

Informatik II Woche 1



Einführung Python, JupyterLab

Website: n.ethz.ch/~kvaratharaja

Password for Polybox: kvaratharaja

Welcome!

- Kissan Varatharajan
- 4. Semester Maschinenbau
- Software Engineer @ ARIS Nautilus Team
- Interessen: Robotics, AI, Controls

Übersicht

- **1. Teil: Einführung Python (Woche 1-4)**

Python Containers, List/Dict Comprehension, numpy, matplotlib, pandas, Python Classes

- **2. Teil: Algorithmen & Datenstrukturen (Woche 5-9)**

Asymptotik, Runtime, Suchalgorithmen, Trees, Dynamic programming

- **3. Teil: Machine Learning (Woche 11-13)**

decision trees, regression, overfitting, cross-validation, unsupervised learning

Heute






- Intro Python
- JupyterLab / JupyterHub
- Tipps für die Übungsaufgaben

Python

- Einer der populärsten Programmiersprachen
 - Einfach zu lernen
 - Viele nützliche Libraries
- Machine learning!

10,000+ projects with the selected classifier Order by Relevance

TOPIC :: SCIENTIFIC/ENGINEERING x

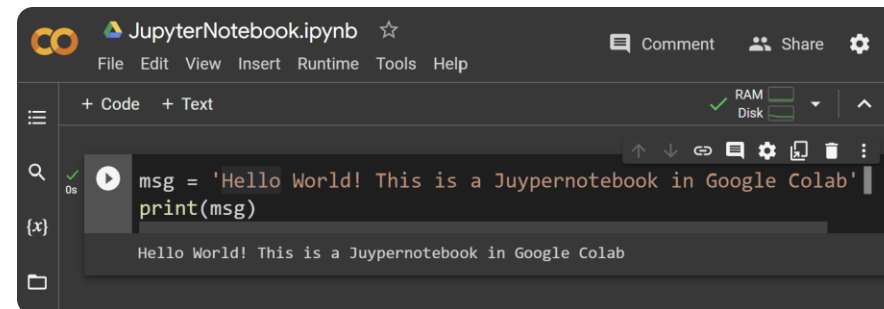
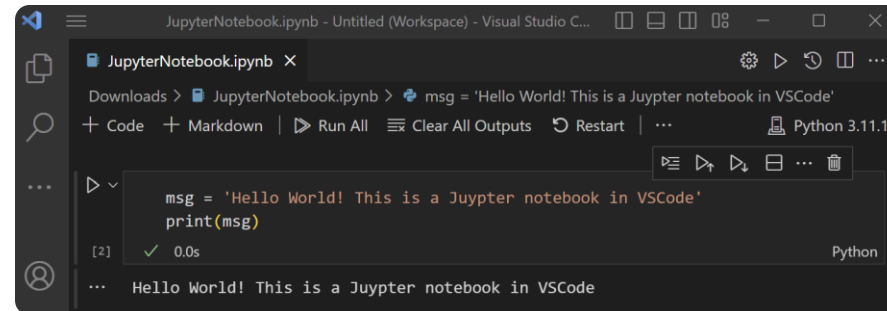
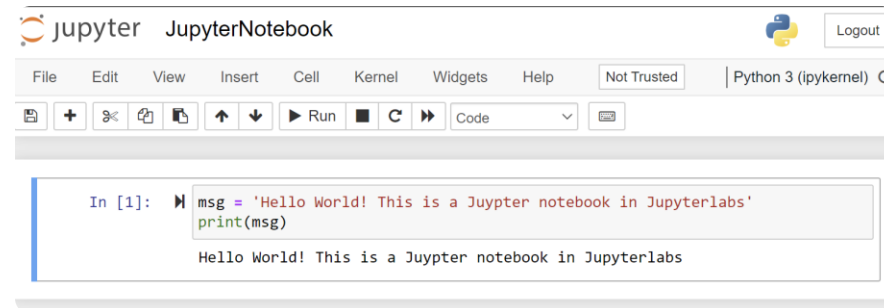
	flytekitplugins-notebook 0.1.0 Your microlib descrition	Jan 26, 2021
	moarchiving 0.6.0 Biobjective Archive class with hypervolume indicator and uncrowded hypervolume improvement computation	Jun 12, 2021
	itk-ransac 0.1.4 An ITK-based implementation of RANSAC used for point cloud registration.	Jan 20, 2023
	itk-skullstripping 0.1.0 Automatic skull-stripping for neuroimage analysis	May 23, 2021
	milc 1.6.6 Opinionated Batteries-Included Python 3 CLI Framework.	Mar 27, 2022

Python vs C++

- Wenig bis kaum Memorymanagement (Dynamic Memory allocation & Garbage Collector)
- Python: Interpreted (&Compiled)
- Dynamische Typisierung (Python: Laufzeit, C++: Kompilierzeit)
- Integration von externem Code einfach
- ...
- Langsamer ☹️

Python Code

- JupyterLab
- JupyterHub
- VS Code with the Jupyter Notebook extension
- Google Colab



Jupyter Notebooks

- Zeigt den Output von mehreren Versionen des Codes
- Einfache Gliederung des Codes
- Unterstützt Kommentare wie Texte und Bilder, um den Code zu komplementieren
- Die Vorlesungsnotizen werden auch in einem Jupyter Notebook bereitgestellt, so könnt ihr es live testen!

Nützliche Shortcuts

- Shift+Enter: Führt aktuelle Zelle aus und springt zur nächsten
- Ctrl+Enter: Führt aktuelle Zelle aus, bleibt in derselben Zelle
- Alt+Enter: Führt aktuelle Zelle aus, fügt darunter neue Zelle ein
- A: Fügt neue Zelle oberhalb der aktuellen Zelle ein
- B: Fügt neue Zelle unterhalb der aktuellen Zelle ein

JupyterHub

- Keine Installation notwendig
- Kann im Browser geöffnet werden
- Auch auf Tablet nutzbar
- Alternative zu Google Colab

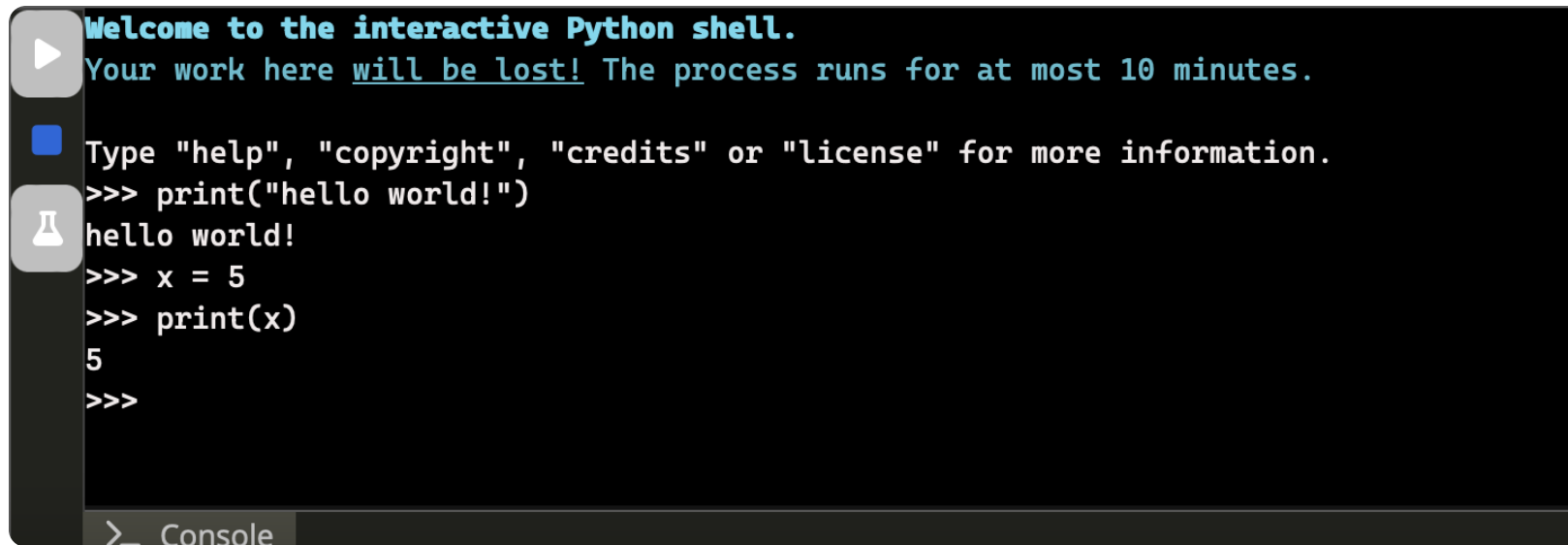


C++ to Python Tutorial

- Schritt für Schritt Einführung in Python
- Vergleich zwischen C++ und Python Syntax
- Link: <https://lecturers.inf.ethz.ch/tutorials/cpp-to-py>
- freiwillig

Neu in Code Expert: Interactive Shell

- Schnelles Testen
- Code geht verloren



The screenshot shows the Code Expert Interactive Python Shell interface. On the left, there is a vertical toolbar with three icons: a play button (top), a blue square (middle), and a flask icon (bottom). The main area displays the following text:

```
Welcome to the interactive Python shell.  
Your work here will be lost! The process runs for at most 10 minutes.  
  
Type "help", "copyright", "credits" or "license" for more information.  
>>> print("hello world!")  
hello world!  
>>> x = 5  
>>> print(x)  
5  
>>>
```

At the bottom, there is a tab labeled "Console" with a prompt character ">".

Tipps für die Übungen der Woche 1

- Ex 1: Python und JupyterLab installieren-> Anleitung hier: [slides](#)
 - **WICHTIG: INDENTING ÄNDERN**

Empfohlene Zusammenfassung 1

- Diese Zusammenfassung stammt von Julian Lotzer und Daniel Steinhauser (ist in meiner Polybox)

ZF CompSci II D-MAVT

Julian Lotzer – lotzer@student.ethz.ch
 Daniel Steinhäuser – steinhauser@student.ethz.ch
 Version 21.02.2023

This summary is based on the D-MAVT Computer Science II lectures by Dr. Ralf Sasse and Dr. Carlos Corinti. The ZF is constantly updated, as the lecture takes place for the first time in this form. No guarantee can be given for correctness or completeness.

1 General Python

1.1 Reference Semantics and Aliasing

Everything is a pointer:

```
11 = [1, 3, 'hi', -4] #11 -> [1] [3] [hi] [-4]
```

```
12 = 11
```

If a copy is needed, use:

```
12 = 11.copy()
```

1.2 Functions

Functions do not have to be declared in a specific order, in contrast to C++ (forward declarations). This means this is completely valid:

```
def foo():
    return bar()

def bar():
    return 0
```

1.2.1 Function Declaration

```
def function(arg1, arg2):
    -
    return value
```

1.2.2 Default arguments

Using default arguments, it is possible to have "optional" arguments:

```
def specialprint(data="hello world")
    print(data)

specialprint() #hello world
```

2 Python Containers

Python containers can be divided into ordered (sequences)

Sequences include tuple (all types), list (all types), range (sequences and str (character types))

Collections are e.g. set (non-associative) and dictionary (associative)

2.1 Operations on Containers

- Number of Elements: `len()`
- Containers element `x`? `x in s`
- Iterate over all elements: `for x in c: print(x)`

2.2 Sequences (ordered containers)

- Tuple with 4 Elements: `t = ('a', 0, -6, 3.3)` #1 -> [2] #3 -> [4] [5] [6] [7]
- Write with 1 Element: `t = ('a',)`
Empty tuple: `t = ()`
- List of 4 Items: `l = [1, 3, 'hi', -4]` #1 -> [1] [3] [hi] [-4]
- Range with 4 elements: `s = range(0, 8, 2)` #1 -> [2] [4] [6] [8]
- String with Length 5: `s = "hello"` #1 -> [h] [e] [l] [l] [o]

IMPORTANT: tuple is mutable, tuple, range and string are immutable!

2.2.1 General sequence operations

- Subscript-Operator `l[i]`: `l = [1, 3, 'hi', -4]`
`print(l[2])` output: hi
- Enumeration: `enumerate(iterable, start)`
Iterable = iterable container (sequence)
Start (optional) = (optional) enumeration starts counting but this number, starts at 0 when omitting start-argument
`enumerate(iterable)`
Returns `enumerate(iterable, start)` function returns a tuple: (index, object)
- Enumeration example: `for index, value in enumerate(l): print(index, value)`

```

Output:
# 0 1
# 1 3
# 2 hi
# 3 -4

```

- Combine sequences `s1` and `s2` (zip): `z = zip(s1, s2)`

```

Example:
#1 -> [1] #2 -> [2] #3 -> [3] #4 -> [4]
#1 -> [1] #2 -> [2] #3 -> [3] #4 -> [4]
#1 -> [1] #2 -> [2] #3 -> [3] #4 -> [4]

```

- Output with a for loop:

2.2.3 Dictionary Operations

```

d = {"name": "John", "age": 30}
# Change item:
d["Peter"] = 24
# Delete item:
del d["John"]
# Search for a key:
"Tim" in d returns a bool
# To access value at a key:
d["Tim"] # returns value 19

```

- Make two lists into one dictionary: `cities = {"Zurich", "Basel", "Bern"} #list 1`
`zip code = [8800, 4000, 3000] #list 2`
`d = Dict(zip(cities, zip code))` dictionary D2
- Iterate over the keys of a dictionary: `for key in d.keys(): print(key)` #1 Tim Morris
- Iterate over entries of a dictionary: `for item in d.items(): print(item)` #1 ("Tim", 19) ("Morris", 69)
- Iterate over entries, with keys and values separated: `for key, value in d.items(): print(key + " = " + value)` #1 Tim 19 Morris 69
- Iterate over the values of the dictionary: `for value in d.values(): print(value)` #1 19 69

2.3.3 Dictionary/Set Comprehension

- Transform a set into a dictionary by applying `f(x)` and `g(x)` on every element in the set to obtain key and value respectively:
`d3 = {f(x):g(x) for x in s}` #s being a set
- Transform a set into a dictionary, only if the element satisfies `h(x)`:
`d4 = {f(x):g(x) for x in s if h(x)}`
- Dictionary comprehension with multiple variables:
`d5 = {f(x1):g(x2) for x1, x2 in h(x)}`
#s must return a list of tuples, e.g. zip, d.items, where case of d.items, we are applying f(x) on the key and g(x) on the values of the dictionary.
- Example: Make the value of every odd key in a dictionary by 2:
`d6 = {k:2*v for k, v in d.items if k % 2 == 1}`

2.2.2 Common List Operations

- Change item: `l[i] = value`
- Add value at the end: `l.append(value)`
- Remove item at location i: `del l[i]`
- Invert list: `l.reverse()`
- Create a list of k elements with value v: `l = [v]*k`
- To convert a string to a list: `s.split(separator, maxsplit)`
Separator and maxsplit are optional
`My.split()` -> split at all whitespaces
`My.split(",")` -> split at every ","
`My.split(maxsplit = 10)` -> split 10 times at the first 10 whitespaces -> list will have 11 entries

2.2.3 List Comprehension

- Apply a function `f(x)` to all items in list l: `l2 = [f(x) for x in l]` #2.g. `2*x for f(x)`
- Apply a function `f(x)` to a range: `r2 = [f(x) for x in range(1,6)]`
- Apply a function `f(x)` only to items in list l that satisfy `h(x)` (filter):
`l3 = [f(x) for x in l if h(x)]`
- Example: Read a sequence of numbers:
`l = [int(x) for x in input("input: ").split()]`

2.3 Collections (unordered containers)

- Set with 3 items: `s = {1, 29, 12}`
- Dictionary with 3 items: `d = {"name": "John", "Tim": 19, "Morris": 69}` #key value

2.3.1 Set Operations

- `s = {1, 29, 12}` #set create
- Add item: `s.add(69)`
- Remove item: `s.remove(29)`
- Search for an item: `12 in s` returns a bool

- Diese Zusammenfassung stammt von Gino Kreis (Head TA) und wurde von mir ergänzt. (Weniger Erklärungen, weniger visuell, mehr Code)

Kissan Varatharajan - n.ethz.ch/~kvaratharaja

Credits

Die Slide(-templates) stammen ursprünglich von Julian Lotzer und Daniel Steinhauser, vielen Dank!

→ Checkt ihre Websites ab für zusätzliches Material in Informatik I, Informatik II und Stochastik & Machine Learning.

- <https://n.ethz.ch/~jlotzer/>
- <https://n.ethz.ch/~dsteinhauser/>