Effective Programming Practices for Economists

Basic Python

Strings

Janoś Gabler and Hans-Martin von Gaudecker

Topics

- Representing text: Strings
- Methods to manipulate strings
- String formatting

Assigning strings

```
>>> a = "Hello"
>>> type(a)
str

>>> b = 'single quote string with e
>>> c = "double quote string with e

>>> not_an_int = "123"
>>> type(not_an_int)
str

>>> not_an_int * 2
'123123'
```

- Strings can hold arbitrary text data
- Defined with single or double quotes
- Strings containing numbers do not behave like numbers!

Everything is an object == Everything has methods

- Any language has `` `int` ``, `` `float` ``, `` `bool` `` and `` `string` ``
- C, Fortran, ...:
  - low level types to store data efficiently and do fast calculations
- Python: Everything is an object
  - Objects with convenient methods
  - Trade efficiency for convenience
  - We can still get efficient when needed!

Some string methods

```
>>> a = "Hello World!"
>>> a.lower()
'hello world!'

>>> a.replace("!", ".")
"Hello World."

>>> a.startswith("Hello")
True
```

- There are many methods for string manipulation
- Full documentation

String formatting

```
>>> a = "Hello"
>>> b = "3"
>>> c = 3
>>> f"{a} {b}"
'Hello 3'
```

- `` `f-strings` `` allow you to puzzle together different strings
- Many useful applications:
  - Embed results of calculations in messages
  - Write good error messages
- Variables used in formatting are automatically converted to strings

Strings are a sequence type

```
>>> a = "Hello World!"
>>> len(a)
12

>>> a[0]
'H'

>>> a[1]
'e'

>>> a[-1]
'!'
```

- Most of the time, you can think of strings as scalar variables
- They are actually sequences of characters
  - Have a length
  - Can be indexed
  - *Can be sliced*
  - *Can be iterated over*
- Indexing starts at 0
- Negative indices start from the end