Python Programming

Beginners Workshop

held at the Fablab Lisbon

January 2016

Requirements for the Workshop

- Installation of the Python program
  Download from http://www.python.org the current version of python3
- At http://python-rocks.blogspot.com there is a growing number of articles, that give additional explanations and useful links for this workshop.

The installation of Python provides us with a simple editor and a console window to execute Python commands and programs:

IDLE

There are better editors and development tools available, but for the beginning, IDLE should work well.

„I need more information!“

Many sources of help and useful information are available:

- The original Python documentation (F1 in the IDLE)
  ➡ create a link to the local documentation in your working environment
- Python documentation online at https://docs.python.org/3/
- There are many online tutorials on the web,
  e.g.: http://www.python-course.eu/python3_course.php
  or, more advanced: http://docs.python-guide.org (hitchhiker's guide)
- Documentation and tutorials are also available in portuguese:
  https://wiki.python.org/moin/PortugueseLanguage
- Ask your tutor (by mail: python-ws@bodens.de)

➡ **Go and search the Internet**

Usage of the Python shell (console)

Use of literal values

- Enter numbers, like: 3, 17 or 23482398473982457293487592465
- Enter simple calculations: like 2+3 or 7*12
- Try divisions, like: 12/4, 9/3, 11/5
- See the difference between integers and floating point numbers (3.14)
- Enter text strings, like: „hallo world“ or 'I am tired'
- Try out different mathematical operators: +, -, *, **, /, //, %, &, |
- Try the logical operators: <, >, ==, !=

➡ **The Python shell can be used as a calculator**

More usage of the Python shell

Use of variables

- Try to assign a name to a value (variable), like: a=7, b=2*3.9, t=“what“
- Enter the name of the variable
- Understand the difference between a name and a (literal) value

Call of functions

- print()
- len() - length of strings
- int() - convert a value to an integer, like: int(“765“), int(3.75)
- max(), min() - for numeric values
- input() - what does happen here?

Writing a first program

The infamous „Hello World“

- Use the IDLE editor (or any editor of your choice)
- Write a print statement
- Save the program text as a .py file
- Execute it (in the IDLE shell or on the command line)

The second program: Interaction with the user

- Ask the user for his/her name
- Write a greeting message like: „Hello Alice, it's nice to meet you“
- Save and execute the script

Control the flow

Conditional execution of a statement

- Let the user enter a number,
- then tell him/her, if the number is even or odd
- Use an if … else … statement

➡ **Indentation matters (no braces! - but colons!)**

- Execute a print statement several times
- Use a for loop and the range function
- Save and execute the script

Iterations

„for item in list:“

- Loop over an iteration
- Use „break“ to leave the loop
- Use „continue“ to skip the rest of the block
- „for“ also has an optional „else“ part

„while condition == True:“

- Loop with a condition expression
- Infinite loops can be useful
- Usage of „break“ and „continue“ as in „for“ loops

Import of Modules

Import of modules from the Python standard library

- Math-Module
  - import math - offers these functions (examples):
    - math.sqrt() calculates the square root
    - math.sin(math.radians(30)) => 0.5
- Time-Module
  - import time
    - time.time() - returns the current time in seconds (as a float)
    - time.sleep(2.0) - waits for 2 seconds
- Random-Module
  - import random
    - x = random.random() - returns a float: 0.0 <= x < 1.0
    - x = random.choice(list-of-items) – returns one of the items in the list

Workshop Projects (1)

Things we could do in the next few weeks

- Sure not all of them – the time is too short
- Perhaps not each of them – some may be too complex

- Easy Projects:
  - Rock, Scissor, Paper (simple game against the computer)
  - Hangman (word guessing)
  - Mastermind (decode numbers)
  - Towers of Hanoi (a puzzle)

Workshop Projects (2)

Things we could do in the next few weeks

- Useful Projects
  - Work with files and directories, search and rename Files
  - Music player
  - Read and write Office documents (Excel), process data
  - WebScraper – Get data from the Internet
  - ScreenBot – Automate browser games (not only for games)
  - Gui-Application programming

- Simulations
  - Game of Life
  - Elevator Simulation

Workshop Projects (3)

Things we could do in the next few weeks

- Artistic and Creative
  - Maze generator
  - Fractal graphics
  - Fractal Gallery (work with HTML)
  - Turtle graphics

- Other
  - Work with the Laser Cutter (SVG)
  - Work with the Raspberry Pi
  - Client-Server-App (Group Chat application)
  - Text-Adventure game
  - Puzzle solving

Rock, Paper, Scissor

Write a Program to play RPS with the computer

- The rules are simple:
  - Rock wins against Scissor (makes it 'unsharp')
  - Scissor wins against Paper (cuts it)
  - Paper wins against Rock (wraps it)
  - Equal bets are draws, no win points
- The computer makes an internal bet on R, P or S
- The user enters his bet
- The computer shows the result. (my bet, your bet, I win, you win, draw)
- The computer counts the win points.
- Who first gets 10 win points, wins the game

*==> A **detailed step-by-step description** of a possible solution is now **online.***

Even - Odd

Write a Program to distinguish even and odd numbers

- The program asks the user for a number
- It determins if the number is eve or odd
- A message is written about the result
- The user is asked again, until he/she wants to quit

This is easier than the RPS project, but it shall be written, using functions

- Use a function for the main part of the program
- Use a function for the user input
    - This function could also check, if the user input is „valid“
- Use a function for the even-odd checking

OOTS Scripts

Out Of The Shell - Show Python at work

- OOTS scripts are lists of python expressions and statements
- The scripts are converted into PDF documents
- Each page shows a number of „expressions“ and „statements“
  - '#' (green) are comments. Comments try to explain, what's goin on
  - '>>>' (black) are the expressions or statements
  - '. . .' (black) is for the continuation of a multi-line statement
  - '==>' (blue) are results or return values
  - 'p( )' (violet) shows printed output
  - 'err' (red) displays error descriptions

OOTS Scripts

Out Of The Shell - Show Python at work

- The black lines are „expressions“ or „statements“
    - The are „evaluated“ or „executed“ like in a Python shell
    - The difference is either obvious or not important. It will become clear over time
    - executed statements (like assignments) have no return value
    - evaluated expressions return a value, but it may be None
    - both can have side effects (like printing some text, defining a name, ...)

OOTS Scripts

OOTS Documents - Learn Python by example

Each OOTS document tries to cover one important concept of Python.

Each page of an OOTS document should be understandable by itself.

The OOTS documents are pesented during the workshop and are available for download from the workshop homepage.

The input text and a „text-version“ of the output text file are also available for examination.

Its **an exercise**, to look at the statements and **determine the output**

If the output of a statement is not completely clear: **ASK**!

Access to Files

„open() for reading“

- Open returns a „file“ object
- Beware: strings are not bytes. Encoding matters
- A file can be used as an iterator over the lines
- Closing files „.close()“ is recommended, can be manually, or automatically: „with“ context

fablab
LISBOA

python™