Funded by
DFG Deutsche Forschungsgemeinschaft
German Research Foundation

416591334

CRC 1412
REGISTER

Python for linguists

SFB 1412 Methodschool

André Renis

26-28 May 2021

Python quick Facts

CRC 1412 REGISTER

- Python programming language has been developed in the early 90s by Dutch programmer Guido van Rossum
- Name "Python" comes from "Monthy Python's Flying Circus“
- Objectives has been:
  - Easy to learn and to understand (readability)
  - Platform independent
  - General-purpose programming language:
    - Supports procedural, functional and object-oriented programming
  - Highly extensible (includes huge standard library)
- One of the most popular programming languages (together with C and Java)
- Now often pre-installed on many Unix-compatible operating systems such as MacOS and Linux.

Python version history

Version 1:
January 1994

Version 2:
October 2000

Version 3:
December 2008

Versions are not compatible!

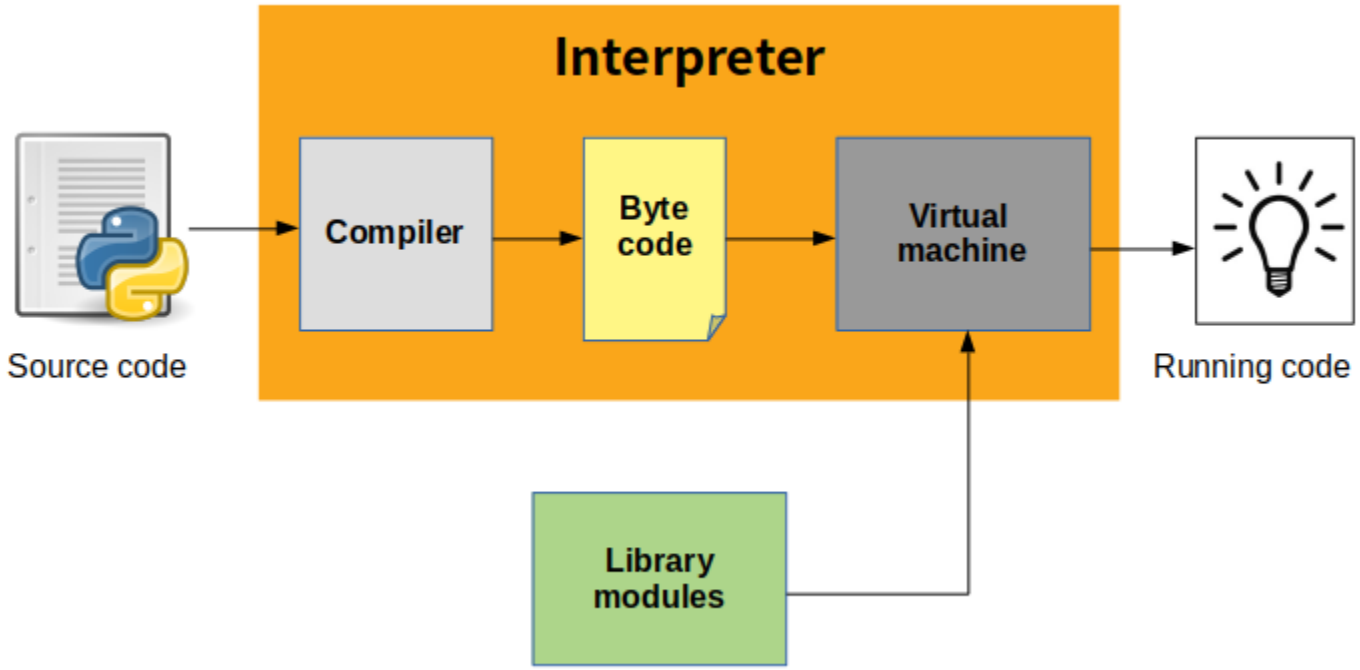
On many Linux-Systems Python2 **and** Python3 are installed at the same time! → Some problems/difficulties

Top 10 uses of Python in the real-world

CRC 1412 REGISTER

1. Web application development (including mobile apps)
2. Data Science
3. Artificial Intelligence
4. Game development
5. Internet of Things (logistics)
6. Web Scraping (big data)
7. Desktop GUI
8. Enterprise applications
9. Image recognition and text processing
10. Education programs

How Python works on your computer



Quelle: https://python-kurs.github.io/sommersemester_2019/units/S01E01.html

Python editors

CRC 1412 REGISTER

- Code editors are providing a lot of help with syntax check and testing/running/debugging your code on the fly
- You cannot use for example MS Word for programming because it‘s *formated text* and not *plain text*
- Common editors for programming in python are:
  - IntelliJ (PyCharm)
  - Visual Studio Code
  - Atom
  - Sublime Text
  - Vim
  - Emacs
  - Eclipse

Hello World! & Hello Visual Studio Code

CRC 1412 REGISTER

Hands on! We want to write our first program together

```
>>> print("Hello World!")
Hello World!
>>>
```

And we want to explore Visual Studio Code and check if our installation works.

Remarks and information on data privacy you can find here: https://thomaskrause.github.io/nlp-mit-python/setup.html

How do I write a program or a script?

CRC 1412 REGISTER

- A script runs from the top of the file until the end, line by line
- Every line is ONE command and will be executed
- First you break your task into logical units, for example (modifying content of a file):
  1. Read content of a file
  2. Modify content
  3. Save modified content to file
- Second you break each unit into single *commands,* step-by-step, line-by-line (according to documentation of used lib or your own commands)
- But, to save for example the modified file-content somewhere, we need some kind of cache and assign a value to it in our script. With *data types*, *variables* and *operators* we can do that.
- A program (only more complex compared to a script) doesn‘t run necessarily from top to end, because a program is divided into *modules* with *methods/functions* and *control structures* for different scenarios and cases we want to distinguish in the behavior of our program.
- Variables, data-types, operators, control structures and functions are the most basic ingredients of every (higher) programming-language. More later!

What makes Python a ready to use programming-language?

CRC 1412 REGISTER

Core Python

- Operators, data types & variables
- Special keywords
- Control structures
- Built-in-functions

Python standard libraries

- Already installed (& precompiled to .pyc)
- Functions can simply be used with ***import***

3rd party libraries

- Can be downloaded and installed on system with ***pip***
- Functions can be used with ***import***

Core Python: Scripting

CRC 1412 REGISTER

```
#!/usr/bin/env python3

'''

Fuel Calculator, a more complex script with variables and the use of
built-in-functions

'''

fuel = 34
kilometers = 456
petrolConsumption = kilometers/fuel
petrolConsumption = round(petrolConsumption, 2)
message = "Your petrol consumption has been "
print(message + str(petrolConsumption)) # type cast from float to string
```

Core Python: built-in-functions

CRC 1412 REGISTER

abs()	delattr()	hash()	memoryview()	set()
all()	dict()	help()	min()	setattr()
any()	dir()	hex()	next()	slice()
ascii()	divmod()	id()	object()	sorted()
bin()	enumerate()	input()	oct()	staticmethod()
bool()	eval()	int()	open()	str()
breakpoint()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	

Core Python: Variables always have a data-type

CRC 1412 REGISTER

Text Type:	str
Numeric Types:	int, float, complex
Sequence Types:	list, tuple, range
Mapping Type:	dict
Set Types:	set, frozenset
Boolean Type:	bool
Binary Types:	bytes, bytearray, memoryview

```
a = 4
b = "test"
print(type(a)) # built-in-type-check
print(type(b))
```

Core Python: Keywords

CRC 1412 REGISTER

We will use Pythons „interactive help“ to display a list of the so called „keywords“ in Python. They are a little bit different in all programming languages. They are reserved for special purposes.

1.) Type „help()“ in python console to start interactive help

2.) Type „keywords“

→ Offline available and useful. We will have a look at the “modules” later

Core Python: functions and methods – parameters & return value

CRC 1412 REGISTER

What is the main difference between the built-in-functions

A.) print(„Something“)
B.) round(3. 1415926) ?

```
import math
def printAndRound(number):
    number = round(number, 2)
    print(number)
    return number
result = printAndRound(math.pi)
```

Core Python: Control structures

CRC 1412 REGISTER

```
if (condition): # True/False
    expression1 # executed only if True
elif(other_condition): # True/False - More elif
                      #conditions can be added
    expression2 # executed only if True
else:
    expression3 # executed only if both above False
```

Core Python: Loops

CRC 1412 REGISTER

```
while(condition): # True/False
    expression # is executed every loop

for(start stop sequence): # for example runs from
                          # first word of
                         # a text to the last one;
                          # iterates from start
                          #  to stop condition
    expression # is executed every loop
```

Core Python: if … elif … else

CRC 1412 REGISTER

```
if temperature > 20:
  turnOffHeating()
elif temperature == 20:
  print("We have reached 20°C now!")
else:
  turnOnHeating()
```

Core Python: lists & for-loop

CRC 1412 REGISTER

Hands on:

We want to calculate the percentage of articles in a text.

We will use python online documentation for that. We need lists an loops for that.

A list in python looks like this:

```
articles = ["a", "an", "the"]
```

https://www.w3schools.com/python/default.asp

Core Python: lists & for-loop

CRC 1412 REGISTER

```
text = '''
The CRC Register: Language Users' Knowledge of Situational-Functional Variation investigates
aspects of the register knowledge of the speakers of a language. Competent speakers can adapt
 their linguistic behavior on every level in response to the current situation:
 They know, for example, that the German word sauer 'ticked off' is appropriate
 in different situations than the word verärgert 'angry',
 that one uses less complex sentences when speaking with children than in an academic function,
 and that sometimes it matters whether one says around 8 o'clock or 7:49 am, and sometimes it doesn't.
 We are thus concerned with intraindividual variation.
'''
articles = ["a", "an", "the"]
articlesCount = 0
words = text.split()
for word in words:
    if word.lower() in articles:
        articlesCount = articlesCount+1
    print(word)
print("The percentage of articles in this text is: " + str(articlesCount/len(words)*100) + "%")
```

Core Python: while + break & continue

CRC 1412 REGISTER

```
def while1(end):
    i = 1 # loop runs from 1 to end
    while i < end:
        print(i)
        i += 1

def while2(end):
    i = 1
    while i < end:
        print(i)
        if i == 3:# loop ends at break keyword
            break :# break is end of loop
        i += 1
```

```
def while3(end):
    i = 0
    while i < end:
        i += 1
        if i == 3:
            continue # execution ends here but
                     # continues
        print(i) # NOT executed in case of i==3

# while1(6)
# while2(6)
while3(6)
```

Core Python: Real-World Example

CRC 1412 REGISTER

Hands on:

Lets open a text file with german umlauts. Then we replace the german umlauts and save the content to a new file.

We will use python online documentation for that.

https://www.w3schools.com/python/default.asp

Core Python: Real-World Example

CRC 1412 REGISTER

```
file = open("example.txt", encoding='utf8')
text = file.read()
text = text.replace('ü','ue')
text = text.replace('ä','ae')
text = text.replace('ö','oe')
text = text.replace('Ä','Ae')
text = text.replace('Ü','Ue')
text = text.replace('Ö','Oe')
text = text.replace('ß','ss')
file2 = open("example2.txt", "x", encoding='utf8')
file2.write(text)
file2.close()
file.close()
```

Core Python: Real-World Example (Valentinas solution)

CRC 1412 REGISTER

```
#python3

f = open("berlinWiki.txt", "r")
text = f.read()
umlauts = ["ä", "ö", "ü", "Ä", "Ö", "Ü", "ß"]
replacements = ["ae", "oe", "ue", "Ae", "Oe", "Ue", "ss"]

for umlaut in umlauts:
    replacement = replacements[umlauts.index(umlaut)]
    text = text.replace(umlaut, replacement)
print (text)

file = open("newText.txt", "x")
file.write(text)
file.close
```

py -m pip install -r requirements.txt

Installing & using 3rd party modules

CRC 1412 REGISTER

- Tons of python-projects (modules/libraries) are available at https://pypi.org/ for almost every purpose and task (the search for “linguistics” returns 544 matches (!))
- They all have a unique „name“ and can easily be added to your python installation with a command-line tool called „pip“ (normally already installed with python)
- For example to install Excel support (read and write xls(x) files in python) simply type into your terminal window:

  python -m pip install openpyxl

- That’s it! All files are downloaded automatically and added to your python-installation
- With statement “import openpyxl” at the top of your script you can start working with Excel-Files!

py -m pip install -r requirements.txt

Real-World Example

CRC 1412 REGISTER

1. Please install pip on your computer
2. Please add spacy and openpyxl with pip to your python environment
3. Create a txt file in your Visual Studio with any kind text in English or German
4. Write a python script that reads the content of the file and stores it in a variable named “text”
5. Read the text with the spacy module and perform a part-of-speech tagging with your text
6. Create an Excel Sheet in your script and create a header in the first row:
   Text Lemma POS Tag Dep Shape alpha stop
7. Save the values of your part-of-speech tagging in this excel-file

py -m pip install -r requirements.txt

Real-World Example

CRC 1412 REGISTER

```
import spacy
from openpyxl import Workbook # imports only parts of module

nlp = spacy.load("de_core_news_sm") # load german language files
file = open("maerchen.txt", "r", encoding='utf8') # open text file
text = file.read() # read content of file
doc = nlp(text) # create DOC Object by using api (api, modules and libraries are meaning the same) method
                # (you always have to look it up how it works depending on lib)
                # Now Spacy scans the text and does the work
wb = Workbook() # Create Excel File
ws = wb.create_sheet("Goethes Märchen", 0) # One Excel-File has one or more sheets. Let's create a first sheet at position 0
headers = ['Text', 'Lemma', 'POS', 'Tag', 'Dep', 'Shape', 'alpha', 'stop'] # a list for the headers
ws.append(headers) # the append() method of Object Worksheet is easy to use.
                   # A list in python will be transformed to a new row (at the end) in excel
for token in doc:
    ws.append([token.text, token.lemma_, token.pos_, token.tag_, token.dep_,token.shape_, token.is_alpha, token.is_stop])
    # same from above applies here. Every token becomes a new line in excel with selected information from
ws.auto_filter.ref = ws.dimensions # we apply auto-filter function to all columns
wb.save('maerchen.xls') # we save excel-file at same directory
```

Python: Useful resources

CRC 1412 REGISTER

W3C tutorial
https://www.w3schools.com/python/default.asp

Thomas Krause on NLP
https://thomaskrause.github.io/nlp-mit-python/

Tutorial on GitHub
https://python-kurs.github.io/sommersemester_2019/intro.html

python.org official tutorial
https://docs.python.org/3/tutorial/index.html