

Discovering Python

Workshop Vilnius 2017
Michael Weigend

Draft version - to be extended

Just choose something you are interested in and start.
If you need more information about Python use the Python manual.

Part 1

1 Expression Statements

Use the interactive mode of IDLE3 (the Python Shell) and type in these expressions. Can you explain the results?

```
>>> 3 * "Hurray "  
>>> 2/3  
>>> 2//3  
>>> type(3/2)  
>>> -2**-3  
>>> 99 ** 234  
>>> not(2.0 > 2)  
>>> 'a' in 'Delft'  
>>> len('123')  
>>> type(len)
```

Important Shortcuts for the IDLE Shell:

ALT + P previous command
ALT + N next command

2 Interactive Programs

Use *IDLE3*. Open a new editor window (*file/New file*). Save the file somewhere. Use a file name with the extension `.py`, for example `zoo.py`.

Write this program, save it (*CRTL + S*) and test it (*Run/Run module* or *F5*).

```
print ("Welcome to the zoo!")  
age = float(input("Your age: "))  
if 14 < age < 60:  
    print ("Please pay 10 Euros")  
else:  
    print ("Please pay 5 Euros")
```

Write an interactive program, which does this:

The user is asked for height and weight. The program calculates the bmi (body-mass-index) and prints a result. Example dialogue:

```
Your height (in meters): 1.8  
Your weight (in kg): 80
```

```
Your BMI is 24.  
Your weight is just fine.
```

Some Hints

```
bmi = weight / (height * height)
```

The normal bmi is 18.5 to 24.9. Values between 25 and 30 indicate overweight. A value greater than 30 indicates severe overweight. A bmi lower than 18.5 indicates underweight, and if it is lower than 16 the underweight is critical.

3 Sequences

Lists, strings and tuples are sequences.

Examples: `[1, 2, 3]`, `"word"`, `("Tom", 20)`

Lists are mutable, tuples and strings are not. Try this:

```
>>> a = [1, 2, 3]
>>> a[0]
>>> a[1]
>>> a[-1]
>>> del a[1]
>>> a

>>> b = "Hallo"
>>> b[0]
>>> del b[0] # you will get an error message
```

Sequences have common attributes and methods. For example they have a length and you can concatenate them. Try this

```
>>> len("word")
>>> len([])
>>> len([[[]]])
>>> s = [1, [2, 3, 4]]
>>> len(s)
>>> len([[[]], []])

>>> s = ["moon", "honey", "ship", "space"]
>>> print(s[0])
>>> print(s[1]+s[0])
>>> print(s[-1]+s[-2])
>>> for word in s:
    print(word[0])
```

The range function generates a range-Object representing a sequence of numbers. If you want to see the numbers you must create a list or another explicit sequence. Try this :

```
>>> range(5)
>>> list(range(5))
>>> list(range(2, 10))
>>> list(range(-2, 2))
>>> for i in range(10):
    print(i)
```

4 Dictionaries

This statement creates a dictionary mapping e-numbers to chemical substances:

```
>>> d={'E260':'acetic acid',
      'E200':'Sorbic acid',
      'E210':'Benzoic acid'}
```

Try these statements.

```
a) >>> d['E210']
b) >>> print (d.keys())
c) >>> del d['E210']
   >>> print (d.values())

d) >>> for k in d.keys():
      print k+'': '+d[k]

e) >>> d['E239']='potassium nitrate'
   >>> for k in d.keys(): print (k, d[k])
```

Write an interactive program that reads a E-number and returns the name of the corresponding chemical. (You find a solution in the workshop folder.)

5 Turtle Graphics – Experiments With Recursive Functions

Python supports the famous Logo Turtle graphics. Open the script `triangle.py` with IDLE and run it.

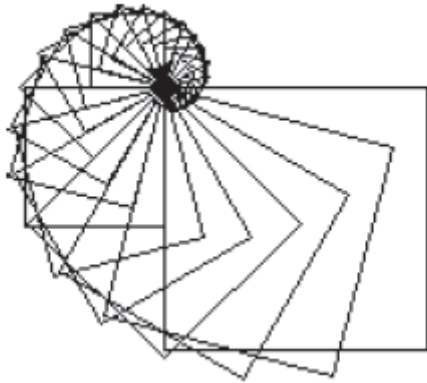
```
# triangle.py
from turtle import *

def triangle(n):
    for i in range(3):
        forward(n)
        right(120)

def shape(n):
    if n > 1:
        triangle(n)
        right(60)
        shape(n/2)

clear()
speed(5)
left(90)
shape(200)
hideturtle()
```

Try to create a different structure using a recursive function, for example this:



6 Quicksort and Assertions

```
# quicksort
from random import randint
def qsort (sequence):
    s = sequence[:]          #1 s is a copy of sequence
    if s == []:
        result = s          #2 end of recursion
    else:
        x = s[0]            #3 take first element
        s1 = []             #4 split remaining list
        s2 = []
        for i in s[1:]:
            if i <= x:
                s1.append(i)
            else:
                s2.append(i)
        result = qsort(s1) + [x] + qsort(s2) # recursive calls

    return result

# main program
s = [randint(0, 10) for i in range(10)] #5 ten random numbers
print (s)
print (qsort(s))
```

Tasks

- Start IDLE and open this program. You find it in the workshop folder. Test it.
- Modify statement #5. It should create a list with 20 random numbers between 0 and 100.
- What happens, when you replace statement #1 by this line?
`s = sequence`

- Add some `assert` statements that check the logical correctness, for example:

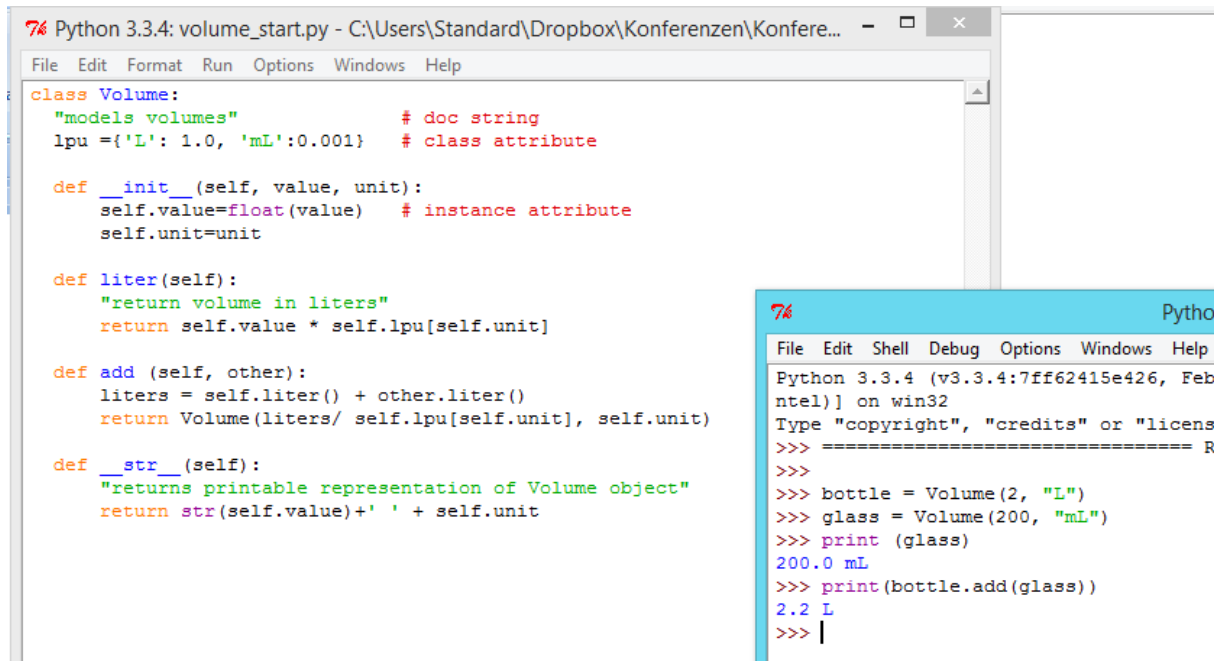
```
assert len(result) == len(sequence)
assert result[0] = min (sequence)
```

7 Object Oriented Modelling

Use IDLE3 and open the script `volume_start.py`. You find it in the workshop folder.

Execute the script and then test the class `Volume` by entering a few statements in the Python Shell window (see image):

```
>>> bottle = Volume(2, "L")
>>> glass = Volume(200, "mL")
>>> print (glass)
>>> print(bottle.add(glass))
```



The image shows a Python IDE window titled "Python 3.3.4: volume_start.py" with a menu bar (File, Edit, Format, Run, Options, Windows, Help). The code in the editor defines a class `Volume` with the following methods and attributes:

```
class Volume:
    """models volumes"           # doc string
    lpu = {'L': 1.0, 'mL':0.001} # class attribute

    def __init__(self, value, unit):
        self.value=float(value) # instance attribute
        self.unit=unit

    def liter(self):
        """return volume in liters"
        return self.value * self.lpu[self.unit]

    def add (self, other):
        liters = self.liter() + other.liter()
        return Volume(liters/ self.lpu[self.unit], self.unit)

    def __str__(self):
        """returns printable representation of Volume object"
        return str(self.value)+' ' + self.unit
```

To the right, a Python Shell window titled "Python 3.3.4" shows the execution of the script:

```
Python 3.3.4 (v3.3.4:7ff62415e426, Feb
ntel)] on win32
Type "copyright", "credits" or "licens
>>> ===== R
>>>
>>> bottle = Volume(2, "L")
>>> glass = Volume(200, "mL")
>>> print (glass)
200.0 mL
>>> print(bottle.add(glass))
2.2 L
>>> |
```

Tasks

a) Change the name of the method `add()` to `__add__()`. Execute the script again and try these statements

```
>>> a = Volume(20, "ml")
>>> print(a + a + a)
```

b) Add another method which implements multiplication of volumes:

```
def __mul__ (self, x):
    ...
>>> a = Volume(20, "mL")
>>> print (a * 20)
400.0 ml
```

Part 2 GUI Programming and Raspberry Py

Choose one of these projects.

Project 1 Text Editor

This is a minimalist XP project.

The Metaphor

The goal is to develop an editor for a special purpose (love letters, letters in English, reporting accidents, writing experiment records ...) and a special group of users (girls, boys, policemen, ...)

Choose a specific metaphor for your project (for example "Letter fairy. A program that supports writing a letter in a foreign language.").

Release Planning

Write down a few stories that describe the product that you are intending to create. Example:

- GUI with pull down menus.
- Boiler plates (text modules) for frequently used phrases
- The user can load and save text files
- The user can change the background colour

Architectural Spike.

Try out the python programs in the folder "editor". Choose an appropriate program as a starting point for your project.

Iterations

In each iteration you implement a story. You implement a story by editing the existing program and adding some features. Check the other scripts and steal ideas from them.

Some Background Information

The components of a GUI are called widgets. Examples are windows (instances of the class Tk), Labels (instances of the class Label) and Buttons (instances of the class Button).

You can configure the attributes (options) of widgets in two ways:

- Using keyword parameters when you create a widget. Example:
`self.labell=Label(master=self.window, bg="green")`
- Calling the method config(). Example: `self.labell.config(bg="green")`

Some Attributes(Options) and Methods of Widgets

Option	Explanation
bd, borderwidth	integer that determines the width of a border, example bd=5
bg, background	background colour. examples: bg='#23FF10', bg='red', bg='blue', bg='green'
fg, foreground	foreground colour or text colour
font	Font descriptor, for example: font=('Arial', 20) means

	font Arial with a height of 20 points.
height	height of a widget
image	name of an image object that is displayed on a widget
justify	Justify text on a widget, values: CENTER, LEFT, RIGHT
relief	form of a boarder, values: SUNKEN, RAISED, GROOVE, RIDGE, FLAT
text	text on a widget
width	width of a widget

Table 1: Standard attributes of widgets

Method	Explanation
after (ms , func[,arg1[,...]])	Call a function after ms milliseconds
bell()	Ring a bell
cget(option)	Returns the value of the specified widget
config(option1=value1, ...)	Configure a widget label.config(text='new text')
pack()	Layout. Put the widget on its master widget

Table 2: Some common methods of widgets

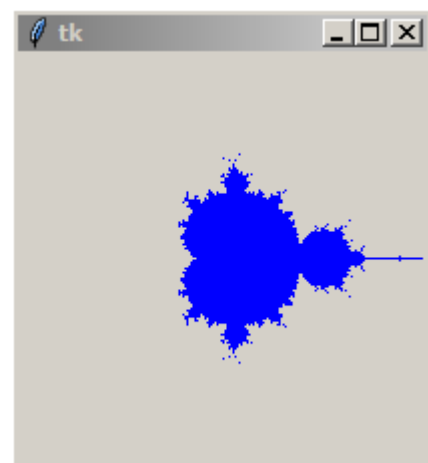
Project 2 Visualizing a Mandelbrot Set

You find this program in the workshop folder.

```
# mandelbrot.pyw
from tkinter import *
RADIUS = 2.0
ZOOM = 50.0

class Mandelbrot:
    def __init__(self):
        self.window = Tk()
        self.image = PhotoImage(width=200,
height=200)
        self.image_label =
Label(master=self.window,
        image=self.image)
        self.image_label.pack()
        self.draw()
        self.window.mainloop()

def draw(self):
    interval = [x/ZOOM for x in range(-100, 100)] #1
    mandelbrot = [(x, y) for x in interval
        for y in interval
        if self.test(x, y)]
    for x, y in mandelbrot:
```



```

        self.image.put("#0000ff", (int(ZOOM*x+100), int(ZOOM*y+100)))

def test (self, x, y):
    c = x + 1j * y          # j is the imaginary number i
    z = 0
    for i in range(20):
        if abs (z) < RADIUS:
            z = z*z - c
        else: return False # not in the Mandelbrot set
    return True           # element of the Mandelbrot set

m = Mandelbrot()

```

Comment

#1: This is called a list display. It is a very concise way to define a list. In this case interval is a list of 200 numbers from -2.0 to +2.0.

Tasks

- 1) There are two constants RADIUS and ZOOM. What happens when you change these? Find out the meaning of these constants.
- 2) Change the colors of the background and the points of the Mandelbrot set

Project 3 Image Processing

```

from tkinter import *
class App:
    def __init__(self):
        self.filename="manchester_6.ppm"
        self.window = Tk()
        self.pic = PhotoImage(file= self.filename)
        self.c = Canvas(self.window, width=self.pic.width(),
                        height=self.pic.height())

        self.c.pack()
        self.c.create_image(0, 0, anchor=NW, image=self.pic)
        self.ExtractButton = Button(master=self.window,
                                    text="Find Words",
                                    command=self.extract)

        self.ExtractButton.pack()
        self.window.mainloop()

    def extract(self):
        w = self.pic.width()
        h = self.pic.height()
        colors = [self.pic.get(i,0) for i in [0, 1, 2, 3]]
        pixels = [(x, y) for x in range(w) for y in range(h)]
        for (x, y) in pixels:
            if self.pic.get(x, y) not in colors:
                self.pic.put("white", to=(x, y))
            else:
                self.pic.put("{black black} {black black}", to=(x, y))

App()

```

Task

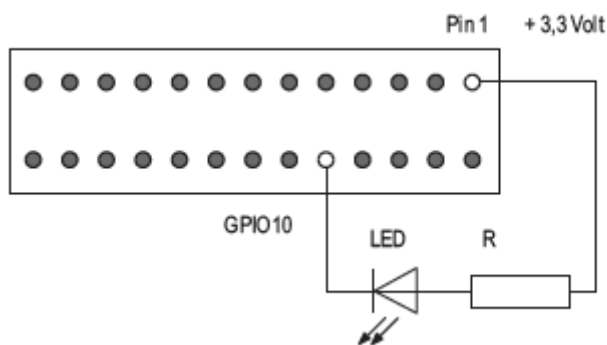
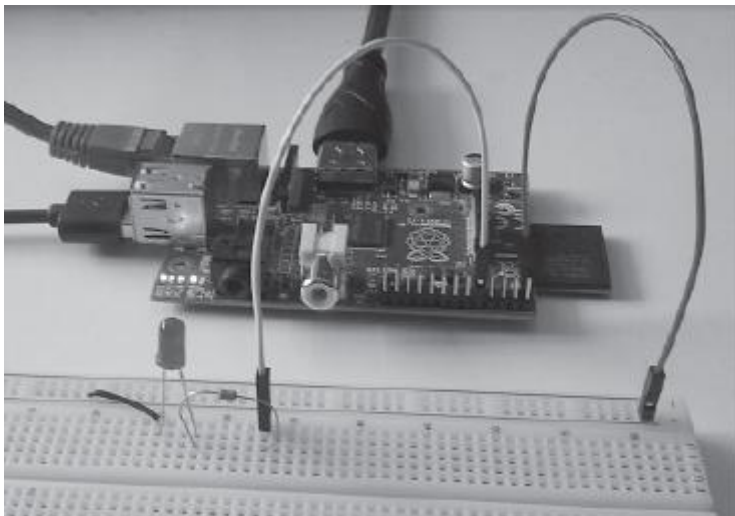
This time you are the teacher. Invent a few tasks that are based on this script.

Special Projects for the Raspberry Pi

Make sure that you have copied the workshop folder to your home directory on the SD card of your RPi.

Project RPi1 Blinking LED

Connect an LED and a resistor (120 Ohms) to the GPIO according to the wiring diagram. At Pin 1 of the GPIO you can see "P1" on the board of the RPi. The shorter Pin of the LED is (-) and must be connected to Pin 10 of the GPIO, the longer Pin is (+).



```
# blinking:led.py
from RPi import GPIO
from time import sleep
GPIO.setmode(GPIO.BOARD)
GPIO.setup(10, GPIO.OUT)
while True:
    GPIO.output(10, False)
    sleep(0.5)
    GPIO.output(10, True)
    sleep(0.5)
```

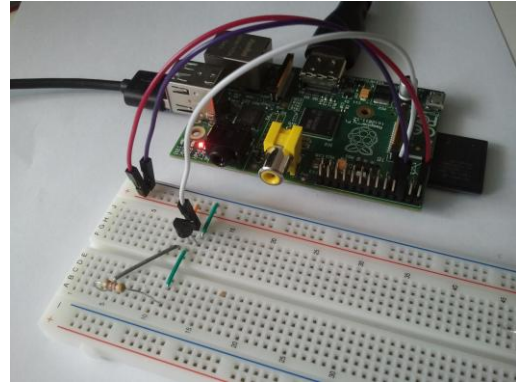
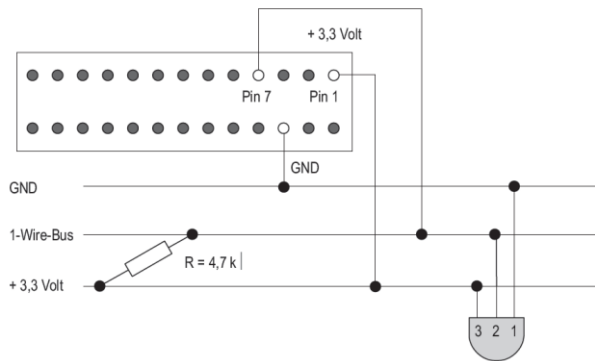
#1 Pin 10 of the GPIO is Output
#2 repeat forever
#3 set Pin 10 on low voltage
#4 wait
#5 set Pin 10 on high voltage

Open this program and run it. The LED should blink. You can stop the program by a keyboard interrupt. Use `ctrl + C`.

Task: Make the LED signaling "SOS"!

Project RPi2 Logging Temperature Data

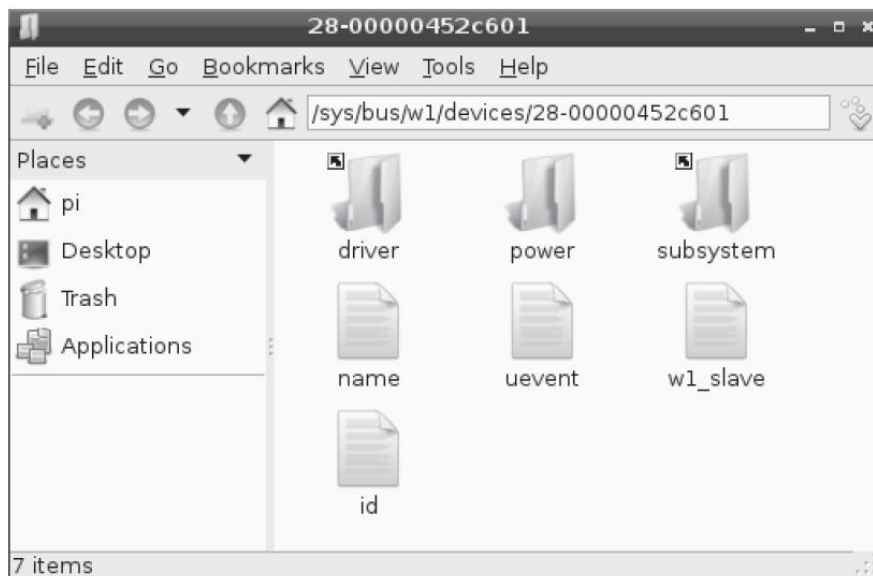
Connect the temperature sensor according to the wiring diagram. Pin 1 of the GPIO is marked with P1. The DS18B20 or DS18S20 is shown from above. Ask an expert for checking the wiring before connecting the RPi to the power supply.



Open an LX-terminal and type these commands:

```
sudo modprobe wire
sudo modprobe w1-gpio
sudo modprobe w1-therm
```

Find the directory of the thermometer (see image) and read the file `w1_slave`.



Start IDLE 3. Click on *File/Open* and open the file `/home/pi/workshop/temperature.py` XYZ is the identifier of the memory stick.

```
# temperature.py
import os
os.system("modprobe wire")
```

#1

```

os.system("modprobe wl-gpio")
os.system("modprobe wl-therm") #2

for d in os.listdir("/sys/bus/wl/devices"):
    if d.startswith("10") or d.startswith("28"):
        deviceFile = "/sys/bus/wl/devices/" + d + "/wl_slave" #3

def readTemp():
    ok = False
    while not ok:
        f = open(deviceFile, "r")
        firstLine, secondLine = f.readlines() #4
        f.close()
        if firstLine.find("YES") != -1:
            ok = True #5
    tempString = secondLine.split("=")[1] #6
    return int(tempString)/1000

while True:
    print(readTemp())
    time.sleep(1)

```

Comments

- #1: The module `os` contains functions that are related to the operating system.
- #2: Start the program *modeprobe*.
- #3: You have found the file containing the temperature data
- #4: Read the two lines of the text file.
- #5: The word YES has been found in the first line. The data are valid. The loop can be left.
- #6: The second line is split in two parts. The second part is the temperature string.

Task

Make the output of this program nicer, for example:

```

21.0 °C
21.2 °C
21.3 °C
...

```

Project RPi3 The Camera Module

Ask an expert (a person who has done this before) to connect the camera module. You can use the IR camera module (noIR) which does not filter IR light. Then you can use this camera in the dark with an IR LED as illumination.

Open an LXTerminal and type

```
raspistill -v -o image.jpg
```

You will see the camera live image for five seconds. Then it is stored in the file `image.jpg`.

Start the program `photoshooting.py`. It takes a photo each 10 seconds and stores it.

References

Most program examples and images are taken from these books:

- Michael Weigend: Python 3 lernen und professionell anwenden, Heidelberg (mitp) 2013
- Michael Weigend: Raspberry Pi programmieren mit Python, Heidelberg (mitp) 2014
- Michael Weigend: Raspberry Pi für Kids, Heidelberg (mitp) 2014