Introduction to Python for Economists

Session 1: Variables, simple data types & lists

Roland Mühlenbernd

26. Februar 2020

Organizational Matters

- **Frame literature:** Eric Matthes (2019), *Python Crash Course* (2nd edition), No Starch Press
- **Course website:** https://www.muehlenbernd.net/IPE
- **Evaluation**
  - Development of two applications
  - Documentation (Deadline: 16.03.2020)
- **Further valuable links:**
  - **Python ressources:** https://python.org/

Preliminary Schedule

- Feb. 26th: Introduction to the basic concepts
  - Session 1: Variables, simple data types, lists
  - Session 2: Coding, conditionals, dictionaries, loops
  - Session 3: User input, functions, classes
  - Session 4: Introduction and topic discussion
- Feb. 27th: Homework I: Course exercises
- Feb. 28th: Homework II: Main task session 1
- Mar. 2nd: Homework III: Main task session 2
- Mar. 3rd: Homework IV: Main task session 3
- Mar. 4th: Development of applications
  - Session 1: Files and exceptions
  - Session 2: Developing application 1
  - Session 3: Developing application 2
  - Session 4: Final discussion

Two steps for getting started

1. Installation of programming language ‘Python’ (version 3.6)
2. Installation of the editor ‘Sublimes Text’
3. Create a folder IPE on your desktop

Installing Python (Windows)

1. Test if Python already installed
   - 1.1 Open a command window (SHIFT+right click on desktop)
   - 1.2 Terminal opens: type **python**
2. If not, go to https://python.org/
   - 2.1 Go to downloads and and download Python 3.8.x
   - 2.2 Install Python (Select option: Add Python 3.8 to PATH)
   - 2.3 Test in terminal: type **python3 --version**
   - 2.4 Type **python3**
   - 2.5 Type **print("Hello Python 3")**
   - 2.6 Restart your computer (sets the python3 path)

Installing Python (MacOS)

1. Test if Python is latest version ($\geq$3.6)
   - 1.1 Open a terminal window (CMD+space bar, type **terminal**)
   - 1.2 Terminal opens: type **python** (it returns the version)
   - 1.3 Type **exit()** to leave python an go back to terminal
   - 1.4 Type **python3** to check is Python 3 is installed
2. If not, go to https://python.org/
   - 2.1 Go to downloads and and download Python 3.8.x
   - 2.2 Install Python
   - 2.3 Test in terminal: type **python3 --version**
   - 2.4 Type **python3**
   - 2.5 Type **print("Hello Python 3")**

Installing and setting up Sublime Text editor

1. Go to https://sublimetext.com/
2. Choose the download option for your OP, download and install
3. Open sublimes
4. X Go to: Tools → Build Systems → New Build System
5. X Write the following:

```
{
    "cmd": ["python3", "-u", "$file"],
}
```

6. X Save file as Python3.sublime-build
7. open new file and save as test_file1
8. Write print("Hello World")

Your first program: ‘Hello world’

```
print("Hello world!")
```

- save file as... **slides010809_hw.py** in folder Desktop/IPE
- press CTRL+B (Windows) or cmd+B (MacOS)
- open a terminal
- cd Desktop + ENTER
- cd IPE + ENTER
- **python3 slides010809_hw.py**

What is a variable? What is a value?

1. `message = "Hello Python world!"`
2. `print(message)`
3. `print(mesage)`

Naming and using variables

- variables can only contain letters, numbers and underscore
- variables cannot begin with a number
- underscores can be used to separate words:

```
greeting_message = "Hello world!"
```

- don't use variable names that are reserved for Python keywords (e.g. print)
- variables should be short but descriptive

Data type String

You can use single or double quotes to define a String:

1. `"This is a string."`
2. `'This is also a string.'`
3. `'I told my friend, "Python is great!"'`
4. `"I like the language 'Python' a lot!"`

String methods

1. `name = "ada lovelace"`
2. `print(name)`
3. `print(name.title())`
4. `name = "Ada Lovelace"`
5. `print(name.upper())`
6. `print(name.lower())`

Using variables in Strings

```
7. first_name = "ada"
8. last_name = "lovelace"
9. full_name = f"{first_name} {last_name}"
10. print(full_name)
11. print(f"Hello, {full_name.title()}!")
```

Note: F-strings were introduced with Python 3.6

Alternative:

```
full_name = first_name+""+last_name
print("Hello, "+full_name)
```

Special String commands

- \n: new line
- \t: tab

```
12. print("\nLanguages:\n\tPython\n\tC\n\tJava")
13. print()
14. print("Languages:")
15. print("\tPython")
16. print("\tC")
17. print("\tJava")
```

Exercise I

- Create two variables, one that has the value ‘albert’, and the other has the value ‘einstein’.
- Use the variables to print the following text:

Albert Einstein once said:
“A person who never made a mistake never tried anything new.”

Integers

1. `print(22)`
2. `print(22+5)`
3. `print(22-24)`
4. `print(3*5)`
5. `print(3*(-5))`
6. `print(6/3)`
7. `print(3**3)`
8. `print(2+3*4)`
9. `print((2+3)*4)`

Floats

```
10. print(0.1+0.1)
11. print(2*0.1)
12. print(3.0)
13. print(0.1+0.2)
14. print(4/2)
15. print(1+2.0)
16. print(3.0**2)
```

Variables of type Integer and Float

17. `my_number = 7`
18. `print(my_number)`
19. `my_number_doubled = my_number*2`
20. `print(my_number_doubled)`
21. `my_number_half = my_number/2`
22. `print(my_number_half)`
23. `print(my_number + my_number_half)`
24. `x, y, z = 1, 2, 3`
25. `print(x, y, z)`

Accessing a first list

1. `bicycles = ['trek', 'cannondale', 'redline']`
2. `print(bicycles)`
3. `print(bicycles[0])`
4. `my_bike = bicycles[1]`
5. `print(my_bike)`
6. `print(bicycles[3])`
6. `print(bicycles[-1])`
7. `text = f"My bike is a {bicycles[2].title()}."`
8. `print(text)`

Changing and Adding

1. `motorcycles = ['honda', 'yamaha', 'suzuki']`
2. `print(motorcycles)`
3. `motorcycles[0] = 'ducati'`
4. `print(motorcycles)`
5. `motorcycles.append('honda')`
6. `print(motorcycles)`

Inserting and Removing bp (by position)

```
7. motorcycles = []
8. motorcycles.append('honda')
9. motorcycles.append('yamaha')
10. motorcycles.append('suzuki')
11. print(motorcycles)
12. motorcycles.insert(1, 'ducati')
13. print(motorcycles)
14. del motorcycles[2]
15. print(motorcycles)
```

Removing bp using the pop() Method

```
16. my_last_mc = motorcycles.pop()
17. print(motorcycles)
18. print(my_last_mc)
19.
    print(f"My last mc was a {my_last_mc.title()}.")
20. my_first_mc = motorcycles.pop(0)
21.
    print(f"My first mc was a {my_first_mc.title()}.")
22. print(motorcycles)
```

Removing bv (by value) using the remove() Method

```
23. motorcycles = ['honda', 'yamaha', 'suzuki']
24. print(motorcycles)
25. motorcycles.remove('yamaha')
26. print(motorcycles)
27. expensive_mc = 'honda'
28. motorcycles.remove(expensive_mc)
29.
    print(f"{expensive_mc.title()} is too expensive.")
30. print(motorcycles)
```

Removing bv (by value) using the remove() Method

```
31. motorcycles.append('yamaha')
32. motorcycles.append('suzuki')
33. print(motorcycles)
```

Question: What happens when we remove suzuki?

Organizing a list: sorting

1. `cars = ['bmw', 'audi', 'toyota', 'ford']`
2. `cars.sort()`
3. `print(cars)`
4. `cars.sort(reverse=True)`
5. `print(cars)`
6. `print(sorted(cars))`
7. `print(cars)`

Note: Sorting a list alphabetically is a bit more complicated when the values are not all in lowercase. We will come to this point later again.

Organizing a list: reverse, length, and index errors

```
8. cars = ['bmw', 'audi', 'toyota', 'ford']
9. cars.reverse()
10. print(cars)
11. print(len(cars))
12. cars.remove('audi')
13. print(len(cars))
14. print(cars[3])
```

What happens when you want to print cars[3]?.

Exercise II

- Create a list with 4 US presidents in chronological order of office
- Create a variable ‘my_first_president’ with attributing the value from the first item of the list
- remove the first item from the list by using the variable
- add ‘my_first_president’ to the list (pos 2) by using the variable
- reverse the list and then sort it alphabetically
- add ‘kennedy’ to the list (last pos)
- Create a variable ‘number_of_my_presidents’ with attributing the value of number of list items by using len()
- Use the variable ‘number_of_my_presidents’ to access the last element of the list and print the following sentence.

Kennedy once said:

“Ich bin ein Berliner.”

Looping through lists

```
1. magicians = ['alice', 'david', 'carolina']
2. for magician in magicians:
3.     print(magician)
4.     print("A")
5. print("B")
6. for wizard in magicians:
7.     print(f"{wizard.title()}, great trick!")
8. print(wizard)
```

Looping with range

```
1. for value in range(1, 5):
2.     print(value)
3. numbers = list(range(1, 6))
4. print(numbers)
5. even_numbers = list(range(2, 11, 2))
6. print(even_numbers)
```

slides012930_range.py

Creating lists with range; min, max and sum

```
7.  squares = []
8.  for value in range(1,11):
9.      square = value**2
10.     squares.append(square)
11. print(squares)
12. print(min(squares))
13. print(max(squares))
14. print(sum(squares))
```

Slicing a List

```
1. players = ['ann', 'ben', 'tim', 'jen', 'eli']
2. print(players[0:3])
3. print(players[1:4])
4. print(players[:4])
5. print(players[2:])
6. print(players[-2:])
7. print("The first 3 players of the team are:")
8. for player in players[:3]:
9.     print(player.title())
```

Tuples: non-editable lists

1. `dimensions = (200, 50)`
2. `print(dimensions[0])`
3. `print(dimensions[1])`
4. `dimensions[0] = 250` (error)
4. `print(dimensions)`
5. `dimensions = (250, 50)`
6. `print(dimensions)`

Exercise III

- create a list that contains the odd numbers from 1 to 999
- print the minimum, the maximum and the sum of the list
- print the slice with the numbers 51 to 79 of the list
- create a list with cube numbers (raised to 3rd power) of 1 to 10
- create a list with five pizzas
- use the for loop to print a sentence for each pizza, such as “I like pizza x”.
- print the name of the first and of the last pizza of the list in the sentence “My favorite ones are pizza y and pizza z”.