

efl Data Science Course

Introduction

The efl



Industry - Academic Partnership

Universities



Sponsors

FACTSET



Deutsche Leasing

finanz informatik



DZ BANK Gruppe

Who we are



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
 - General, cross-sectional research

Lecturers



M.Sc. Tino Cestonaro

Research Assistant

Tino Cestonaro joined the team in April 2020. His research focuses on Market Microstructure, Financial Machine Learning,...



M.Sc. Micha Bender

Research Assistant

Micha Bender focuses on empirical financial market research. Furthermore, he is interested in the application of machine...



M.Sc. Johannes Chen

Research Assistant

Johannes Chen studied Business Informatics at the Technical University of Darmstadt. During his studies, he focused on...



Dr. 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...



Agenda

Teaching

Day 1 (Python Course) (04.11.2022)

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) (11.11.2022)

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) (18.11.2022)

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

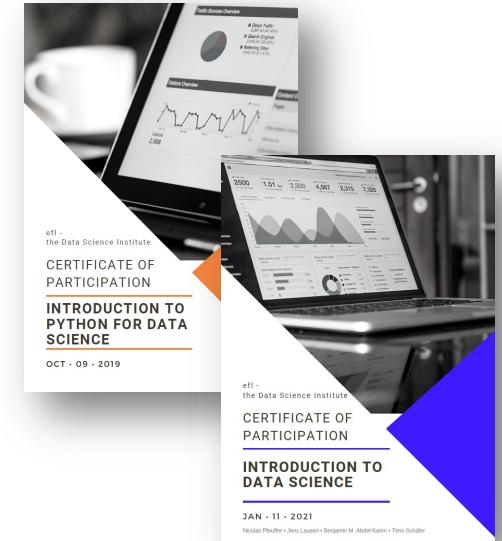
Data Analysis I

Classification

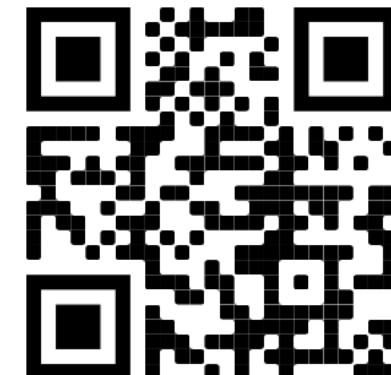
15:15 - 16:45 Uhr

Data Analysis II

Neural Networks



<https://www.eflab.de/teaching>

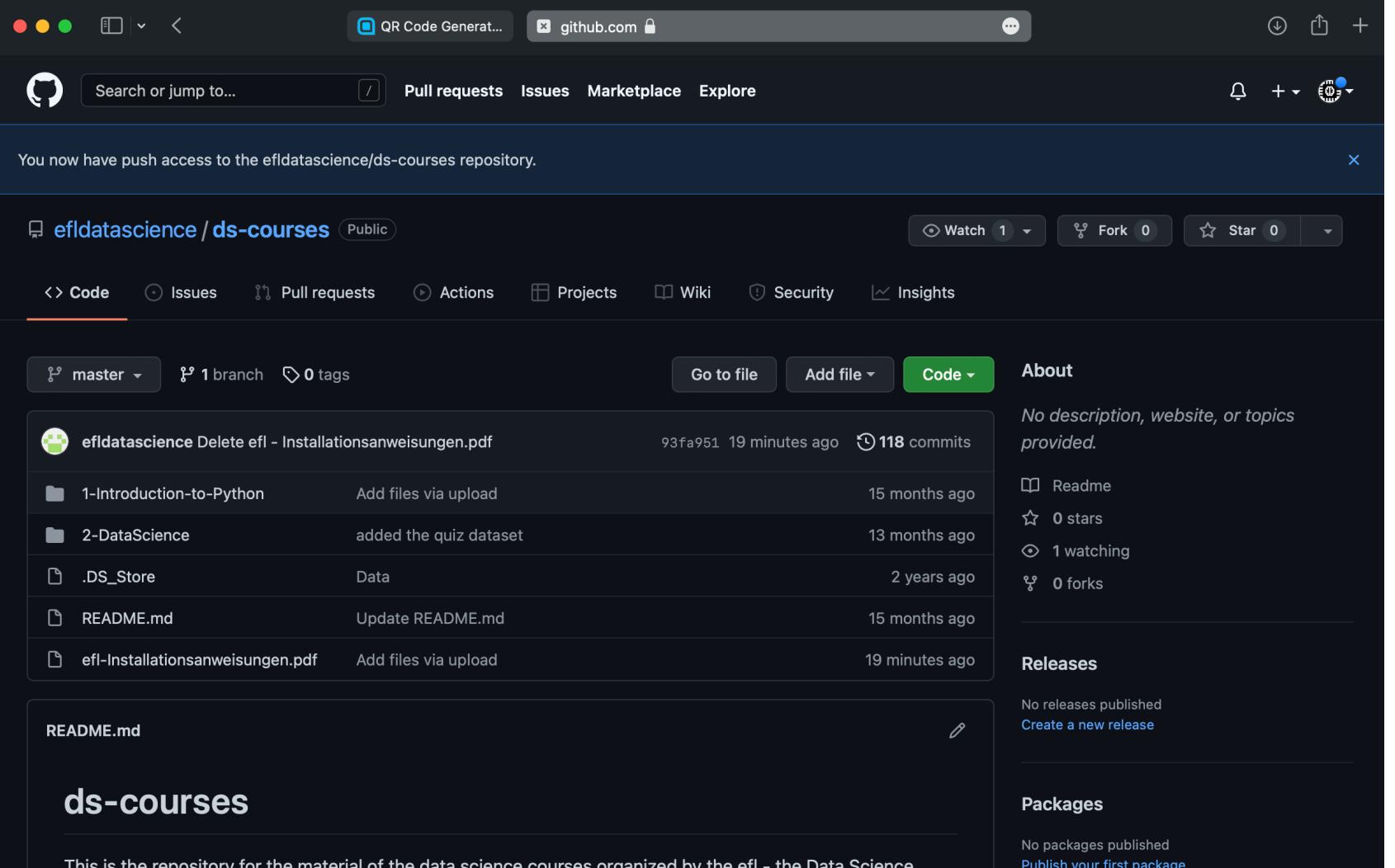


Certificates



- Completion of tasks from the last class
- 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 for 'efldatascience / ds-courses'. The repository is public and has push access granted. It contains one branch ('master') and no tags. The last commit was made 19 minutes ago by 'efldatascience' with the message 'Delete efl - Installationsanweisungen.pdf'. Other commits include adding files via upload for '1-Introduction-to-Python', '2-DataScience', and 'ds-courses', and updating the README.md. The repository has 0 stars, 0 forks, and 1 person watching it. There are no releases or packages published.

You now have push access to the efldatascience/ds-courses repository.

efldatascience / ds-courses Public

Code Issues Pull requests Actions Projects Wiki Security Insights

master 1 branch 0 tags Go to file Add file Code About

No description, website, or topics provided.

Readme 0 stars 1 watching 0 forks

Releases No releases published Create a new release

Packages No packages published Publish your first package

ds-courses

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

<https://github.com/efldatascience/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 €

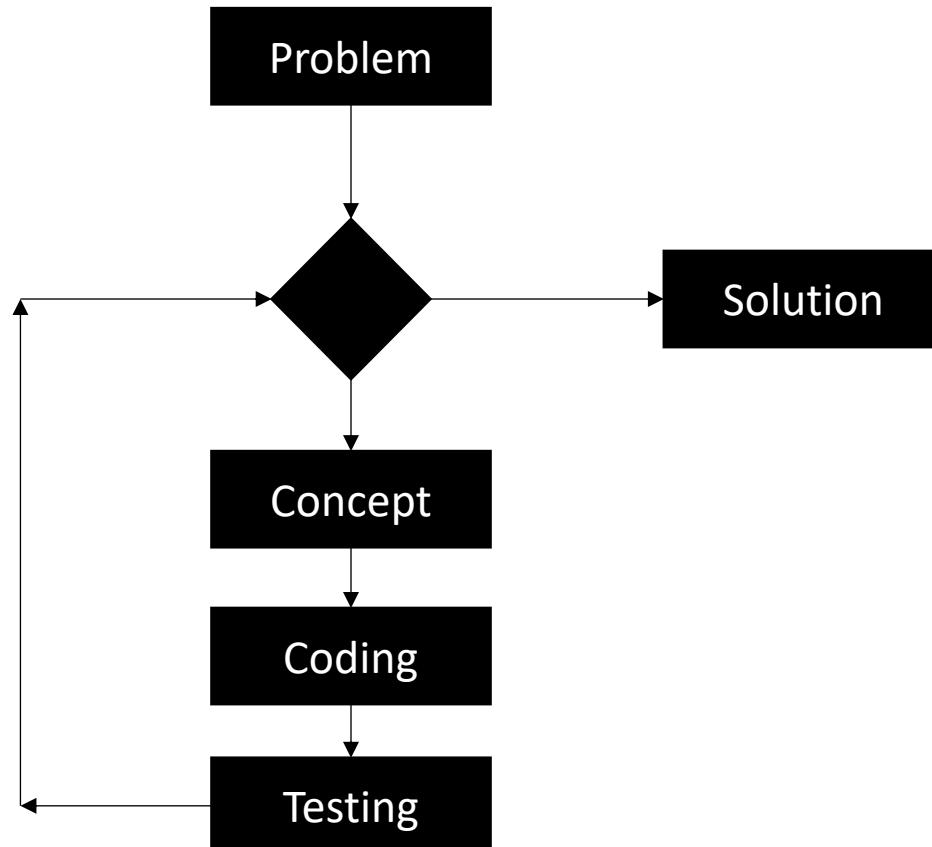
Unternehmensberatung,
Wirtschaftsprüfung und Recht
82.336 €

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

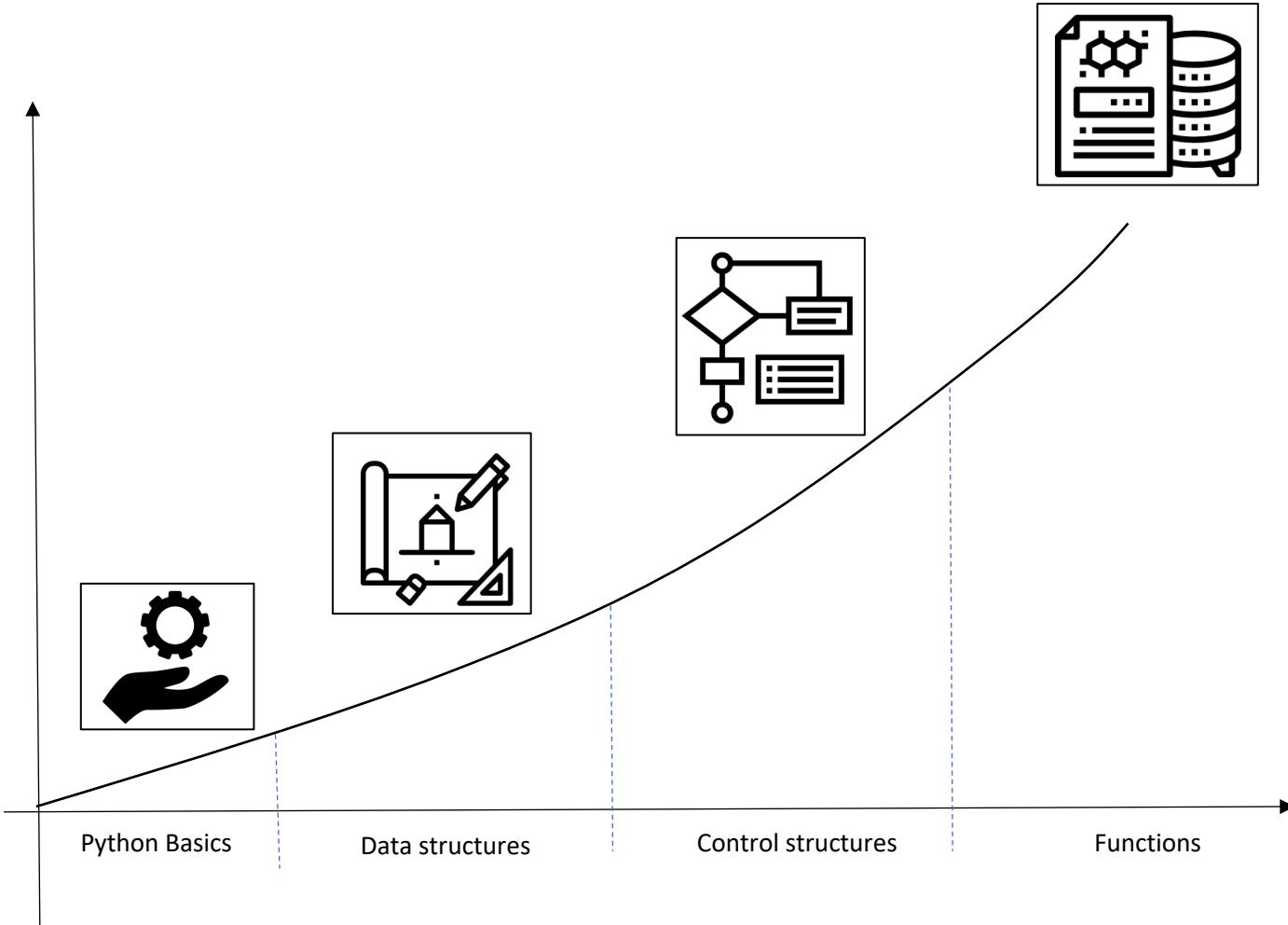
- Philosophy and difference to conventional programming languages
 - Higher programming language
 - It's simple
 - Fast to read
 - Structuring by indenting
 - No {} or ; => Faster to Code
 - Data types are managed dynamically. There is no static type check like in java
 - Widespread in science
 - Extensive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

Introduction Goals

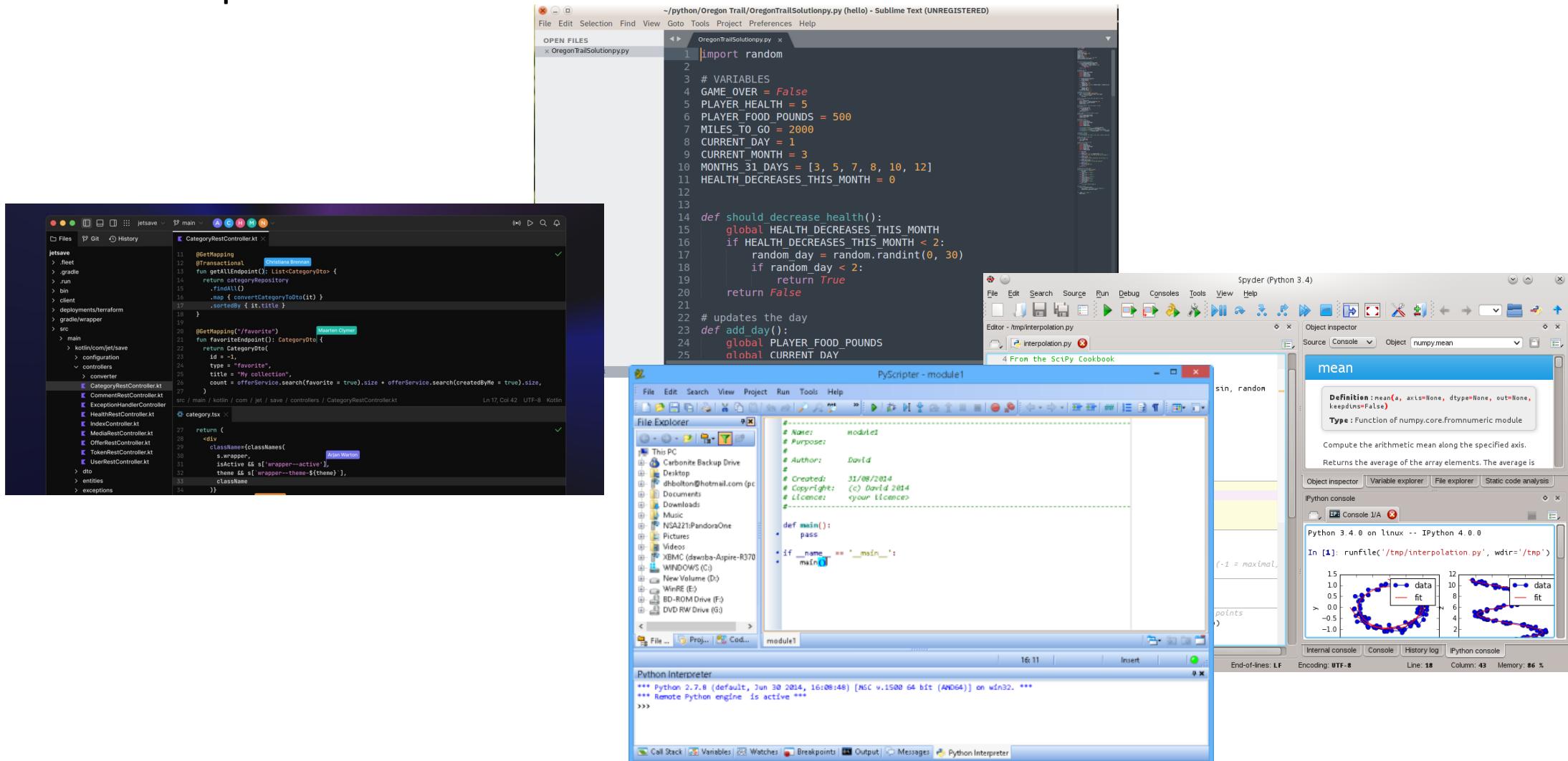


- Introduction to programming
- For beginners
- The module is interactive! Use your computer
- **We develop the solutions together!**
- Please be on time
- There are no dumb questions
- Nobody knows everything
- Copying solutions is plagiarism

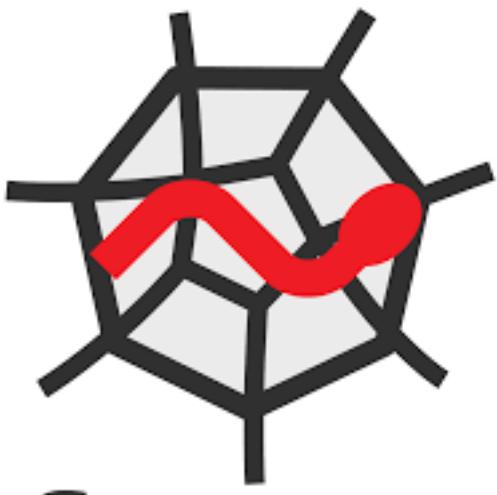
Learning Curve for Today



Integrated Development Environment IDE a Deeper look? The Problem of Choice



What We Use for This Course



SPYDER

The screenshot shows the Spyder Python IDE interface. On the left, there is a code editor with several files open, including `temp.py`, `base.py`, and `test_mainwindow.py`. The code in `temp.py` includes imports from `sklearn.preprocessing`, `numpy`, and `pandas`, and defines a class `testClass` with various methods and array operations. On the right, there is a help documentation panel for the `array` module, showing its definition, parameters, and examples. Below the documentation, the Spyder console shows the output of running `temp.py`, including error messages and stack traces for `SPYDER_DEV` and `SPYDER_PYTEST`.

```
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:/Users/C. A. M. Gerlach/Documents/dev/spyderDev/Spyder 4.0.0/devel (Python 3.7) [DEBUG MODE]
C:/Users/C. A. M. Gerlach/Documents/dev/spyderDev/temp.py
temp.py  base.py  test_mainwindow.py - SpyderDev.../Tests  mainwindow.py
temp.py  base.py  test_mainwindow.py - SpyderDev.../Tests  mainwindow.py

1 # -*- coding: utf-8 -*-
2
3 # Copyright © Spyder Project Contributors
4 # Licensed under the terms of the MIT License
5 # (see spyder/_init_.py for details)
6
7 """
8 Spyder base configuration management
9
10 This file only deals with non-GUI configuration features
11 (in other words, we won't import any PyQt object here, avoiding
12 SIP API incompatibility issues in spyder's non-gui modules)
13 """
14
15 from __future__ import print_function
16
17 import codecs
18 import getpass
19 import locale
20 import os
21 import os.path as osp
22 import platform
23 import shutil
24 import sys
25 import tempfile
26 import warnings
27
28 # Local imports
29 from spyder import __version__
30 from spyder.utils import encoding
31 from spyder.py3compat import (is_unicode, TEXT_TYPES, INT_TYPES,
32                               to_text_string, is_text_string)
33
34 """
35 # Only for development
36
37 # To activate developer settings for development
38 # SPYDER_DEV is (and *only* has to be) set in bootstrap-py
39 DEV = os.environ.get('SPYDER_DEV')
40
41 # Manually override whether the dev configuration directory is
42 USE_DEV_CONFIG_DIR = os.environ.get('SPYDER_USE_DEV_CONFIG_DIR')
43
44 # Make Spyder use a temp clean configuration directory for tests
45 # SPYDER_SAFE_MODE can be set using the --safe-mode option of b2
46 SAFE_MODE = os.environ.get('SPYDER_SAFE_MODE')
47
48 def running_under_pytest():
49     """
50         Return True if currently running under py.test.
51
52         This function is used to do some adjustments for testing. The
53         variable SPYDER_PYTEST is defined in configst.py.
54
55         return bool(os.environ.get('SPYDER_PYTEST'))
56     """
57
58     def is_stable_version(version):
59         """
60             Return True if version is stable, i.e. with letters in the j
61
62             Stable version examples: '1.2', '1.3.4', '1.0.5'.
63             Non-stable version examples: '1.3.4beta', '0.1.0rc1', ...
64
65         """
66
67         Parameters
68         -----
69         df : TYPE
70             DESCRIPTION.
```

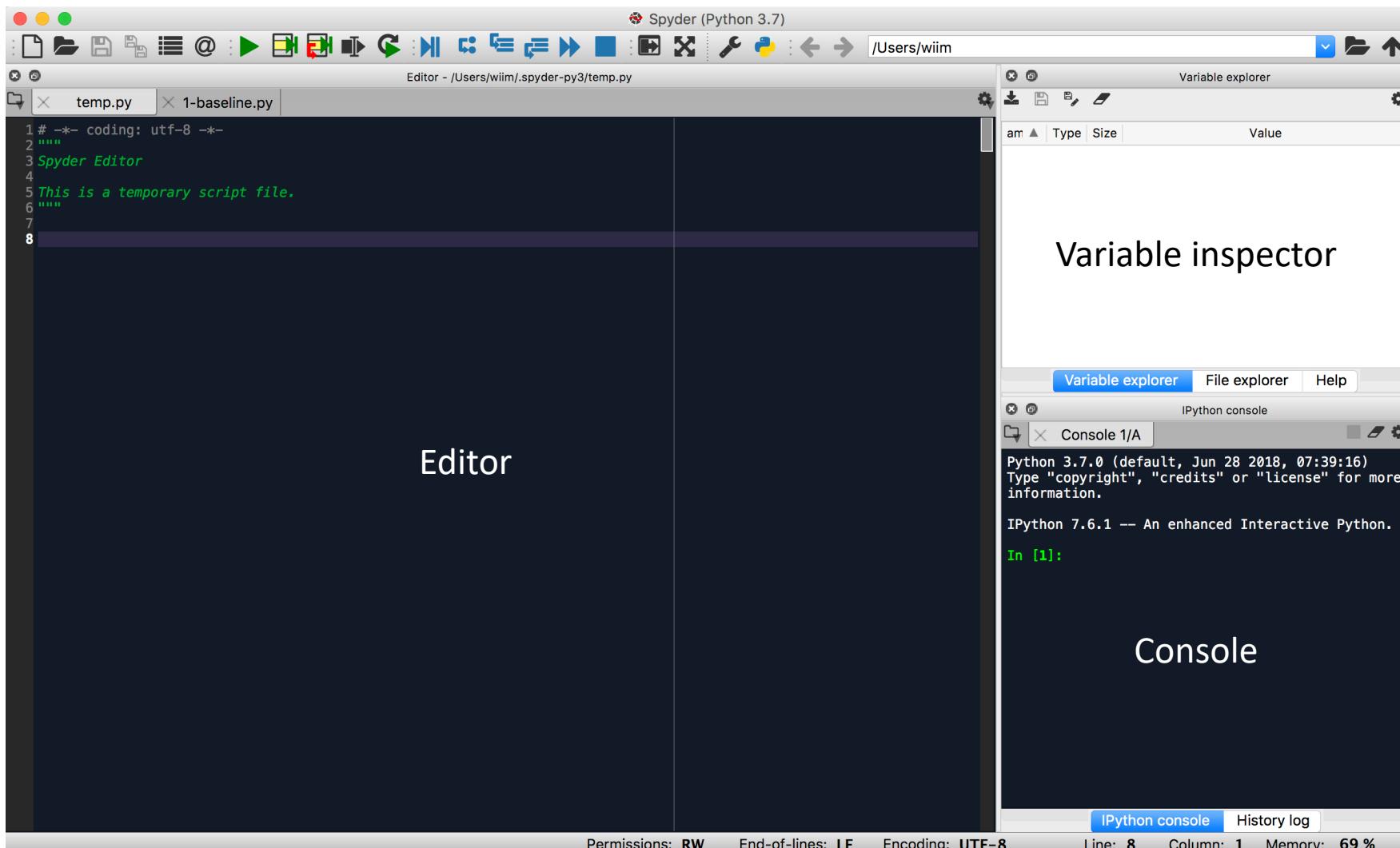
Practice and Questions

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

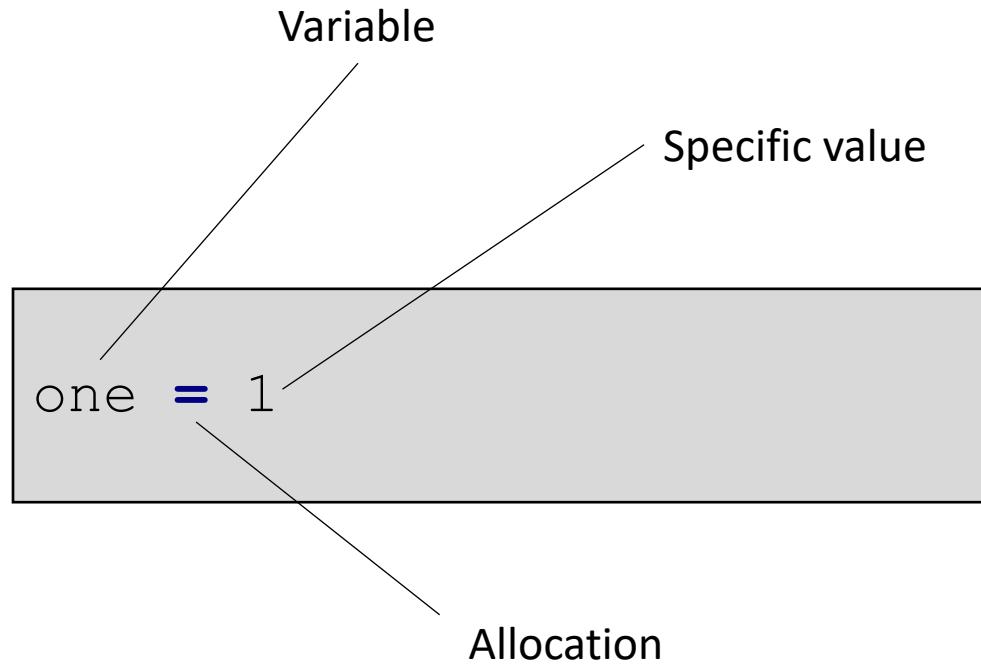
- Hello World
- Let's try our first example!
- Display "Hello World" on the console

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.

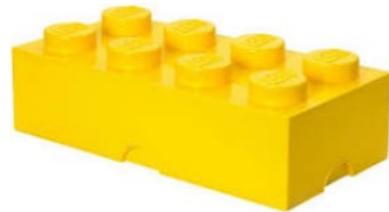
– John V. Guttag



Different Building Blocks



Integer



Float



Strings



Boolean

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 (in Python == Strings)

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!

Unicode code point	character	UTF-8 (dec.)	name
U+0000		0	<<control>
U+0001	1	1	<<control>
U+0002	2	2	<<control>
U+0003	3	3	<<control>
U+0004	4	4	<<control>
U+0005	5	5	<<control>
U+0006	6	6	<<control>
U+0007	7	7	<<control>
U+0008	8	8	<<control>
U+0009	9	9	<<control>
U+000A	10	10	<<control>
U+000B	11	11	<<control>
U+000C	12	12	<<control>
U+000D	13	13	<<control>
U+000E	14	14	<<control>
U+000F	15	15	<<control>
U+0010	16	16	<<control>
U+0011	17	17	<<control>
U+0012	18	18	<<control>
U+0013	19	19	<<control>
U+0014	20	20	<<control>
U+0015	21	21	<<control>
U+0016	22	22	<<control>
U+0017	23	23	<<control>
U+0018	24	24	<<control>
U+0019	25	25	<<control>
U+001A	26	26	<<control>
U+001B	27	27	<<control>
U+001C	28	28	<<control>
U+001D	29	29	<<control>
U+001E	30	30	<<control>
U+001F	31	31	<<control>
U+0020	32	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	0	48	DIGIT ZERO
U+0031	1	49	DIGIT ONE
U+0032	2	50	DIGIT TWO
U+0033	3	51	DIGIT THREE
U+0034	4	52	DIGIT FOUR
U+0035	5	53	DIGIT FIVE
U+0036	6	54	DIGIT SIX
U+0037	7	55	DIGIT SEVEN
U+0038	8	56	DIGIT EIGHT
U+0039	9	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	λ
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
W	W	W	W
U	W	U	U
F	W	U	F

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

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

- 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

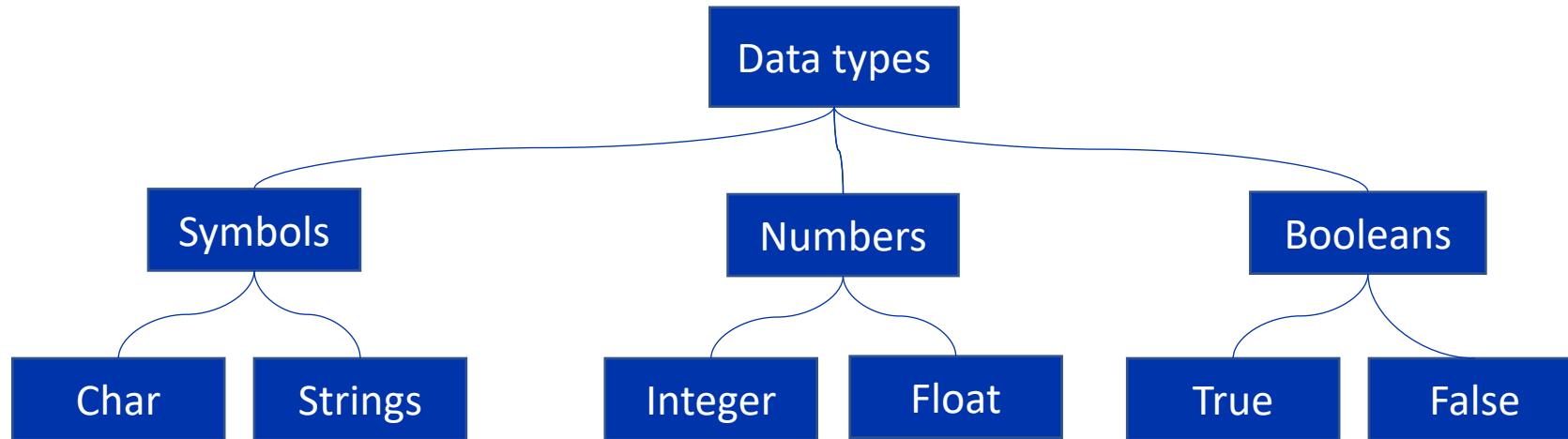
Logical operations

```
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: 2021-12-20
@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...



These are all approaches. Find your own style!