# Introduction in Python — Part 1

Login: `XXXloginXXX`
Password: `XXXpasswortXXX`

Please change Your password (`passwd`)!

Editors: kate, kwrite, emacs, gvim, gedit, jedit, eric

Python documentation: http://docs.python.org

## Data Types I

**Exercise 1** (First steps, numeric data types) Start Python shell.

- Try the online help: **help**(), **help**(**complex**), **dir**(**complex**)
- Take a look at the diverse data types `int`, `float`, `complex` and the basic operations.
- What happens if you use different numerical data types in a calculation?
- Try to convert 1/3 into a floor division and get an integer result (discarding any fractional result)
- What is the result of 1/0?
- Use variables. What happens when objects of different data types are assigned to the same variable?

**Exercise 2** (Strings)

**2.1** Write a program which will ask for the forename and surname. Then it should write a greeting using both names, e.g.:

```
Your forename? Jim
Your surname? Kirk
Live long and prosper, Jim Kirk!
```

**2.2** Extend the last program to give additional output:

- The characters 2 to 5 of the forename and the last but one[1] character of the surname.
- The complete name in upper case.
- Does the surename end with „mann“?
- . . . ? Test various string methods:
  `https://docs.python.org/3/library/stdtypes.html#string-methods`.

**Exercise 3** (Lists) Start a Python shell.

- Create a list with several elements and access target list element: The second, the fourth to the sixth, the last. What happens, when the list index is out of range?
- Add new elements to the existing list, delete some. What happens if the data types vary between the elements in the same list?
- Join two lists into one.
- Can a list be added as one element in another list? How can you access then an element from this sublist?

## Control Statements

**Exercise 4** (Loops and conditional statements) Write a program to read ten numbers (keyboard input). Finally the sum of these numbers should be printed.

*Hint:* **int**() and **float**() can be used on strings.

**4.1** Modify your code to stop reading numbers, when the subtotal is greater than 42.

**4.2** Modify your code to ignore negative input numbers.

**4.3** Redesign your program to read numbers until the user entered `end`.

**Exercise 5** Write a program, which will read ten strings (keyboard input) and save them into a list. Afterwards the character `a` should be replaced by `e` in all strings. Print the resulting strings.

*Hint:* No index required!

**Exercise 6** the **ord**-function returns the Unicode code point for a one-character string. Write a program which will ask for an input string and will print it out as a sequence of numbers in this notation.

*Hint:* No index required!

[1] vorletzte

## Functions

**Exercise 7** (Factorial) Write a function, which will return the factorial of number $n$: $n! = 1 \cdot 2 \cdot ... \cdot n$. There is no recursion necessary.

**Exercise 8** (Collatz-Sequence) Write a function, which will return the *Collatz* sequence to a given number $a_0$ as a list. This sequence is defined as:

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

When $a_n == 1$ the sequence is finished.

Example:

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

**Exercise 9** (Palindrome) Write a function to test if a given string is a palindrome or not. A palindrome is a word which reads the same backward or forward (it can be assumed that there is no mixed case). What happens if the function gets a list instead of a string?

**Exercise 10** (Free Fall) The formula for the free fall is:

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

where $h(t)$ the height of the object at time $t$ is, $h_0$ the initial height at time 0 and $g$ the gravity acceleration. Implement a function `freefall_height(h_0, t, g)`!

**10.1** Call this function by using keyword parameter. What happens, when the order of the keyword parameter is changed?

**10.2** Modify your function so that the gravity acceleration $g$ is 9.81 by default.

**Exercise 11** What happens on the following code? Will there be an exception?

```
def nothing():
    print("I will return nothing.")

a = nothing()
print(a)
```

## Input/Output

**Exercise 12** (String Formatting) Start a Python shell.

- Print out **float** numbers using fix digit before and after the decimal point. Also try scientific notation form.

- Print out **int** numbers in hexadecimal and octal notation.
- You have the variables `forename`, `surname`, `location`, `age` with proper contents. Print out a line with a similar layout like here:

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

- Give some other formatting a try:
  `https://docs.python.org/3.5/library/string.html`

**Exercise 13** (Read file) Write a program which expects a filename as command line parameter. Count the appearance of the word „spam“ in this file. *Hint:* There is a helpful string method. Regular Expressions are not necessary!

**Exercise 14** (Read and write file) Write a program which expects two arguments (input filename and output filename). Create a copy of the input file where each line has a leading row number.

Example output file:

```
1. hello
2. world
```

## Exceptions

**Exercise 15** Write a program which ask the user for a number $n$ and will print out the square of it ($n^2$). What happens, if the user will enter something different than a number. Try to give a user-friendly error message!

**Exercise 16** (Extended factorial) The factorial is only defined for natural numbers. How does the function from exercise 7 react on negative values? Modify your function to raise a `ValueError` in this case.

**Exercise 17** Write a program which will run an endless loop, where a user input string is tested, if it is a palindrome or not. (Use your function from exercise 9.)

What happens if the user presses `Ctrl-C` or `Ctrl-D`? Change the code so that the user is asked to exit the program when he has pressed `Ctrl-C` or `Ctrl-D`.

**Exercise 18** The module `readline` adds extended features to the `raw_input` method like editing and autocomplete. This module is available on *nix systems by default. The following code does not import the module on windows systems:

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

Why is the approach using exceptions the better way? Rewrite the code to use exceptions instead.

## Bonus exercises

**Exercise 19** (Guessing numbers) Develop a game: One player (computer) figures out a random number from 1 to 100. The second player (user) has to guess this number. After every try the computer player will tell, if the guessed number is correct, to low or to high. The round ends when player two has found the right number.

Realize the program in these steps:

**19.1** The program will play one round with the user. At the end the counted tries will be printed out.

*Hint:* A random number from 1 to 100 can be created with the `random.randint`(1, 100) method.

**19.2** At the end of each round the computer will ask the user to play once again.

**19.3** The program will save the results in a high-score file.

Not enough? More exercises:

http://www.pythonchallenge.com