INTRODUCTION TO PYTHON PROGRAMMING FOR GEOSCIENCE

CHRISTOPH PAULIK

CHRISTOPH.PAULIK@GEO.TUWIEN.AC.AT

COURSE ORGANIZATION

FOUR LECTURERS

- Hahn Sebastian
- Karel Wilfried
- Mandelburger Gottfried
- Paulik Christoph

HOW TO GET TO A GRADE

There will be three small projects.

- Short report
- Source code
- Short talk about the project with the lecturer.

TECHNICAL STUFF

- Customized Winpython installation will be provided
  - We had some Windows 8 problems in the past
- If you work on Linux I can help you with the setup if needed.
- Any OS X users?

COURSE MATERIALS

- Slides will be in TUWEL and probably Github (my parts at least)

VERSION CONTROL

Who of you has worked with version control

(CVS, SVN, Git …) ?

WE CAN NOT GO INTO DETAIL IN THIS COURSE.

Do yourselves a favor and learn Git. It will make your lives easier.

List of Tutorials

WHAT IS PYTHON

> *Python is an easy to learn, powerful programming language.*
>
> *– Python Tutorial first sentence*

Developed by Guide van Rossum , early 1990^{S}

- Benevolent Dictator For Life (BDFL) of the Python community.
- Worked at Google, now at Dropbox
- Named Python after Monthy Python

LANGUAGE OVERVIEW

General purpose

can write anything from websites (Youtube, Reddit…) to scientific code.

High Level

Strong abstraction from inner workings of the computer, e.g. no memory management. [wiki]

Interpreted

executes instructions directly, no compilation [wiki]

Multi paradigm

Object oriented, functional, imperative or procedural styles are supported

LANGUAGE OVERVIEW

Dynamically typed

Variable types are checked during execution.

Multi Platform

Works on Linux, Mac OS, Windows, …

Batteries included

Powerful standard library (e.g. file reading, URL parsing …)

DIFFERENT VERSIONS

Currently Python 2.7 and 3.4 are the latest versions.

Python 3 introduced some incompatible changes. Nowadays most 3rd party packages should work on both versions.

Python 2 or 3 Guide

List of most popular packages and Python 3 status

We use Python 2.7 in this course.

PYTHON SYNTAX

Whitespace is important

```
# set variable x to 1
x = 1
if x == 1:
"""
This condition checks if x is one and
prints it to stdout
"""
    print "x is one"
```

Use spaces if possible, be consistent in your code.

Check the Python Style Guide PEP8

VARIABLES

Variables can be assigned without type declaration.

```
a = 1
b = 2.8
c = "text"
print "a is", a
print "b is", a
print "c is", c
c = a
print "c is", c
```

```
a is 1
b is 1
c is text
c is 1
```

Variable types can be changed. Careful!

Variables are case sensitive

You can not use one of the predefined keywords. Should also not use python builtins.

```
var1 = 5
Var1 = 6
print var1, Var1
import keyword
print keyword.kwlist
```

```
5 6
['and', 'as', 'assert', 'break', 'class', 'continue', 'def', 'del', 'elif', 'el
se', 'except', 'exec', 'finally', 'for', 'from', 'global', 'if', 'import', 'in'
, 'is', 'lambda', 'not', 'or', 'pass', 'print', 'raise', 'return', 'try', 'whil
e', 'with', 'yield']
```

PYTHON AS A CALCULATOR

```
width = 20
height = 5 * 9
# calculate rectangular area
rect_area = width * height
print "Rectangular Area is", rect_area

# calcualte area of square
square_area = width ** 2
print "Square area is", square_area
```

```
Rectangular Area is 900
Square area is 400
```

DATA TYPES

These are some basic data types:

```python
integer = 1
float_number = 1.345
complex_number = 3 + 4j
string = "a text string"
a_list = [1, 1.34, string]
a_tuple = (1, 1.34, string) # immutable list
a_set = set("mama") # list of unique items
a_dict = {'jan': 1, 'feb': 2}
a_boolean = True
none_type = None
```

Official Python Documentation on data types

INTEGERS AND FLOATS

```
print 1+4
print type(1+4)
print type(1+4.) # automatic conversion of resulting data type
```

```
5
<type 'int'>
<type 'float'>
```

```
# careful with division
print "12 / 7 is", 12/7
print type(12/7)
print "12 / 7 really is", 12/7.
print type(12/7.)
```

```
12 / 7 is 1
<type 'int'>
12 / 7 really is 1.71428571429
<type 'float'>
```

COMPARISONS

```
#comparison
print "is 7 less than 5?" , 7<5
print "is 5 less than 7?" , 7>5
print "is 5 less or equal 5?" , 5<=5
```

```
is 7 less than 5? False
is 5 less than 7? True
is 5 less or equal 5? True
```

STRINGS

```
s1 = "Monthy"
s2 = 'Python' #single quotes are also fine
# joining string can be done in different ways
print s1+s2, s1*3, ";".join([s1, s2])
```

```
MonthyPython MonthyMonthyMonthy Monthy;Python
```

```
# sometimes double quotes are necessary
print 'This string doesn't work'
```

STRING SLICING

- slice syntax is [start:stop:step]
- start is inclusive, stop is exclusive

```
s = "0123456789"
print s[1:4]
print s[3:8]
print s[-1]
print s[-6:-3]
print s[0:5:2]
print "negative step reverts", s[::-1]
```

```
123
34567
9
456
024
negative step reverts 9876543210
```

LISTS

```
winter = ['jan', 'feb']
spring = ['apr', 'may', 'jun']
summer = ['jul', 'aug', 'sep']
autumn = ['oct', 'nov', 'dec']
# create one list containing all the elements
months = winter + spring + summer + autumn
print "List of months", months
# create a nested list, list of seasons
seasons = [winter, spring, summer, autumn]
print "List of seasons", seasons
```

```
List of months ['jan', 'feb', 'apr', 'may', 'jun', 'jul', 'aug', 'sep', 'oct',
'nov', 'dec']
List of seasons [['jan', 'feb'], ['apr', 'may', 'jun'], ['jul', 'aug', 'sep'],
['oct', 'nov', 'dec']]
```

What happens if we add the missing month of March?

```
winter.append('mar')
print "Winter is now:", winter
print "List of months", months
print "List of seasons", seasons
```

```
Winter is now: ['jan', 'feb', 'mar']
List of months ['jan', 'feb', 'apr', 'may', 'jun', 'jul', 'aug', 'sep', 'oct',
'nov', 'dec']
List of seasons [['jan', 'feb', 'mar'], ['apr', 'may', 'jun'], ['jul', 'aug', '
sep'], ['oct', 'nov', 'dec']]
```

Lists are generally stored as references -> the values are not copied.

LIST MANIPULATION AND SLICING

```
months.insert(2, 'mar') # insert a element before index 2
print months
print months[::2] # slicing works the same as with strings
print months[8:11]
```

```
['jan', 'feb', 'mar', 'apr', 'may', 'jun', 'jul', 'aug', 'sep', 'oct', 'nov', '
dec']
['jan', 'mar', 'may', 'jul', 'sep', 'nov']
['sep', 'oct', 'nov']
```

LIST MANIPULATION

```
li = [1, 4, 8.33, 3.6, 19, 12]
print li.index(3.6)  # get the index of a element
print li[li.index(3.6)]  # this index can be used to address the list
li.sort()  # sort the list
print li
#remove elements from list
del li[2]
print "removed index 2:", li
li.pop(2)
print "removed index 2 again:", li
```

```
3
3.6
[1, 3.6, 4, 8.33, 12, 19]
removed index 2: [1, 3.6, 8.33, 12, 19]
removed index 2 again: [1, 3.6, 12, 19]
```

for even more list related functions see the documentation

SETS - UNIQUE ELEMENTS AND SET OPERATIONS

```python
m, p = set('mama'), set('papa')
print m
print p
print "Union, m or p", m | p
print "Intersection, m and p", m & p
print "Difference, m minus p", m - p
print "Symetric Difference", m ^ p # elements in either one but not both sets
```

```
set(['a', 'm'])
set(['a', 'p'])
Union, m or p set(['a', 'p', 'm'])
Intersection, m and p set(['a'])
Difference, m minus p set(['m'])
Symetric Difference set(['p', 'm'])
```

More information in the documentation

DICTIONARIES

Key value pairs of any data type. Not ordered.

```
d = {'integer': 7, 'string': "test", 1: [1, 2, 3]}
print d
print d[1]
print d['integer'] # get a value by the key
print d.keys() # list of the keys
print 'integer' in d # test for presence of key
```

```
{'integer': 7, 'string': 'test', 1: [1, 2, 3]}
[1, 2, 3]
7
['integer', 'string', 1]
True
```

```python
# keys can be any hashable(unique) object
d1 = {(1, 2): "tuple with 1 and 2",
      (1, 3): "tuple with 1 and 3"}
print d1[(1, 2)]

# add element to dictionary
d1['additional element'] = "I am new"
print d1
# delete it again
del d1['additional element']
print d1
```

```
tuple with 1 and 2
{(1, 2): 'tuple with 1 and 2', 'additional element': 'I am new', (1, 3): 'tuple
 with 1 and 3'}
{(1, 2): 'tuple with 1 and 2', (1, 3): 'tuple with 1 and 3'}
```

Dictionary update

```
d1 = {"one": 1, "two": 2}
d2 = {"two": "II", "three": "III"}
d1.update(d2)
print d1
```

```
{'three': 'III', 'two': 'II', 'one': 1}
```

```
answer_dict = {True: 'yes', False: 'no'}
print "is 7 less than 5?" , answer_dict[False]
print "is 7 less than 5?" , answer_dict[7<5]
```

```
is 7 less than 5? no
is 7 less than 5? no
```