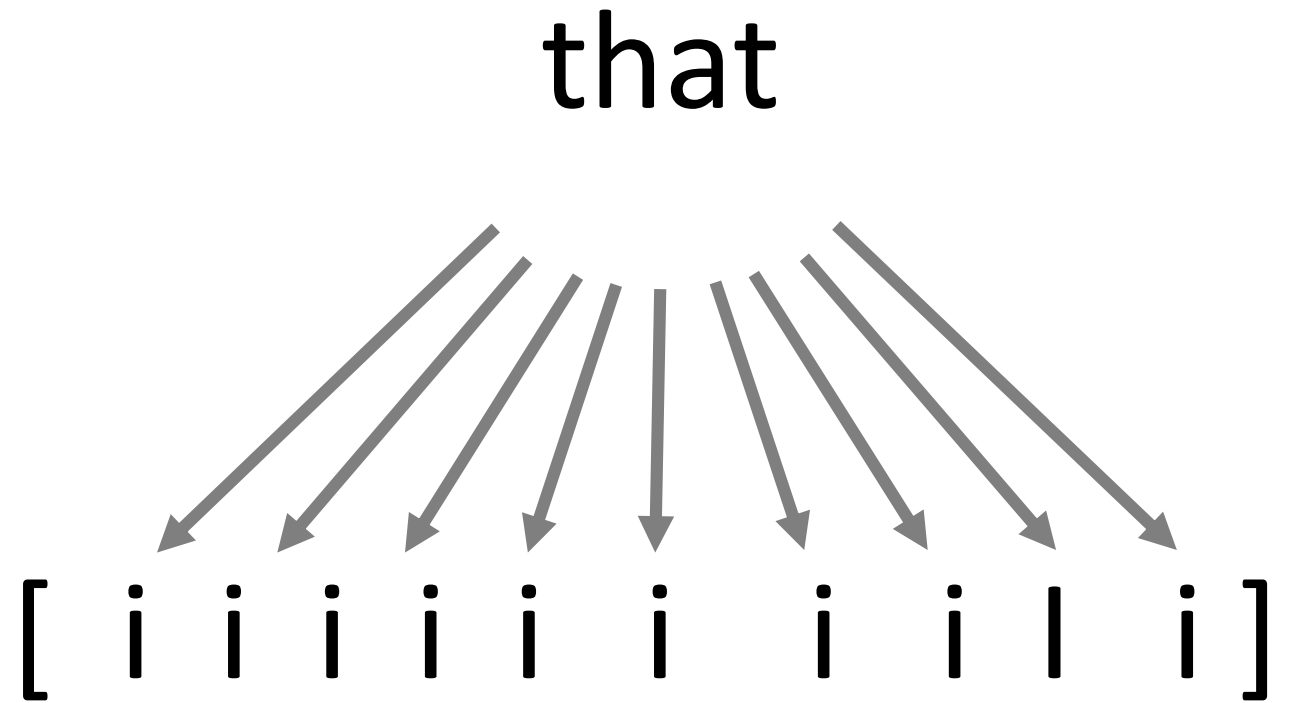
for i in that:

(tab)code

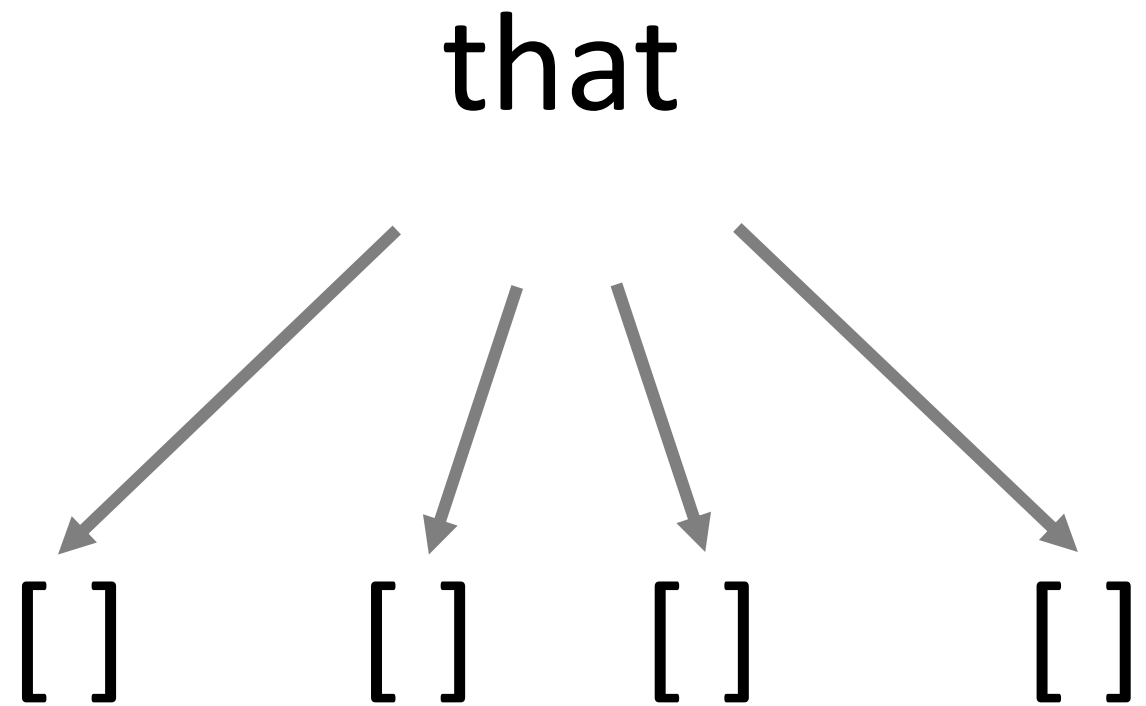
‘i’ can be used as a local variable in the loop.

‘i’ can be anything.

What it
means



Nested Lists



Example (List)

save.py ×

```
1 my_list = [1,2,3,4]
2
3 for i in my_list:
4     print(i)
```

Shell ×

```
>>> %Run save.py
```

```
1
2
3
4
```

Example (String)

save.py ×

```
1 my_string = 'batman'  
2  
3 for i in my_string:  
4     print(i)
```

Shell ×

```
>>> %Run save.py
```

```
b  
a  
t  
m  
a  
n
```

Example (Range)

save.py ×

```
1 my_range = range(1,10,1)
2
3 for i in my_range:
4     print(i)
```

Shell ×

```
>>> %Run save.py
```

```
1
2
3
4
5
6
7
8
9
```

Iterate to a
new list

In each case, our
output is **being printed**.

We can define an
empty list (`[]`) and use
the **.append** method to
rebuild a list.

Example

save.py ×

```
1 my_range = range(1,10,1)
2
3 my_list = []
4
5 for i in my_range:
6     my_list.append(i)
7
8 print(my_range)
9 print(my_list)
10 print(type(my_range))
11 print(type(my_list))
```

Shell ×

```
>>> %Run save.py
```

```
range(1, 10)
[1, 2, 3, 4, 5, 6, 7, 8, 9]
<class 'range'>
<class 'list'>
```

Looping a definition

save.py ×

```
1 def power(x,y=2):  
2     return x**y  
3  
4 my_range = range(1,10,1)  
5  
6 my_list = []  
7  
8 for i in my_range:  
9     val = power(i, 3)  
10    my_list.append(val)  
11  
12 print(my_range)  
13 print(my_list)  
14 print(type(my_range))  
15 print(type(my_list))
```

Shell ×

```
>>> %Run save.py
```

```
range(1, 10)  
[1, 8, 27, 64, 125, 216, 343, 512, 729]  
<class 'range'>  
<class 'list'>
```

Looping an If statement

save.py ×

```
1 my_range = range(1,10,1)
2 divisor = 3
3
4 is_mult = []
5 isnt_mult = []
6
7 for i in my_range:
8     if i%divisor == 0:
9         is_mult.append(i)
10    else:
11        isnt_mult.append(i)
12
13 print(is_mult)
14 print(isnt_mult)
```

Shell ×

```
>>> %Run save.py
```

```
[3, 6, 9]
```

```
[1, 2, 4, 5, 7, 8]
```

Looping with a Try/Except

```
save.py ×  
1 my_list = [1, '5', 2.5, 'test']  
2  
3 results = []  
4  
5 for i in my_list:  
6     try:  
7         val = i+5  
8     except:  
9         val = 0  
10    finally:  
11        results.append(val)  
12  
13 print(results)
```

```
Shell ×  
  
>>> %Run save.py  
  
[6, 0, 7.5, 0]
```


While (Loop)

Syntax

```
while condition:  
    (tab)code
```

Be careful – it can go
on forever!

Example

```
save.py ×  
1 start = 10  
2 divisor = 2  
3 end = 1  
4  
5 current = start  
6 results = []  
7  
8 while current > end:  
9     results.append(current)  
10     current = current/divisor  
11  
12 print(results)  
  
Shell ×  
  
>>> %Run save.py  
[10, 5.0, 2.5, 1.25]
```

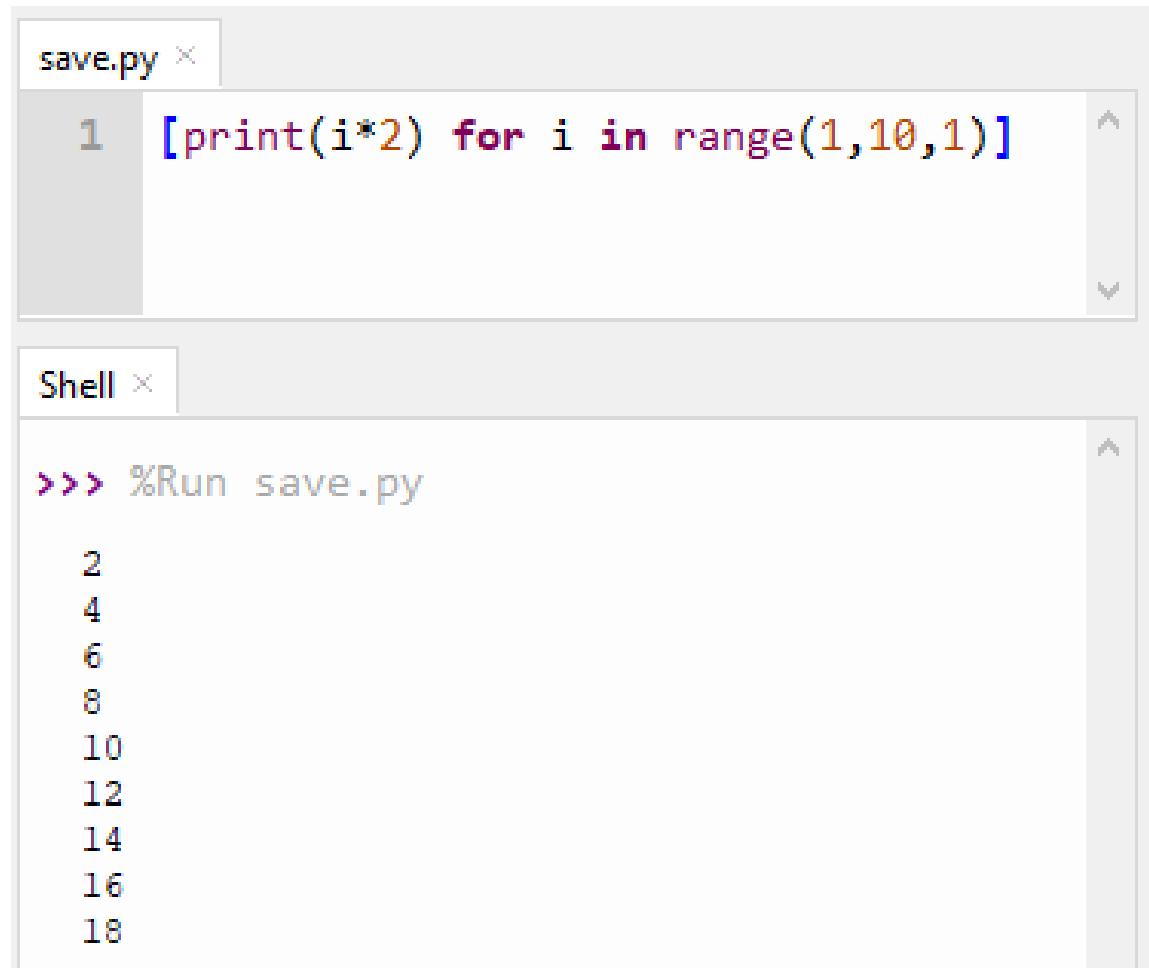
Shorthand (For Loop)

Syntax

[do this **for** i **in** this]

Acts as a contained **list**
comprehension process
(builds a list).

Example



The image shows a Jupyter Notebook interface with two panels. The top panel, titled 'save.py', contains a single line of Python code: `[print(i*2) for i in range(1,10,1)]`. The bottom panel, titled 'Shell', shows the command `>>> %Run save.py` being executed, followed by the output of the code: the numbers 2, 4, 6, 8, 10, 12, 14, 16, and 18, each on a new line.

```
save.py ×  
1 [print(i*2) for i in range(1,10,1)]  
  
Shell ×  
>>> %Run save.py  
  
2  
4  
6  
8  
10  
12  
14  
16  
18
```



Next on #9

Zip Iteration



Python Quick Tips

Loops (for/while)