PYTHON FOR R USERS

Control flow, loops

Daniel Chen
Instructor

Whitespace matters

Whitespace and indentation is a fundamental part of writing python code and is not optional

Control Flow

R

```
> if (5 == 5) {
+     print('TRUE')
+ }
[1] "TRUE"
```

PYTHON

```
In [4]: if 5 == 5:
   ...:     print('True')
   ...:
True
```

if-elif-else

R

```
> val <- 2

> if (val == 1) {
+     print('snap')
+ } else if (val == 2) {
+     print('crackle')
+ } else {
+     print('pop')
+ }
[1] "crackle"
```

PYTHON

```
In [2]: val = 2

In [3]: if val == 1:
   ...:     print('snap')
   ...: elif val == 2:
   ...:     print('crackle')
   ...: else:
   ...:     print('pop')
   ...:
crackle
```

Loops

R

```r
num_val <- c(1, 2, 3, 4)

for (value in seq_along(num_val)) {
    print(value)
}
```

PYTHON

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

for value in num_val:
    print(value)
```

PYTHON FOR R USERS

Let's practice!

PYTHON FOR R USERS

Functions

Daniel Chen
Instructor

Simple function

R

```
> my_mean <- function(x, y) {
+     num = x + y
+     dem = 2
+     num / dem
+ }
> my_mean(10, 20)
[1] 15
```

PYTHON

```
In [5]: def my_mean(x, y):
   ...:     num = x + y
   ...:     dem = 2
   ...:     return num / dem
   ...:
   ...: my_mean(10, 20)
   ...:
Out[5]: 15.0
```

Reusing Functions

R

```
> my_sq <- function(x) {
+     x^2
+ }
> my_sq_mean <- function(y, z) {
+     (my_sq(y) + my_sq(z)) / 2
+ }
> my_sq_mean(10, 12)
[1] 122
```

PYTHON

```
In [6]: def my_sq(x):
   ...:     return x ** 2
   ...: def my_sq_mean(y, z):
   ...:     return (my_sq(y) + my_sq(z)) / 2
   ...:
   ...: my_sq_mean(10, 12)
   ...:
Out[6]: 122.0
```

Lambda Functions

R

```
> add_1 <- function(x) x + 1

> function(x) x + 1
```

PYTHON

```
In [7]: def add_1(x):
   ...:     return x + 1
   ...:
In [8]: a1_lam = lambda x: x + 1
   ...: a1_lam(3)
   ...:
Out[8]: 4
```

PYTHON FOR R USERS

Let's practice!

PYTHON FOR R USERS

Comprehensions

Daniel Chen
Instructor

Comprehensions are loops

- iterate (loop) through a list
- perform some function
- append results into a new list

List Comprehension

LOOP

```
In [1]: data = [1, 2, 3, 4, 5]
   ...: new = []
   ...:
   ...: for x in data:
   ...:     new.append(x**2)
   ...: print(new)
   ...:
[1, 4, 9, 16, 25]
```

COMPREHENSION

```
In [3]: data = [1, 2, 3, 4, 5]

In [4]: new = [x**2 for x in data]
   ...: print(new)
   ...:
[1, 4, 9, 16, 25]
```

Dictionary Comprehension

LOOP

```
In [1]: data = [1, 2, 3, 4, 5]
   ...: new = {}
   ...:
   ...: for x in data:
   ...:     new[x] = x**2
   ...: print(new)
   ...:
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

COMPREHENSION

```
In [3]: data = [1, 2, 3, 4, 5]

In [4]: new = {x: x**2 for x in data}
   ...: print(new)
   ...:
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

Alternatives to for loop

- for loop to iterate
- R: sapply, lapply, apply functions
- Python: map function and apply method

Map

FOR LOOP

```
In [1]: def sq(x):
   ...:     return x**2
   ...:
   ...: l = [1, 2, 3]
   ...:
   ...: for i in l:
   ...:     print(sq(i))
   ...:
1
4
9
```

MAP

```
In [2]: map(sq, l)
Out[2]: <map at 0x7f41cca38358>

In [3]: list(map(sq, l))
Out[3]: [1, 4, 9]
```

PYTHON FOR R USERS

Let's practice!