Introduction to Python — Excersise 1

Rebecca Breu

Juni 2010

Login: `XXXloginXXX`
Passwort: `XXXpasswortXXX`

Please change the password (`passwd`)!

Editors: emacs, gvim, eric, jedit

Data Types I

Exercise 1 (First steps, numbers) Start the Python interpreter.

- Use the online help: `help()`, `help(complex)`, `dir(complex)`
- Get familiar with the data types `int`, `float`, `complex` and the basic arithmetic operations.
- What happens if you combine different data types in one calculation?
- Display 1/3 as a decimal 0.333333....
- What is 1/0?
- Work with variable assignments. What happens if you successively assign objects of different data types to the same variable name?

Exercise 2 (Strings)

2.1 Write a program which asks both for a user's given name and family name. Afterwards, the program should print the full name and a greeting in one line, e.g.:

```
Your given name? Jim
Your family name? Kirk
Live long and prosper, Jim Kirk
```

2.2 Extend the above program so that it additionally prints:

- The letters 2 to 5 of the given name and the second to last letter of the family name.
- The full name in capital letters.
- Whether the name ends with "man".
- ...? Try more string methods: `http://docs.python.org/lib/string-methods.html`

Exercise 3 (Lists) Start the Python interpreter.

- Create a list with several elements and access those elements: The second element, the fourth to sixth elements, the last element. What happens if you use an invalid list index?
- Add additional elements, delete elements. How does Python handle different data types in a list?
- Join two lists.
- Can you add a list as a list's element? How do you access elements of such a sublist?

Statements

Exercise 4 (Loops and if-branches) Write a program which reads ten numbers from keyboard input and then prints their sum. *Hint:* `int`() and `float`() can be applied to strings.

4.1 Rewrite the program so that it stops reading numbers as soon as the sum is greater than 42.

4.2 Rewrite the program so that it ignores negative numbers.

4.3 Rewrite the program so that it continues reading numbers until the user types `quit`.

Exercise 5 Write a program which reads ten strings from keyboard input and saves them in a list. Afterwards, the program should replace all letters `a` in all strings with an `e` and print the result.

Hint: Use lists, but don't use list indeces `mylist[i]` to access the elements!

Exercise 6 The `ord` functions returns the ASCII representation of a character. Write a program which prints the ASCII representation of a string.

Hint: Again no indeces are needed!

Functions

Exercise 7 (Factorial) Write a function which returns the factorial $n!$ of a number n: $n! = 1 \cdot 2 \cdot ... \cdot n$. Recursion is not needed.

Exercise 8 (Ulam numbers) Write a function which returns the Ulam numbers of a number a_0 as a list. It is:

$$a_{n+1} = \begin{cases} a_n/2, & \text{if } a_n \text{ even,} \\ 3a_n + 1, & \text{if } a_n \text{ odd,} \end{cases}$$

at $a_n = 1$ the sequence terminates. Example:

```
>>> print ulam_numbers(6)
[6, 3, 10, 5, 16, 8, 4, 2, 1]
```

Exercise 9 (Palindrome) Write a function which checks whether a string is a palyndrome, i.e. a word which is the same read forwards and backwards. (You can assume that words are either all upper case or all lowercase.) What happens if you pass a list to the function instead of a string?

Exercise 10 (Free fall) The formula for the free fall is given by:

$$h(t) = h_0 - \frac{1}{2}gt^2,$$

where $h(t)$ is the drop height at time t, h_0 is the initial height at time 0 and g the gravitational acceleration. Implement a function `drop_height(h_0, t, g)`!

10.1 Call the function with keyword parameters. What happens if you change the order of the keyword parameters?

10.2 Adapt the program so that the gravitational acceleration g is 9.81 by default.

Exercise 11 What does the following code do?

```
def nothing():
    print "I don't return anything."

a = nothing()
print a
```

Input/Output

Exercise 12 (String formatting) Start the Python interpreter.

- Display `float` numbers with a given number of positions before and after decimal point, and in exponential notation.
- Display `int` numbers in hexadecimal and octal notation.
- Given a person's given name, family name, residence and age, display the data in one line, e.g.:

  ```
  "James Kirk is 35 years old and lives in Iowa."
  ```

- Try more formatting possibilities: `http://docs.python.org/lib/typesseq-strings.html`

Exercise 13 (Command line parameters) Write a program which prints its name and all its command line parameters.

Exercise 14 (Read files) Write a program which counts the frequency of the word "spam" in a file.

Exercise 15 (Read and write files) Given a file in which each lines contains arbitrary many numbers, separated by colons:

```
13.0:14.5:17:0.03:-2
89:-0.2354:6666
```

Read the numbers, add all numbers in one line and write the numbers into a new file in the same format as above, only that each line additionally contains the sum at the end.

Hint: Look at the string methods `strip`, `split` and `join`.

Exceptions

Exercise 16 Read a number from keyboard input and calculate its square. What happens if the user enters something that is not a number? Let the program display a user firendly message in that case!

Exercise 17 (Factorial revisited) The factorial is only defined for natural numbers. How does the function from Excersise 7 handle negative parameters? Let the program throw a `ValueError` in this case.

Exercise 18 Write a program which reads strings from keyboard input in an endless loop and checks each input if it is a palindrome (use the function from Excercise 9). What happens if the user presses `Ctrl-C` or `Ctrl-D`? Adapt the program so that, when `Ctrl-C` or `Ctrl-D` is pressed, it asks the user if they want to exit the program.

Exercise 19 The module `readline` provides shell-like editing features for command line input. Generally, the module is only available on *nix systems. The following does not import the module on Windows machines:

```
if not sys.platform.startswith("win"):
    import readline
    import rlcompleter
    readline.parse_and_bind("tab: complete")
```

Why is a solution with exceptions better? Rewrite the code so that it uses exceptions.

Additional Excersises

Exercise 20 (High/Low) Implement the game High/Low: A player (in this case the computer) thinks of a number between 1 and 100, the other player (in this case the user) needs to guess the number. After each try, the user is informed if his guess was too high, too low, or correct.

Proceed by the following steps:

20.1 The program lets the user play one round and displays the number of guesses at the end of the round. *Hint:* You can create a random number between 1 and 100 by using `random.randint`(1, 100).

20.2 The program asks after each round if the user wants to play again.

20.3 The program saves a high score list in a file.

Not enough? More exercises:

http://www.pythonchallenge.com