

Effective Programming Practices for Economists

Basic Python

Assigning variables and built-in scalar types

Janoś Gabler and Hans-Martin von Gaudecker

Contents

- Representing numbers: integers and floats
- Using Python like a calculator
- Comparing variables
- Representing True and False: Booleans

Integers

```
>>> a = 3
>>> a
3

>>> type(a)
<class 'int'>

>>> type(3)
<class 'int'>

>>> a = 5
>>> a
5
```

- Variables are assigned with a single `=` sign
- Types are inferred, not declared upfront
- Types can be inspected with `type()`
- You can re-assign variables with different values
- Ints can hold arbitrarily large numbers

Floats

```
>>> b = 3.1415
>>> b
3.1415

>>> type(b)
<class 'float'>

>>> c = 0.1 + 0.2
>>> c
0.3000000000000004
```

- Floats represent real numbers
- They are imperfect representations
 - Imperfect precision
 - Can hold values between -10^{308} and 10^{308}
- Will discuss this in detail later

Python as a calculator

```
>>> a = 3  
>>> b = 3.1415  
  
>>> b / a  
1.0471666666666668  
  
>>> (a + b) * 3  
18.42450000000002
```

- Arithmetic works as you would expect
- Brackets work as expected
- Mixing ints and floats converts everything to floats

Some things you need to know

```
>>> a**b  
31.54106995953402  
  
>>> b // a  
1.0  
  
>>> b % a  
0.14150000000000018
```

- `**` is exponentiation (not `^`)
- `//` is floored quotient division
- `%` yields the remainder of a division

Comparisons

```
>>> a = 3  
>>> b = 3  
>>> a == b  
True
```

```
>>> a < b  
False
```

```
>>> a >= b  
True
```

- Comparison operators are `==`, `<`, `>`,
`<=`, `>=`
- Remember: `=` is used for assignment,
not comparison
- The result of a comparison is a
Boolean

Booleans

```
>>> a = True
>>> b = False
>>> type(a)
<class 'bool'>

>>> a and b
False

>>> a or b
True

>>> not b
True
```

- Booleans can be `True` or `False` (case sensitive)
- `and`, `or` and `not` can be used to express complex conditions
- Fundamental for control flow we will see later