

efl Data Science Course

Introduction

Lecturers



M.Sc. Nicolas Pfeuffer

Research Assistant

Nicolas Pfeuffer studied Business Informatics at the Goethe-Universität Frankfurt (M.Sc.). During his master's program,...



M.A. Timo Schäfer

Research Assistant

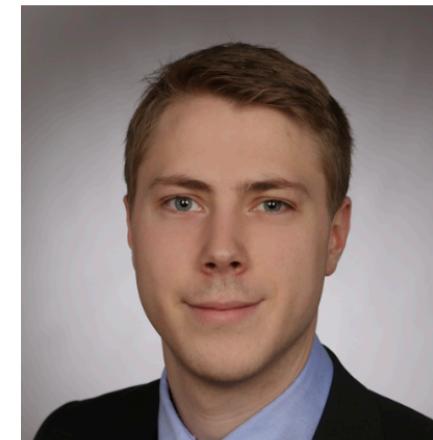
Timo Schäfer received his Master in Banking&Finance and Data Science from the University of Zurich in February 2018 and...



**M.Sc. Benjamin M.
Abdel-Karim**

Research Assistant

Since March 2018, Benjamin M. Abdel-Karim is a research assistant of Prof. Dr. Oliver Hinz at the Chair of Information...



M.Sc. Jens Lausen

Research Assistant

Jens Lausen received a Bachelor's degree in Management and Economics and a Master's degree in Management from Johannes...



The efl



Industry-academic partnership

Universities



Sponsors



The efl



Original Mission:

- Investigate and co-shape Digital Finance 2.0
 - Web-based selfservices of customers
- Research was performed in three different Layers:
 - Customers in E-Finance
 - E-Financial Markets and Market Infrastructures
 - IT Infrastructures: Service Systems in E-Finance

New Mission (Since 2019):

- Use expertise in Data Science to deliver cutting edge research in the fields of
 - Financial Services
 - Retail & Marketing
 - Health
 - Law

Teaching

Day 1 (Python Course) (07.10.2019)

9:00 - 10:30 Uhr

Python Basics

Introduction and Primitive Data Types

10:40 - 12:10 Uhr

Data Structures

Lists, Sets, Dictionaries

13:30 - 15:00 Uhr

Control Structures

Loops (for, while), case distinction (if, else)

15:15 - 16:45 Uhr

Functions

Structure of Functions and Application

Day 2 (Python Course) (08.10.2019)

9:00 - 10:30 Uhr

Helpful functions for data processing

Libraries: os, re, csv

10:40 - 12:10 Uhr

Data types and data structures

Libraries: numpy, pandas

13:30 - 15:00 Uhr

Data import and visualization

Libraries: csv (cont'd), matplotlib

15:15 - 16:45 Uhr

Outlook: Data Science

Exemplary implementation of a KDD process

Day 3 (Data Science) (11.10.2019)

9:00 - 10:30 Uhr

Introduction to Data Science

Terminology and basic concepts

10:40 - 12:10 Uhr

Working with Data

Preprocessing, explorative data analysis

13:30 - 15:00 Uhr

Data Analysis I

Classification

15:15 - 16:45 Uhr

Data Analysis II

Neural Networks

Teaching

Day 1 (Python Course) (07.10.2019)	Day 2 (Python Course) (08.10.2019)	Day 3 (Data Science) (11.10.2019)
9:00 - 10:30 Uhr Python Basics Introduction and Primitive Data Types	9:00 - 10:30 Uhr Helpful functions for data processing Libraries: os, re, csv	9:00 - 10:30 Uhr Introduction to Data Science Terminology and basic concepts
10:40 - 12:10 Uhr Data Structures Lists, Sets, Dictionaries	10:40 - 12:10 Uhr Data types and data structures Libraries: numpy, pandas	10:40 - 12:10 Uhr Working with Data Preprocessing, explorative data analysis
13:30 - 15:00 Uhr Control Structures Loops (for, while), case distinction (if, else)	13:30 - 15:00 Uhr Data import and visualization Libraries: csv (cont'd), matplotlib	13:30 - 15:00 Uhr Data Analysis I Classification
15:15 - 16:45 Uhr Functions Structure of Functions and Application	15:15 - 16:45 Uhr Outlook: Data Science Exemplary implementation of a KDD process	15:15 - 16:45 Uhr Data Analysis II Neural Networks



efl -
the Data Science Institute

CERTIFICATE OF
PARTICIPATION

**INTRODUCTION TO
PYTHON FOR DATA
SCIENCE**

OCT · 09 · 2019

- Completion of tasks from the last hour
- Submission: Description of how the tasks were solved
- Formalities:
 - Min. 2 pages (Arial 11, 1.5 Line spacing , 3 CM Correction margin)
 - Code must be delivered separately (code folder)

Course Material?

The screenshot shows a GitHub repository page. At the top, there's a banner with the text "Learn Git and GitHub without any code!" and a "Read the guide" button. To the right, a blue callout box says "You can download the course material as .zip file". Below the banner, the repository details are shown: owner "efl-the-data-science-institute", name "ds-courses", 3 forks, 0 stars, and 0 releases. The repository description states it's for data science courses. At the bottom, stats are provided: 4 commits, 1 branch, 0 releases, and 3 contributors.

Learn Git and GitHub without any code!

Using the Hello World guide, you'll start a branch, write comments, and open a pull request.

Read the guide

You can download the course material as .zip file

efl-the-data-science-institute / **ds-courses**

Watch 3 | Star 0 | Fork 1

Code Issues 0 Pull requests 0 Projects 0 Wiki Security Insights

This is the repository for the material of the data science courses organized by the efl - the Data Science Institute.

4 commits 1 branch 0 releases 3 contributors

<https://github.com/efl-the-data-science-institute/ds-courses>

Why you are here

Kenntnis in Python und anderen Softwareprogrammen werden auf dem heutigen **Arbeits- und Praktikumsmarkt oft vorausgesetzt**. Um auch über die in den Vorlesung vermittelten Inhalte hinaus etwas über statistische Softwareprogramme zulernen, möchte ich an dem Kurs teilnehmen.

Ich möchte mich in diesem Bereich fortbilden, um bei einer Bachelor-Arbeit Daten auszuwerten und zu visualisieren. Darüberhinaus möchte ich wahrscheinlich **einen quantitativ-orientierten Master belegen und halte es daher für sinnvoll einen ersten Python-Kurs zu belegen.**

Auffrischen von Python, neue Einblicke in Software

Bei meiner aktuellen Werksstudentenstelle merke ich immer wieder, wie Entscheidungen sehr oft datengetrieben werden und welche Rolle gut aufgearbeitete Daten spielen. Daher würde ich gerne, um mich beruflich und persönlich weiterzubilden, gerne an dem Kurs teilnehmen.

Why Coding?



NACH ABSCHLUSS IN ...
**INFORMATIK, MATHEMATIK,
WIRTSCHAFTSINFORMATIK**

TOP BERUFSTITEL*

IT Manager
92.275 €

IT-Projektmanager/in
86.695 €

Senior Developer
78.635 €

Analyst
74.469 €

Software-Architekt/in
74.152 €

TOP BRANCHE

Banken
85.067 €

Chemie- und Erdöl-
verarbeitende Industrie
84.033 €

Konsum- und
Gebrauchsgüter
82.586 €

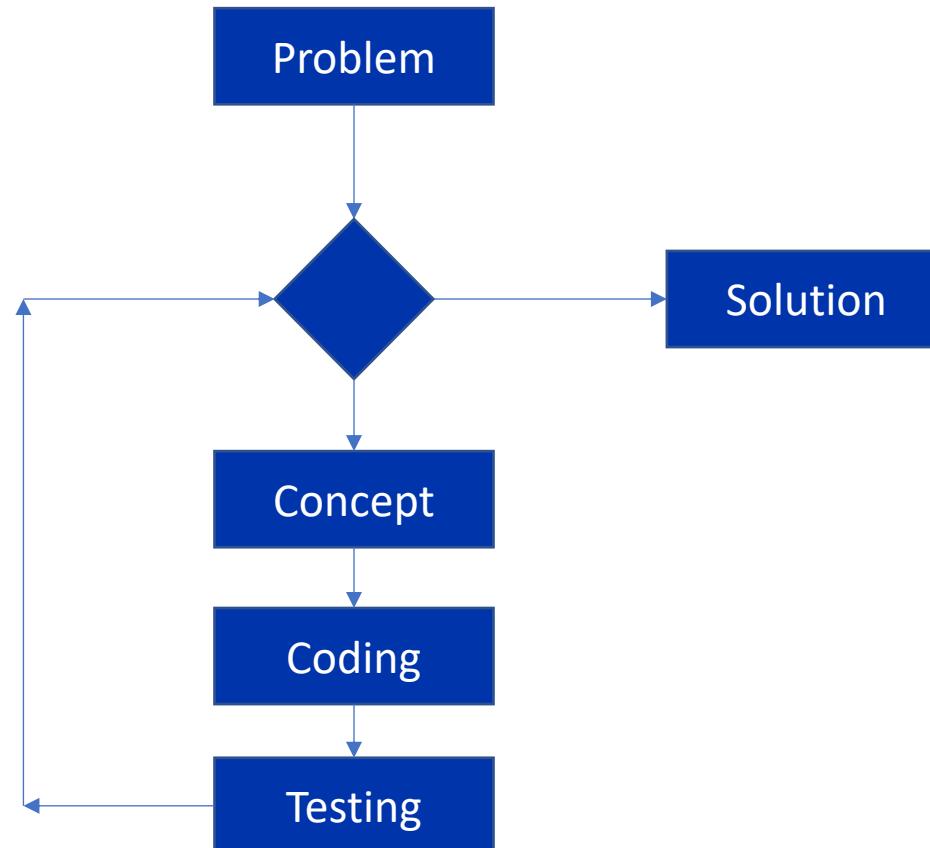
Unternehmensberatung,
Wirtschaftsprüfung und Recht
82.336 €

Finanzdienstleister
81.331 €



*Berufstitel ohne Management- & Personalverantwortung
Bruttodurchschnittsgehalt (inkl. variabler Bezüge), Mittelwert

What is coding?



Why Python?

- Released in 1991 by Guido van Rossum
- With the explosive growth of ‘big data’ in disciplines such as bioinformatics, neuroscience and astronomy, programming know-how is becoming ever more crucial (Perkel 2015, p. 125).



[2]



[3]



[6]



[4]



[8]



[7]

The elementary basics in Python

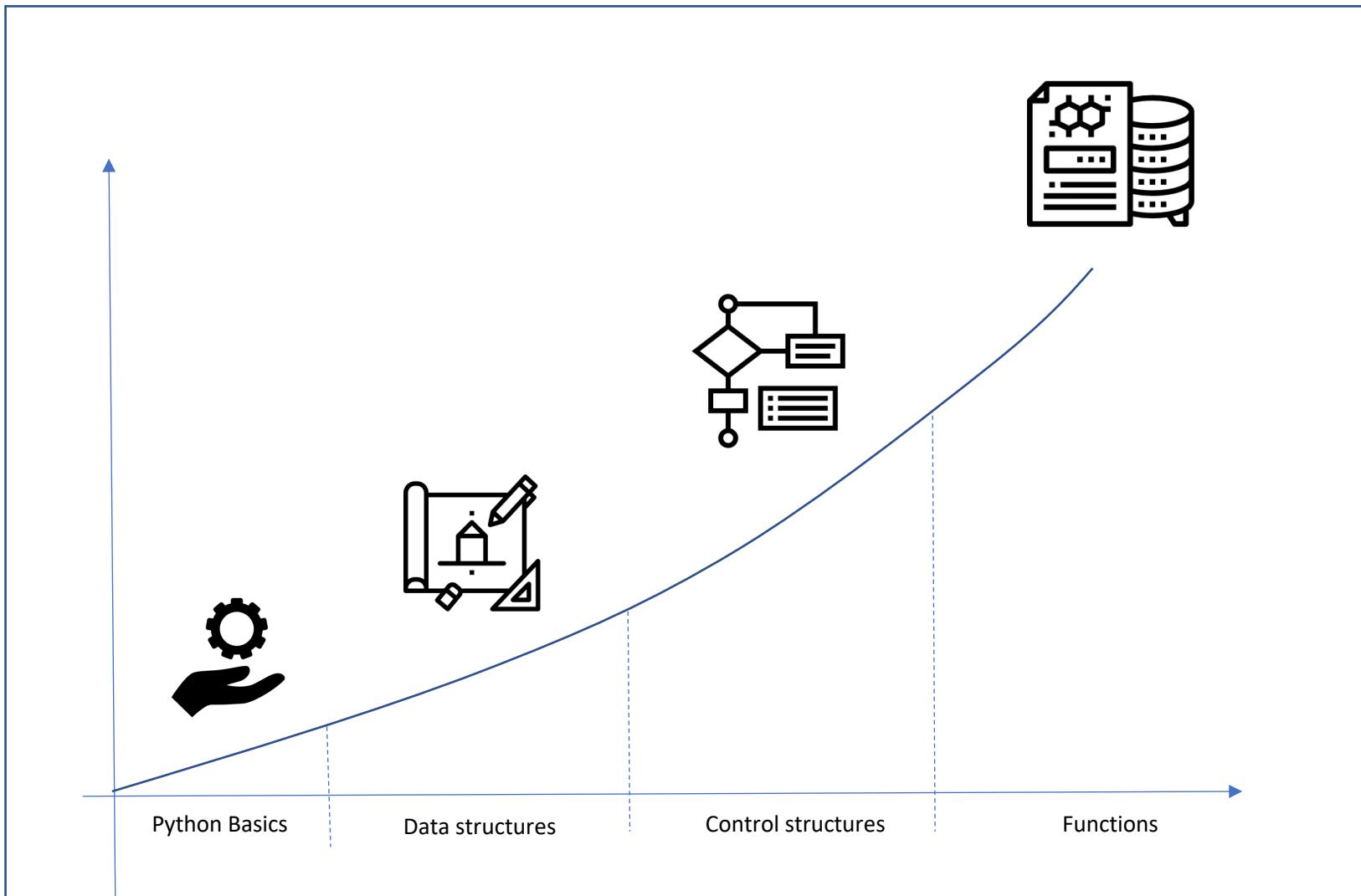
- Philosophy and difference to conventional programming languages
 - Higher programming language
 - It's simple
 - Fast to read
 - Structuring by indenting
 - No {} or ; => Faster to Code
 - Data types are managed dynamically. There is no static type check like in java
 - Widespread in science
 - Extensive Support Libraries (important data science, math and many more)
 - Integration Feature
 - Productivity (Many Frameworks such as unit testing)

Introduction goals

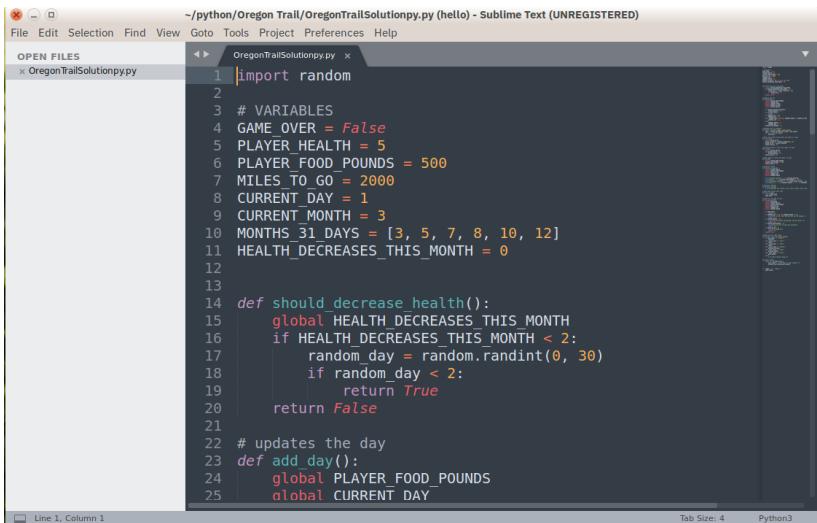


- Introduction to programming
- For beginners
- The module is interactive! Use your computer
- **We develop the solutions together!**
- Please be on time
- There are no dumb questions
- Nobody knows everything
- Copying solutions is plagiarism

Learning Curve for Today

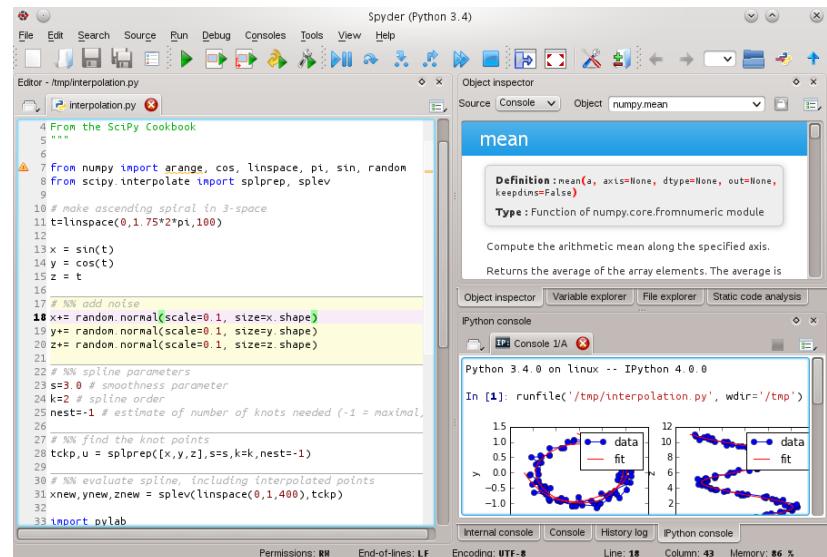


IDE a deeper look? The Problem of Choice



A screenshot of the Sublime Text editor. The title bar says "File Edit Selection Find View Goto Tools Project Preferences Help". The tab bar shows "OregonTrailSolution.py (hello) - Sublime Text (UNREGISTERED)". The code in the editor is:

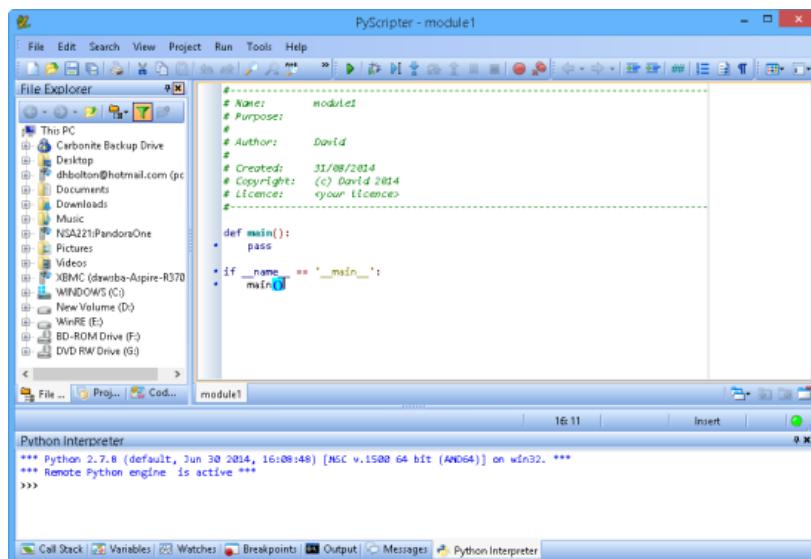
```
1 import random
2
3 # VARIABLES
4 GAME_OVER = False
5 PLAYER_HEALTH = 5
6 PLAYER_FOOD_POUNDS = 500
7 MILES_TO_GO = 2000
8 CURRENT_DAY = 1
9 CURRENT_MONTH = 3
10 MONTHS_31_DAYS = [3, 5, 7, 8, 10, 12]
11 HEALTH_DECREASES_THIS_MONTH = 0
12
13
14 def should_decrease_health():
15     global HEALTH_DECREASES_THIS_MONTH
16     if HEALTH_DECREASES_THIS_MONTH < 2:
17         random_day = random.randint(0, 30)
18         if random_day < 2:
19             return True
20     return False
21
22 # updates the day
23 def add_day():
24     global PLAYER_FOOD_POUNDS
25     global CURRENT_DAY
```



A screenshot of the Spyder Python IDE. The title bar says "File Edit Search Source Run Debug Consoles Tools View Help". The main window shows a script named "interpolation.py" with code:

```
4 From the SciPy Cookbook
5 ***
6
7 from numpy import arange, cos, linspace, pi, sin, random
8 from scipy.interpolate import splprep, splev
9
10 # make ascending spiral in 3-space
11 t=linspace(0,1.75*2*pi,100)
12
13 x = sin(t)
14 y = cos(t)
15 z = t
16
17 # %% add noise
18 x+= random.normal(scale=0.1, size=x.shape)
19 y+= random.normal(scale=0.1, size=y.shape)
20 z+= random.normal(scale=0.1, size=z.shape)
21
22 # %% spline parameters
23 s=3.0 # smoothness parameter
24 k=2 # spline order
25 nest=-1 # estimate of number of knots needed (-1 = maximal,
26 nest)
27 # %% find the knot points
28 tckp,u = splprep([x,y,z],s=s,k=k,nest=-1)
29
30 # %% evaluate spline, including interpolated points
31 xnew,ynew,znew = splev(linspace(0,1.400),tckp)
32
33 import pylab
```

The right side of the interface has a "mean" object inspector panel showing the definition of the mean function from numpy.



A screenshot of the PyScripter IDE. The title bar says "File Edit Search View Project Run Tools Help". The left pane shows a "File Explorer" with a tree view of the file system. The right pane shows a code editor with a file named "module1.py" containing:

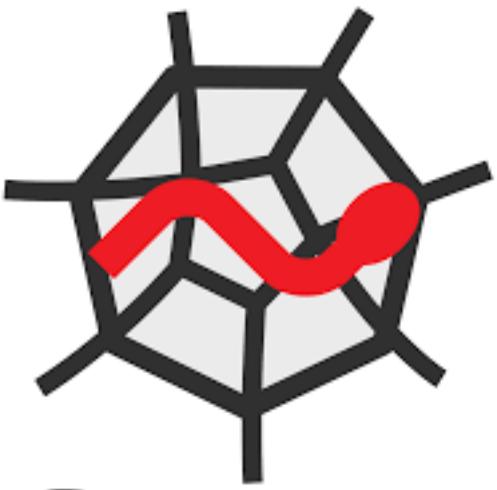
```
# Name: modules
# Purpose:
# Author: David
# Created: 31/08/2014
# Copyright: (c) David 2014
# Licence: your Licence
#
def main():
    pass

if __name__ == '__main__':
    main()

Python Interpreter
*** Python 2.7.8 (default, Jun 30 2014, 16:08:48) [MSC v.1500 64 bit (AMD64)] on win32 ***
>>>
```

At the bottom, there are tabs for Call Stack, Variables, Watches, Breakpoints, Output, and Messages.

What We Use for This Course



SPYDER

The screenshot shows the Spyder Python IDE interface. On the left is a code editor with a large amount of Python code. The code includes imports from numpy, pandas, and scikit-learn, and demonstrates various array operations like concatenation and reshaping. In the center, there's a detailed documentation sidebar for the `array` class, listing parameters like `object`, `dtype`, `copy`, and `order`. On the right is a Python console window showing the execution of several commands, including running a file and displaying its tracebacks.

```
File Edit Search Source Run Debug Cnsoles Projects Tools View Help
C:/Users/C. A. M. Gerlach/Documents/dev/TestPackage/temp.py
temp.py ① test_mainwindow.py - SpyderDev..._Tests ② mainwindow.py ③ pil_patch.py ④ UIQ ⑤ mainwindow.py ⑥

291
292     sklearn.preprocessing.imputation.Imputer
293
294     * class testClass():
295         test = "This is a test class"
296         new_instance = testClass()
297
298
299     # raise Exception
300     type_of_string = "silly"
301     subtype = "very"
302     n_cans = 42
303     n_carton = 9
304
305     test_string = ("This is a {0} {1} string. I have {2} cans of spam and {3} carton"
306                   .format(type_of_string, n_cans, n_carton, subtype))
307
308
309     a %%
310
311     array_int = np.array([1, 2, 3], dtype=np.int64)
312     array_ID = np.array([1, 2, 5, 3, -5, 6, 2, 84], dtype=np.float32)
313     array_2D = np.array([[1, 2, 3], [5, 6, 7]], dtype=np.int64)
314     array_2D = np.array([[1, 2, -3], [5, -6, 7]], [[1, 2, 3], [5, 6, 7]]], dtype=np.int32)
315
316
317     * array_bool = np.array([[True, False, True], [True, True, False]])
318     [[True, False, True], [True, True, False]]
319
320     array_int8 = np.array([1, -2, 3], [-5, 6, -7]), dtype=np.int8
321     array_float32 = np.array([1, -2, 3]), dtype=np.float32
322     array_int32 = np.array([1, -2, 3]), dtype=np.int32
323
324     df = pd.DataFrame(["Ints": [0, 0, 0], "Bools": [True, False, True]])
325     df_complex = pd.DataFrame([0, 1, 2, 3, 4], dtype=np.complex128)
326     series = pd.Series(["test_series": [1, 2, 3, 4]])
327     list1 = [df, array_2D]
328     test_none = None
329
330     + long_text = ("This is some very very very long text! "
331                   "But Spyder can show it all!")
332
333
334     a %%
335
336
337     * test_df = pd.read_csv("C:/Users/C. A. M. Gerlach/Documents/NOAA/VerifR/Data/All")
338     test_df.loc[test_df['City'] == "Texas who Stupid", 'City']
339
340     df = pd.DataFrame({'Col1':[1,2,3], 'Col2':['a','b','c']})
341     df2 = df.copy()
342
343
344     * Imputer nan in a dataframe, if it is string then replace nan by mode else replace
345     def imputerNaN(df):
346
347
348         Parameters
349         -----
350         df : TYPE
351             DESCRIPTION.
352
353
354         a %%
```

Definition: array(object, dtype=None, copy=True, order='C', subok=False, ndmin=0)

Create an array.

Parameters

object : array_like

An array, any object exposing the array interface, an object whose `__array__` method returns an array, or any nested sequence.

dtype : data-type, optional

The data type of the array. If not given, then the type will be determined as the minimum type required to hold the objects in the sequence. This argument can only be used to 'upcast' the array. For downcasting, use the `asarray()` method.

copy : bool, optional

If true (default), then the object is copied. Otherwise, a copy will only be made if `__array__` returns a copy, if obj is a nested sequence, or if a copy is needed to satisfy any of the other requirements (`dtype`, `order`, etc.).

order : {'C', 'F', 'A'}, optional

Specify the memory layout of the array. If object is not an array, the newly created array will be in C order (row major) unless `F` is specified, in which case it will be in Fortran order (column major). If object is an array the following holds.

order	no copy	copy=True
'C'	unchanged	F & C order preserved, otherwise most similar order
'A'	unchanged	F order if input is F and not C, otherwise C order

Variable explorer Help Plot File Find Breakpoints Code Analysis

In [3]: runfile('C:/Users/C. A. M. Gerlach/Documents/dev/TestPackage/temp.py')
[C:/Users/C. A. M. Gerlach/Documents/dev/TestPackage/temp.py]
Traceback (most recent call last):

File "C:/Users/C. A. M. Gerlach/Documents/dev/TestPackage/temp.py", line 1, in module
runfile("C:/Users/C. A. M. Gerlach/Documents/dev/TestPackage/temp.py")

File "C:/Users/C. A. M. Gerlach/Documents/dev/TestPackage/temp.py", line 1, in runfile
execfile(filename, namespace)

File "C:/Users/C. A. M. Gerlach/Documents/dev/TestPackage/temp.py", line 1, in execfile
exec(compile(f.read()), filename, "exec"), namespace)

File "C:/Users/C. A. M. Gerlach/Documents/dev/TestPackage/temp.py", line 35, in module
raise Exception()

Exception

In [3]:

In [3]: runcell('Test Cell Name', 'C:/Users/C. A. M. Gerlach/Documents/dev/TestPackage/temp.py')

In [4]: df = pd.DataFrame({'Col1':[1,2,3], 'Col2':['a','b','c']})

In [4]:

Internal console [Python console] History

P master [42] | Line 16, Col 1 | UFF-B | CRLF | Re | Mem 42% | CPU 12%

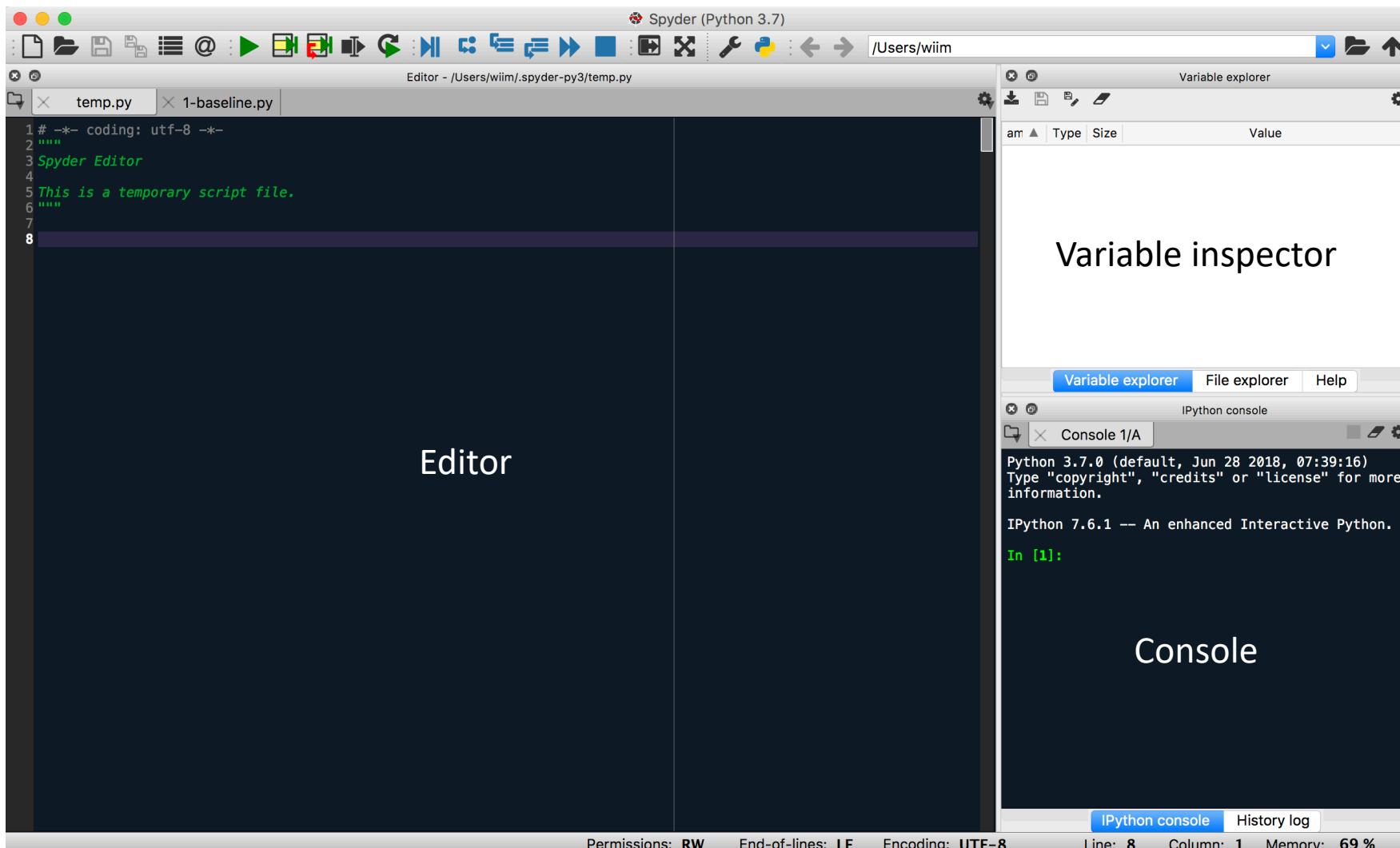
Practice and Questions

Take your computer and let's get started!

Create a Folder 'Day 1' on our Desktop

Launch Spyder and select the folder

Our Tool



Hello World

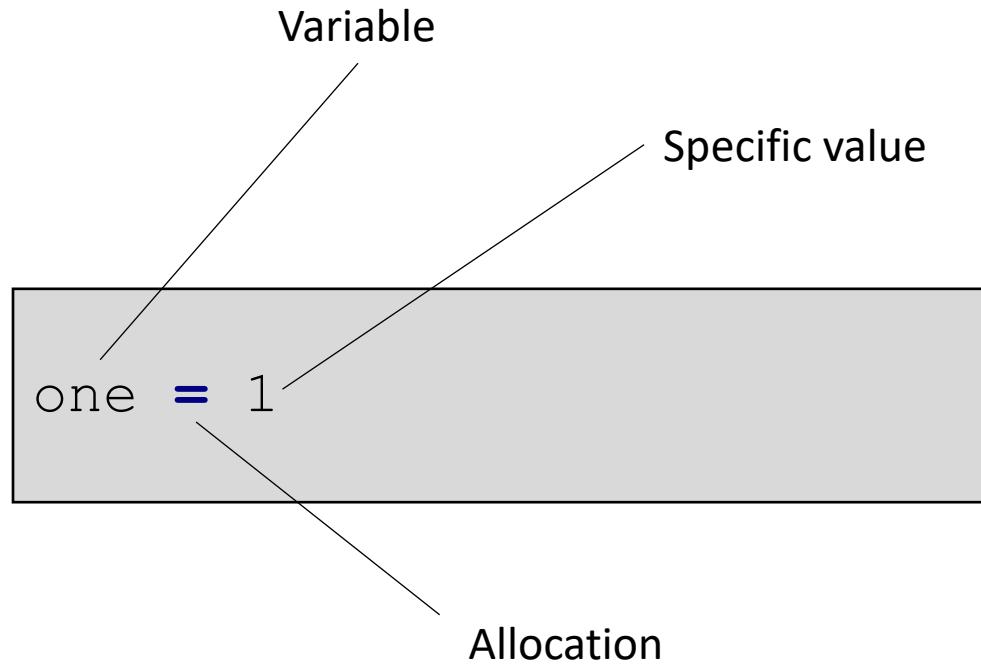
- Hello World
- Let's try our first example!
- Display "Hello World" on the console

Hello World

```
print('Hello World')
```

Abstraction (Computer Science)

The first step - The concept of variable



We assign the value 1 to the variable 'one'.

Now we can continue working with the variable 'one'.

Advantage: We are independent of concrete value

The essence of abstractions is preserving information that is relevant in a given context, and forgetting information that is irrelevant in that context.

– John V. Guttag



Numbers in Python



Integer:

- int (signed integers) – They are often just called integers or ints, are positive or negative whole numbers with no decimal point.

```
iOne = 1
```

Integer:

- Create two variables 'iOne' and 'iTTwo'. Assign the values 2 and 5 to each of these variables.
- Perform the following operations with the variables:
 - iMulti is the multiplication of the two variables.
 - iAdd is the addition of the two variables.
 - iSub is the subtraction of the two variables. Started with iOne.
 - iDiff is the division of iOne by iTTwo.

Integer Action

```
iMulti = iOne * iTwo
print(iMulti)
```

```
#iAdd is the addition of the two variables.
iAdd = iOne + iTwo
print(iAdd)
```

```
# iSub is the subtraction of the two variables. Started with iOne.
iSub = iOne - iTwo
print(iSub)
```

```
# iDiff is the division of iOne by iTwo.
iDiff = iOne / iTwo
print(iDiff)
```

Introduction to Primitive Data Types

Float:

- Float (floating point real values, **double**) – Also called floats, they represent real numbers and are written with a decimal point dividing the integer and fractional parts. Floats may also be in scientific notation, with E or e indicating the power of 10.
- Lets see how we can use float in python code.

```
dNumber = 1.0
```

Integer:

- Create two variable 'dOne' and 'dTwo'. Assign the values 2.5 and 5.5 to each of these variables.
- Perform the following operations with the variables:
 - dMulti is the multiplication of the two variables.
 - dAdd is the addition of the two variables.
 - dSub is the subtraction of the two variables. Started with dOne.
 - dDiff is the division of dOne by dTwo.

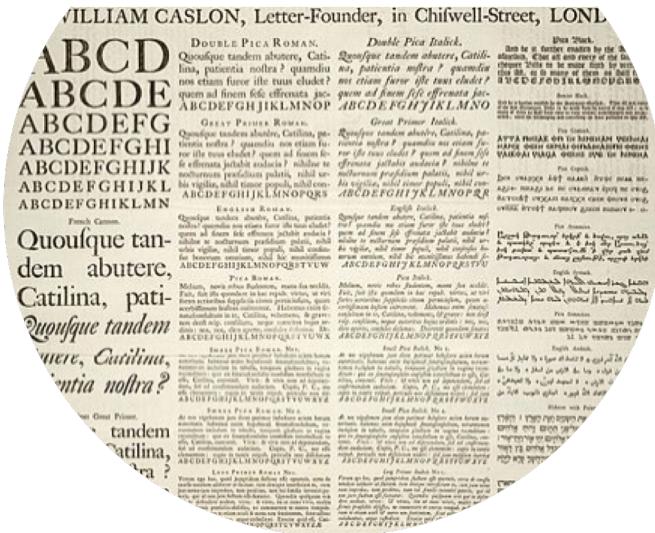
Float Action

```
# dMulti is the multiplication of the two variables.  
dMulti = dOne * dTwo  
print(dMulti)
```

```
#dAdd is the addition of the two variables.  
dAdd = dOne + dTwo  
print(dAdd)
```

```
# dSub is the subtraction of the two variables. Started with dOne.  
dSub = dOne - dTwo  
print(dSub)
```

```
# dDiff is the division of dOne by dTwo.  
dDiff = dOne / dTwo  
print(dDiff)
```



Symbols in Python



Chars

Char:

- Chars are symbols!
- Char is short for character.
- All characters in all languages can be represented. This representation is in the **Unicode** format.
- Unicode is a computer encoding methodology that assigns a unique number for every character. It doesn't matter what language or computer platform it's on.
- Lets look at a new script!

Strings are missing!

U+000A	10	<<control>
U+000B	11	<<control>
U+000C	12	<<control>
U+000D	13	<<control>
U+000E	14	<<control>
U+000F	15	<<control>
U+0010	16	<<control>
U+0011	17	<<control>
U+0012	18	<<control>
U+0013	19	<<control>
U+0014	20	<<control>
U+0015	21	<<control>
U+0016	22	<<control>
U+0017	23	<<control>
U+0018	24	<<control>
U+0019	25	<<control>
U+001A	26	<<control>
U+001B	27	<<control>
U+001C	28	<<control>
U+001D	29	<<control>
U+001E	30	<<control>
U+001F	31	<<control>
U+0020	32	SPACE
U+0021	33	EXCLAMATION MARK
U+0022	34	QUOTATION MARK
U+0023	35	NUMBER SIGN
U+0024	36	DOLLAR SIGN
U+0025	37	PERCENT SIGN
U+0026	38	AMPERSAND
U+0027	39	APOSTROPHE
U+0028	40	LEFT PARENTHESIS
U+0029	41	RIGHT PARENTHESIS
U+002A	42	ASTERISK
U+002B	43	PLUS SIGN
U+002C	44	COMMA
U+002D	45	HYPHEN-MINUS
U+002E	46	FULL STOP
U+002F	47	SOLIDUS
U+0030	48	DIGIT ZERO
U+0031	49	DIGIT ONE
U+0032	50	DIGIT TWO
U+0033	51	DIGIT THREE
U+0034	52	DIGIT FOUR
U+0035	53	DIGIT FIVE
U+0036	54	DIGIT SIX
U+0037	55	DIGIT SEVEN
U+0038	56	DIGIT EIGHT
U+0039	57	DIGIT NINE
U+003A	58	COLON

A unique number for every character.

- Try to get the unique number of ‘A’ as Unicode.

A unique number for every character.

- Try to get the unique number of ‘A’ as Unicode.

```
print(ord(char))
```

From Chars to Strings

- Strings are a sequence of chars.
- We can create them simply by enclosing characters in quotes. “Hello World” is a String!
- Therefore strings in Python are bytes representing Unicode characters.
- **In Detail:** Python does not have a character data type, **a single character is simply a string with a length of 1.**

Create two string variables

- Create two string variables:
 - `sWordOne = 'I Love'`
 - `sWordTwo = 'data'`
- Let's try to build a sentence with these two variables

Concatenated string

```
# Concatenated string
sWordOne = 'I Love'
sWordTwo = 'data'
sStatement = sWordOne + ' ' + sWordTwo
print(sStatement)
```

Built-in String Methods

- Python has a set of built-in methods that you can use on strings.
- How often does the word love appear in this sentence?

Note:

All string methods returns new values.
They do not change the original string.

Outlook: What functions are we will learn later in detail!

Here are just a few useful string operations for now!

Built-in String Methods – Count()

```
iCountSubStrings = sStatement.count('Love')  
print(iCountSubStrings)
```

Lower Case

- I would like to write everything in lower case
- Do I have to rewrite everything now?
- No, thanks to Built-in Methods

Built-in String Methods – lower()

```
sStatementLower = sStatement.lower()  
print(sStatementLower)
```

\circ	$\neg_3 p$	p	$\neg_3^* p$	\wedge_3	W	U	F
W	F	W	F	W	W	U	F
U	U	U	F	U	U	U	F
F	W	F	W	F	F	F	F

\vee_3	W	U	F	\rightarrow_3	W	U	F
W	W	W	W	W	W	U	F
U	W	U	U	U	W	U	U
F	W	U	F	F	W	W	W

	W	U	F	\leftrightarrow_3^*	W	U
W	U	F		W	W	F
U	U			U	F	
F				F		

Boolean



Boolean and logical operators

- Boolean variable
- In computer science, the Boolean data type is a data type that has one of two possible values
- They are used to represent truth values (false or true).
- They are helpful for logical operations.

```
bBoolean = True
```

Logical operations

- We can perform logical operations with True and False
- Let's try it: Execute all combinations with True and False with **and-operation**
- Tip: 2^2 possible combinations

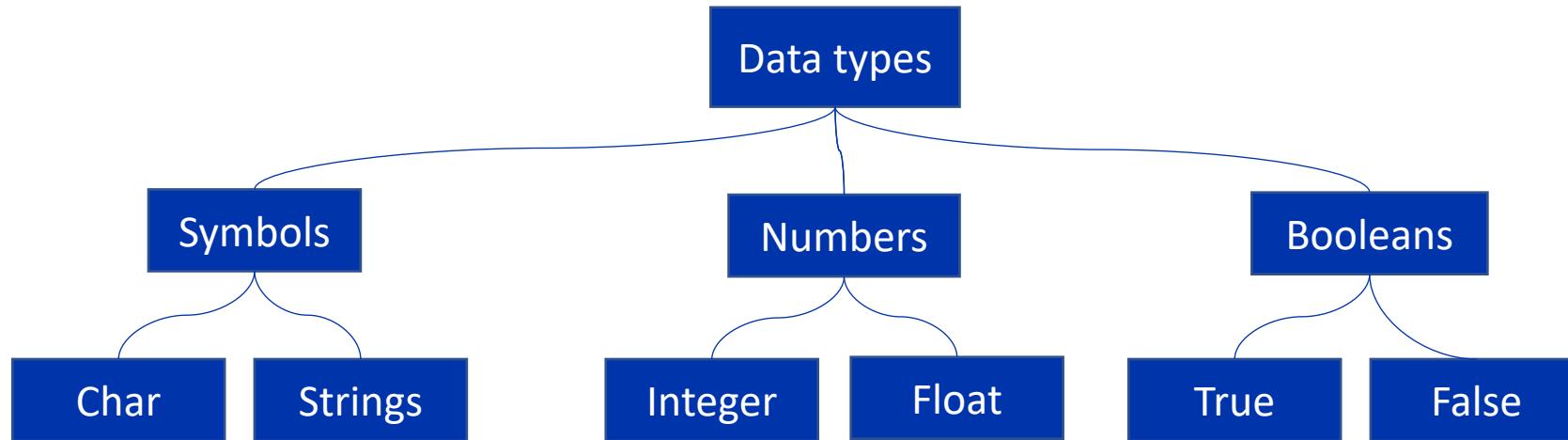
Logical operations

```
bTrue = True
bFalse = False
print(bTrue)
print('True and True is:', bTrue and bTrue)
print('False and False is:', bFalse and bFalse)
print('True and False is:', bTrue and bFalse)
print('True and False is:', bTrue and bFalse)
```

* For the sake of completeness: or is also a operator for True and False.

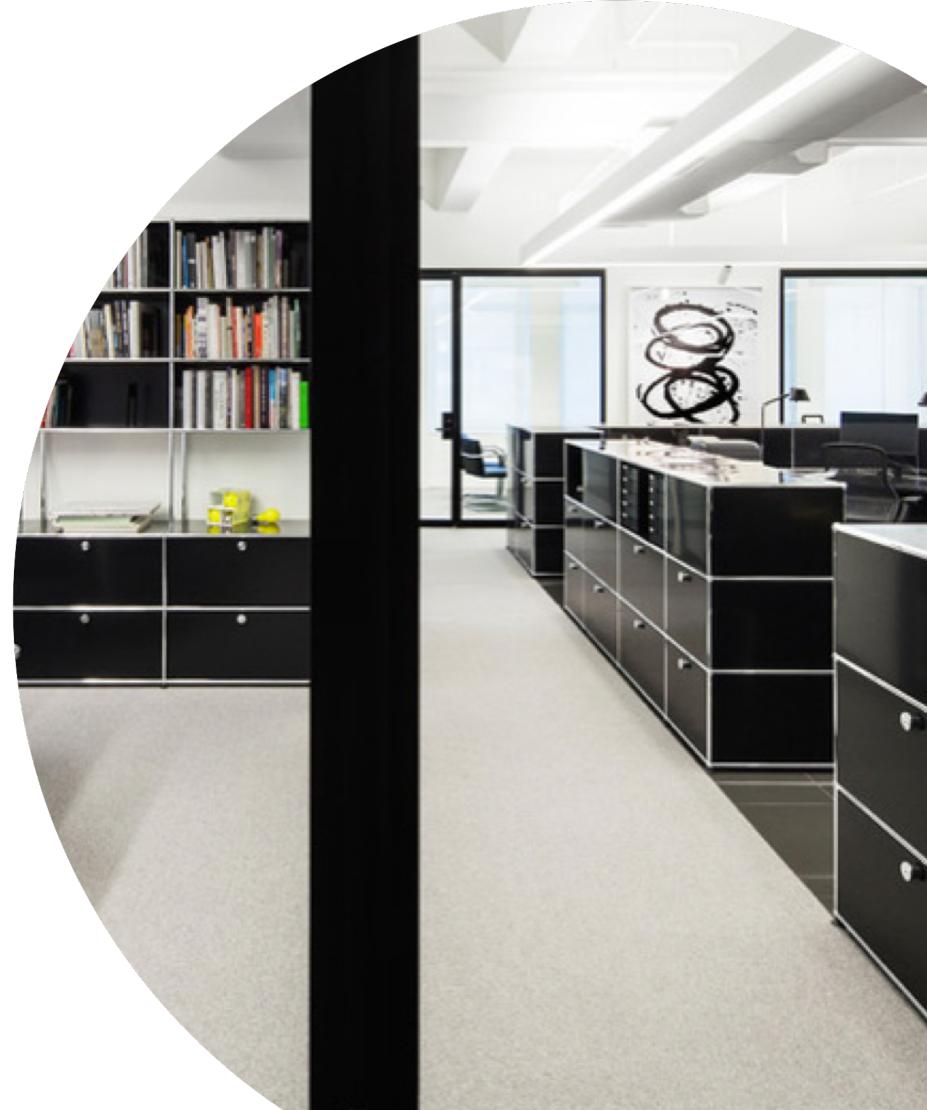
Introduction to Primitive Data Types

- Systematization of primitive data types





Formalities



- Source Code Documentation
- @author: name
- @since: first implementation date
- @version: date of last update
- @source: if you using links etc.
- @code: special code note
- @param: if special parameter is used or you have to describe.

```
# I am a comment
@author: unknown
@since: 20190929
@version: beta1
print('Hello World')
```

Naming convention

- Names of attributes, variables, methods start with a small letter
 - may use letters without ß or similar
 - which points to the data type like i, s or l
- This is standard in professional software development
- Camel Case: Compound words are written in programming language. Every new word is capitalized
 - Name = Is the name of...
 - bScriptName = Simple code file that does something
 - CName = Class (later more)
 - dName = Variable that saves a floating point (double)
 - iName = Variable that saves an integer value
 - sName = Variable that saves a string value
 - bName = Boolean for true or false values
 - LName = Object from type list
 - fName = Self-written function

Naming convention and comments

- Names of attributes, variables, methods start with a small letter
 - may use letters without ß or similar
 - which points to the data type like i, s or l
- This is standard in professional software development.
- Camel Case: Compound words are written in programming language. Every new word is capitalized.

<https://www.python.org/dev/peps/pep-0008/>

Naming convention, but python ...

▲ for everything related to Python's style guide: i'd recommend you read [PEP8](#).

175 To answer your question:

▼ Function names should be lowercase, with words separated by underscores as necessary to improve readability.

[12]



<https://www.python.org/dev/peps/pep-0008/>

Code is some kind of Art. Therefore...

But we are using the presented naming convention

These are all approaches. Find your own style!