Python Quick Tips

Loops (for/while)

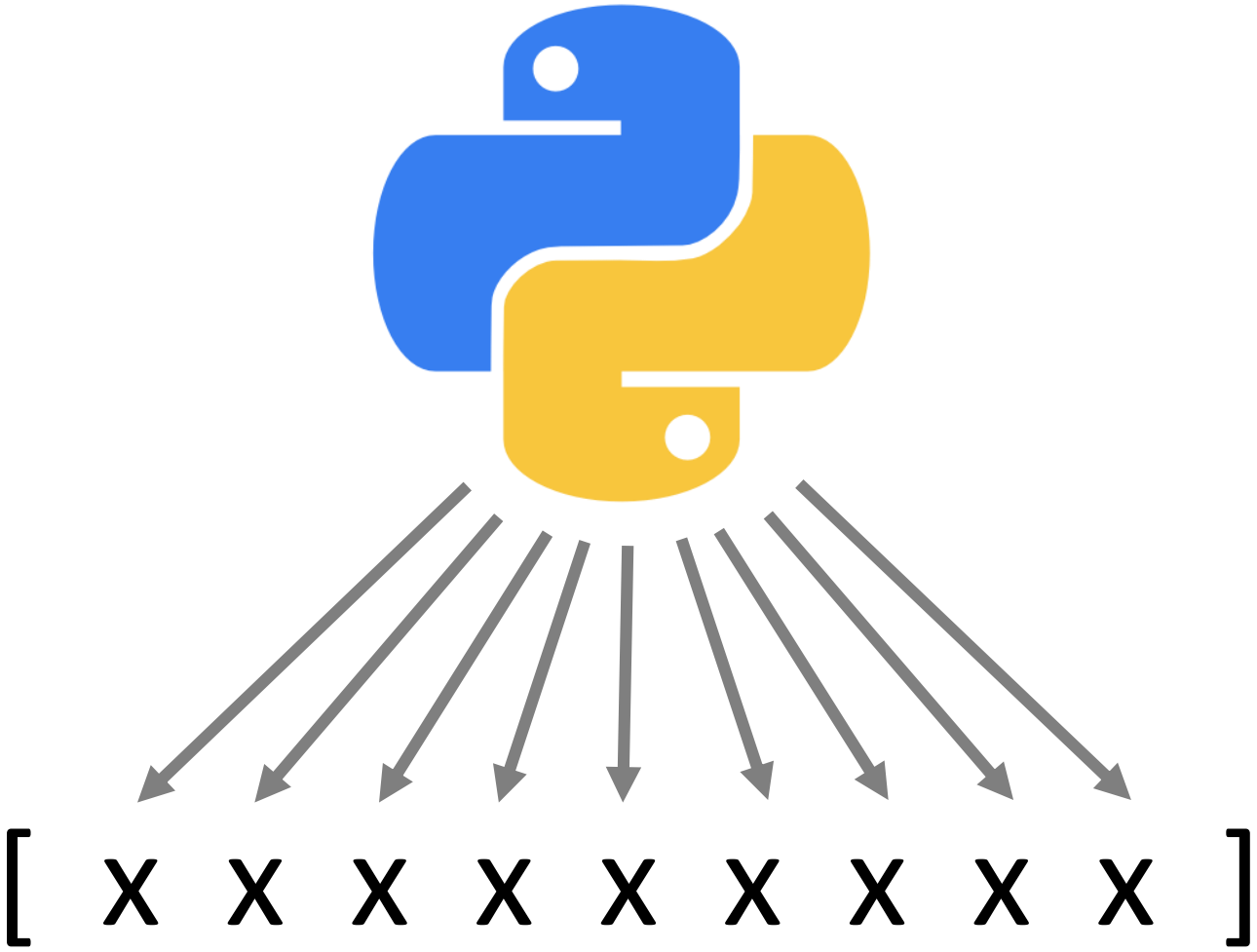
Python Quick Tips #8

Loops (for/while)

Iteration

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

Loops



Why use Loops

Usually, lists

save.py

```
my_list = [1,2,3,4]

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:

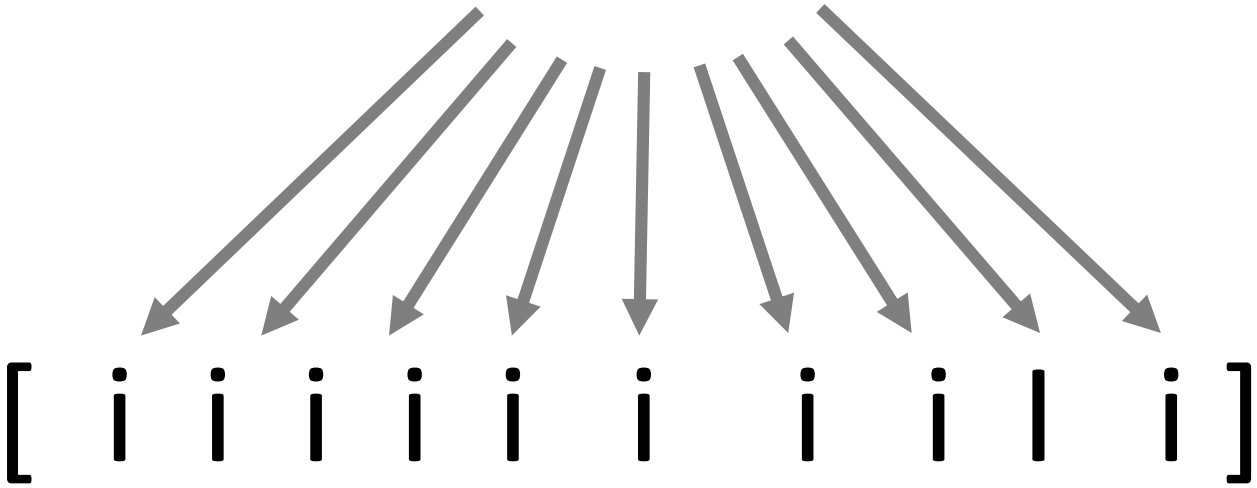
(tab)code

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

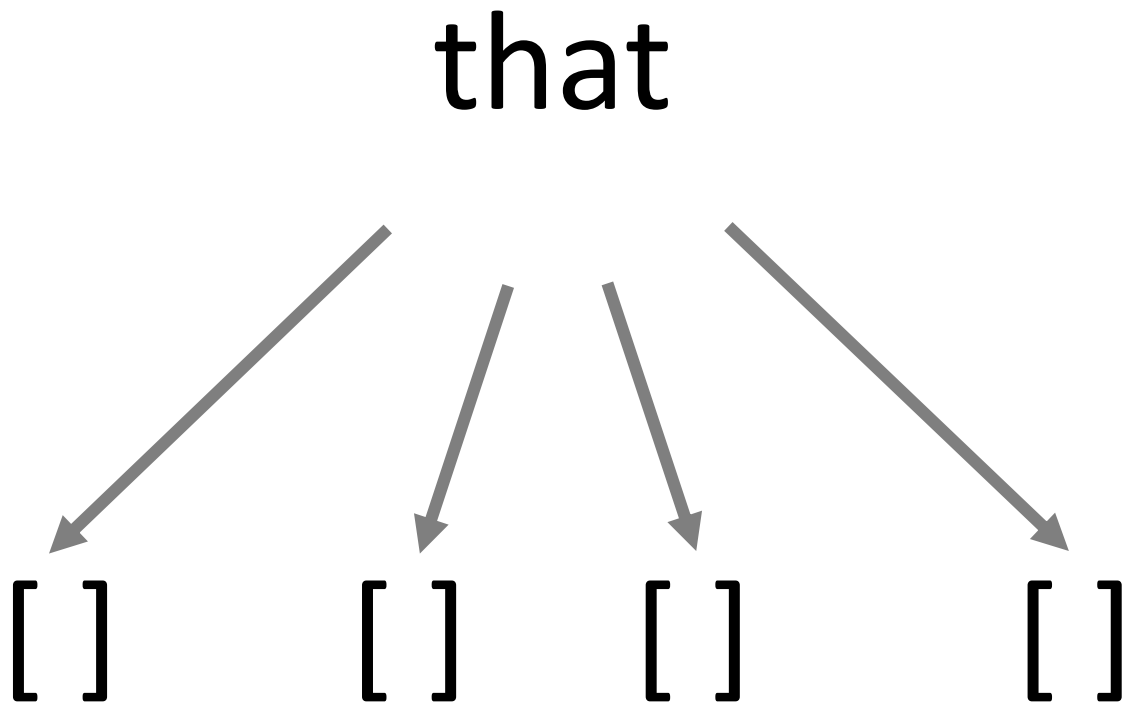
‘i’ can be anything.

What it means

that



Nested Lists



Example (List)

save.py

```python
my_list = [1,2,3,4]

for i in my_list:
    print(i)
```

Shell

```
>>> %Run save.py

  1
  2
  3
  4
```

Example (String)

save.py

```python
my_string = 'batman'

for i in my_string:
    print(i)
```

Shell

```
>>> %Run save.py

  b
  a
  t
  m
  a
  n
```

Example (Range)

save.py

```python
my_range = range(1,10,1)

for i in my_range:
    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

```python
my_range = range(1,10,1)

my_list  = []

for i in my_range:
    my_list.append(i)

print(my_range)
print(my_list)
print(type(my_range))
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

```python
def power(x,y=2):
    return x**y

my_range = range(1,10,1)

my_list  = []

for i in my_range:
    val = power(i, 3)
    my_list.append(val)

print(my_range)
print(my_list)
print(type(my_range))
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

```python
my_range = range(1,10,1)
divisor  = 3

is_mult   = []
isnt_mult = []

for i in my_range:
    if i%divisor == 0:
        is_mult.append(i)
    else:
        isnt_mult.append(i)

print(is_mult)
print(isnt_mult)
```

Shell

```
>>> %Run save.py

  [3, 6, 9]
  [1, 2, 4, 5, 7, 8]
```

Looping with a Try/Except

save.py

```python
my_list = [1, '5', 2.5, 'test']

results = []

for i in my_list:
    try:
        val = i+5
    except:
        val = 0
    finally:
        results.append(val)

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

```python
start   = 10
divisor = 2
end     = 1

current = start
results = []

while current > end:
    results.append(current)
    current = current/divisor

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

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)