Effective Programming Practices for Economists

Basic Python

Comprehensions

Janoś Gabler and Hans-Martin von Gaudecker

Common loop patterns

The mapping loop

```
>>> squares = []
>>> for i in [1, 2, 3, 4, 5]:
...     squares.append(i ** 2)
>>> squares
[1, 4, 9, 16, 25]
```

The filtering loop

```
>>> even = []
>>> for i in range(10):
...     if i % 2 == 0:
...         even.append(i)
[0, 2, 4, 6, 8]
```

- Both initialize and append
- At least three lines
- Can lead to deep indentation

List comprehensions

Set notation

$$\{x^2 | x \in \{1, 2, 3, 4, 5\}\}$$

$$\{x | x \in \{0, 1, ..., 9\}, x \mod 2 = 0\}$$

List comprehension

```
>>> [i ** 2 for i in [1, 2, 3, 4, 5]]
[1, 4, 9, 16, 25]

>>> [i for i in range(10) if i % 2 == 0]
[0, 2, 4, 6, 8]

>>> [i if i % 2 == 0 else 0 for i in range(10)]
[0, 0, 2, 0, 4, 0, 6, 0, 8, 0]
```

- List comprehensions are inspired by set notation
- Can call arbitrary functions
- More readable than loops as long as it fits on one line!
- Not much faster than loops

Dict comprehension

```
>>> {i: i ** 2 for i in [1, 2, 3, 4, 5]}
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25}

>>> skills = {
...     "Raymond": 8,
...     "Guido": 10,
...     "Tim": 9,
... }
>>> {k: v for k, v in skills.items() if v >= 9}
{'Guido': 10, 'Tim': 9}
```

- Inside a dict comprehension you can loop over any iterable
- More readable than loops if it fits on one line

When to use

Speed

- Comprehensions are a few percent faster than for loops
- Vectorization can be 100 x faster than for loops

Readability

- Comprehensions are more readable if they fit on one line
- Loops are more readable if there are if conditions