Informatik II Woche 2

Einführung Container, Exceptions, List & Dict Comprehension

Website: n.ethz.ch/~kvaratharaja

Password for Polybox: kvaratharaja

Welcome!

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

Inhalt

- **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

1. Containers
2. Lists
3. Exceptions
4. In-class Exercises
5. Homework

Variablen

Dynamische Typsierung:[1] Variablentypen existieren in Python. Sie werden zur Laufzeit zugewiesen und nicht vom Programmierer vorab definiert.

Python

```
i = 1
```

```
d = 1.0
```

```
c = 'a'
```

```
b = True
```

C++

```
int i = 1;
```

```
double d = 1;
```

```
char = 'a';
```

```
bool b = true;
```

[1] *Dieses Thema wird zu einem späteren Zeitpunkt ausführlich besprochen.*

1. Containers

Container

Sequences (geordnet)

- `tuple`
- `list`
- `range`
- `string`

Collections (ungeordnet)

- `set`
- `dictionary`

Container Operationen

Python

Anzahl der Elemente

```
len(c)
```

Beinhaltet c x?

```
b = x in c
```

Iteration über c

```
for x in c:
    print(x)
```

C++

```
c.size();
```

```
std::find(c.begin(), c.end(), x);
```

```
for (int i = 0;i<c.size();i++)
    std::cout << c[i] << "\n";
```

Quiz

- What is the output of the following commands?

```
len(c)
```

6

```
2 in c
```

False

```
for x in c:
    print(x)
```

1
3.14
7
‘a‘
1

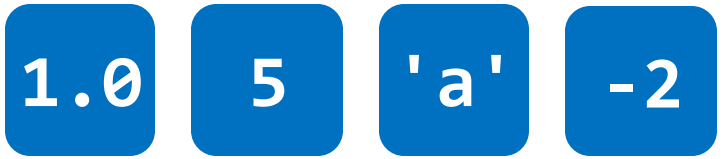
Sequenzen

Tuple *(alle Objekttypen, unveränderlich)*

```
t = (0, 'a', 3.3)
```

list *(alle Objekttypen, veränderlich)*

```
l = [1.0, 5, 'a', -2]
```



range *(Zahlen, unveränderlich)*

```
r = range(1,8,2)
```

string *(Zeichen, unveränderlich)*

```
s = "ETH"
```

Operationen auf Sequenzen

Subscript Operator

```
s[i]
```

Enumeration (Verbinde jedes Element mit seiner Position):

```
for (i,x) in enumerate(s):
   print(i,x)
```

Zip (Verbinde zwei Sequenzen)

```
z = zip(s,t)
l = list(z)
```

Subscript


```
s[3]
```

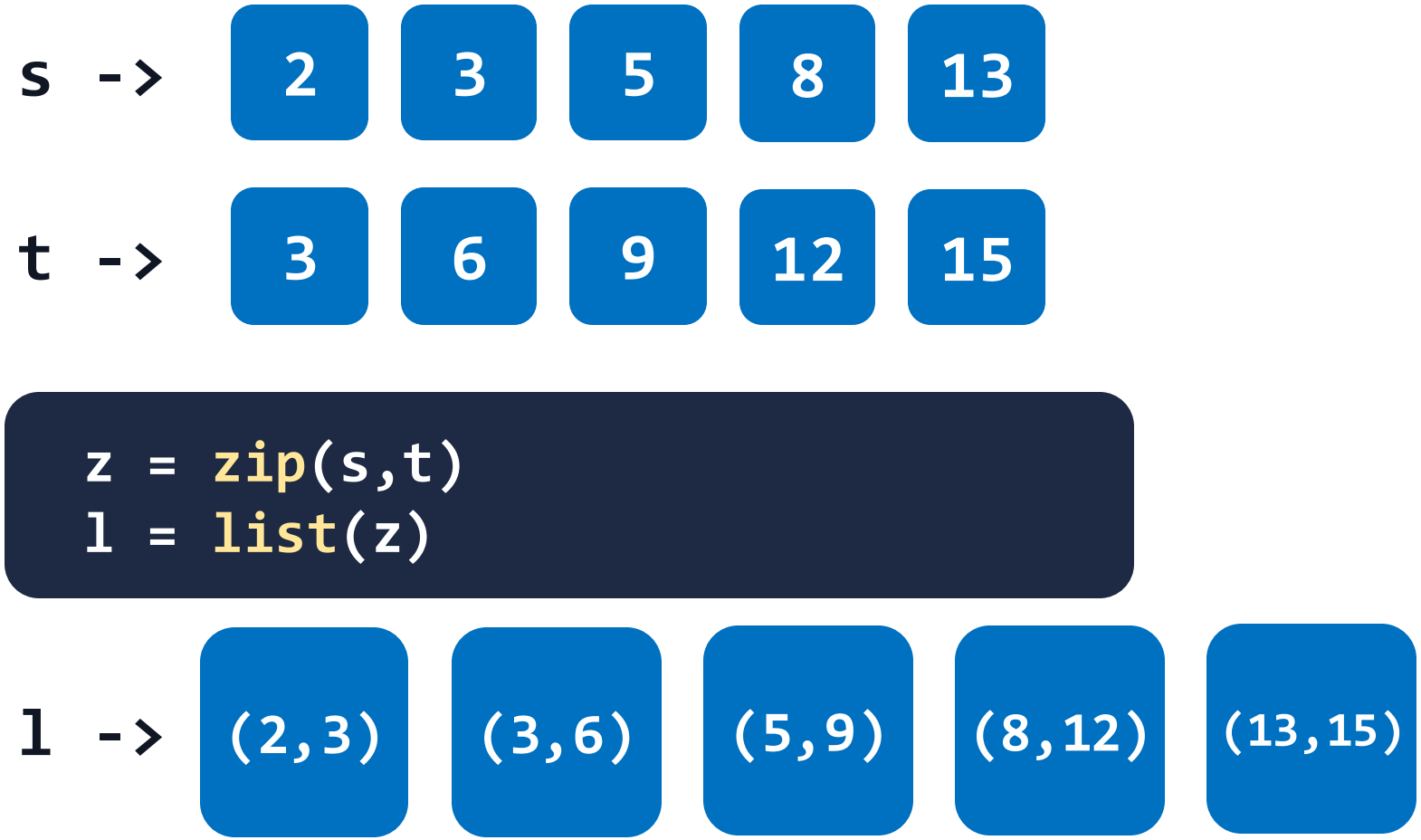
8

Enumeration


```
for (i,x) in enumerate(s):
    print(i,x)
```

```
0 2
1 3
2 5
3 8
4 13
```

Zip



Subscript Quiz

Gegeben ist folgende Liste s

```
s = [0,"Hello World!",('Python', 3.9, 'x'), "z"]
```

Was ist der folgende Output?

```
print(s[2])
```

('Python', 3.9, 'x')

Enumeration Quiz

```
liste1 = ["Apfel", "Banane", "Kirsche"]
```

```
liste2 = ["Rot", "Gelb", "Schwarz"]
```

Was ist der folgende Output?

```
for index, (element1, element2) in enumerate(zip(liste1, liste2)):
    print(f"Index: {index}, Element 1: {element1}, Element 2: {element2}")
```

Enumeration Quiz

Was ist der folgende Output?

```
for index, (element1, element2) in enumerate(zip(liste1, liste2)):
    print(f"Index: {index}, Element 1: {element1}, Element 2: {element2}")
```

Index: 0, Element 1: Apfel, Element 2: Rot
Index: 1, Element 1: Banane, Element 2: Gelb
Index: 2, Element 1: Kirsche, Element 2: Schwarz

Slicing

Auswahl einer Subsequenz gemäss der folgenden Regel:

- Starte bei **start**, Stoppe **bevor stop**, Schrittgrösse **step**

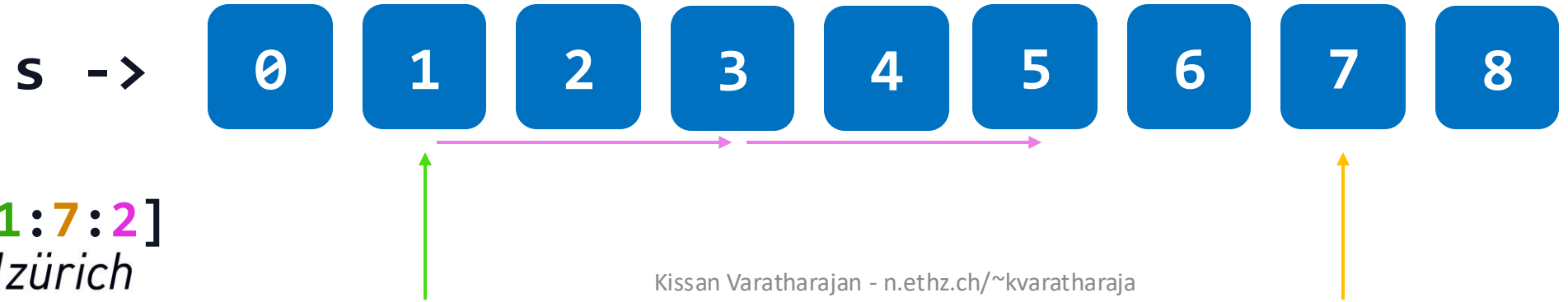
```
s[start:stop:step]
s[start:stop] #step = 1
s[:stop:step] #start = 0
s[start::step] #stop = len(s)
```

Slicing

Auswahl einer Subsequenz gemäss der folgenden Regel:

- Starte bei **start**, stoppe **bevor stop**, Schrittgrösse **step**

```
s[start:stop:step]
s[start:stop] #step = 1
s[:stop:step] #start = 0
s[start::step] #stop = len(s)
```



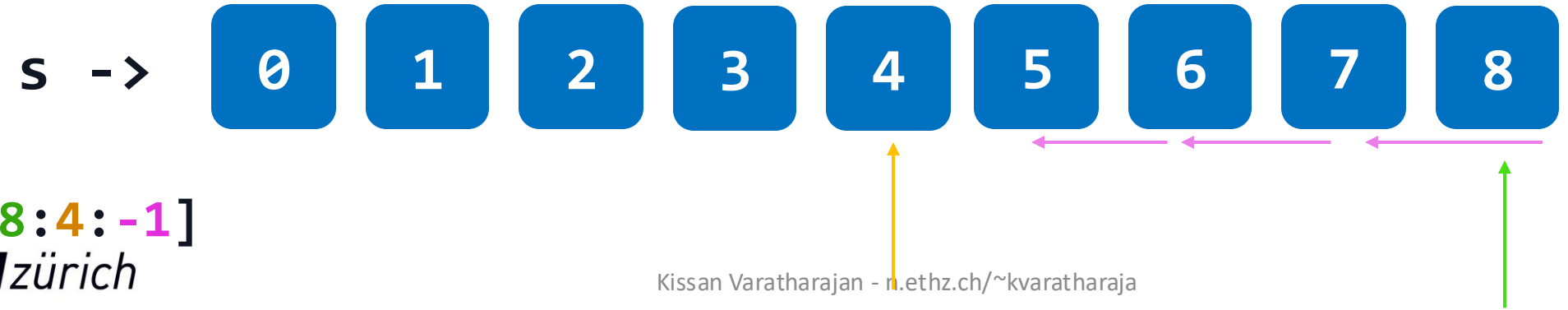
Slicing

Selecting a subsequence according to the following rules:

- Start at **start**, end **before stop**, step size **step**

```
s[start:stop:step]
s[start:stop] #step = 1
s[:stop:step] #start = 0
s[start::step] #stop = len(s)
```

Negative **step**: go backwards.

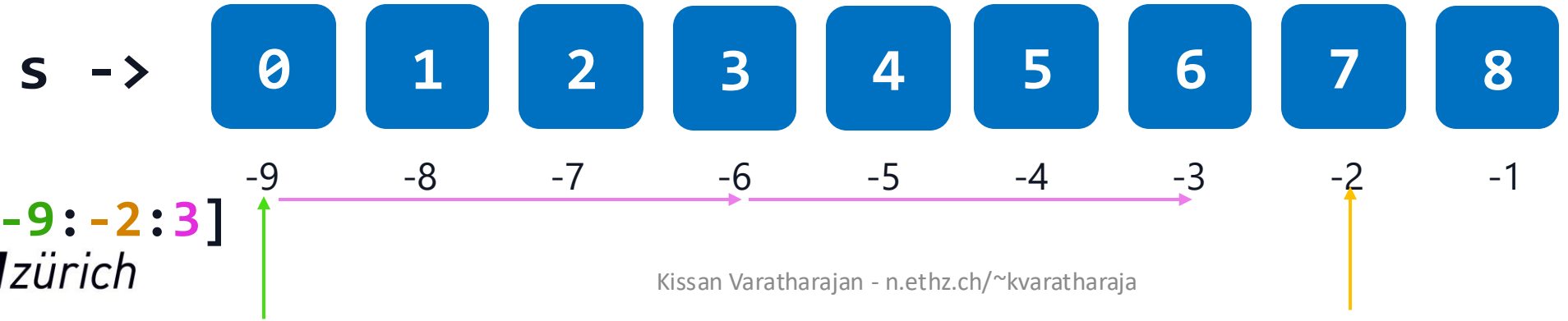


Slicing

Selecting a subsequence according to the following rules:

- Start at **start**, end **before stop**, step size **step**

```
s[start:stop:step]
s[start:stop] #step = 1
s[:stop:step] #start = 0
s[start::step] #stop = len(s)
```



```
s[start:stop:step]
s[start:stop] #step = 1
s[:stop:step] #start = 0
s[start::step] #stop = len(s)
```

Slicing: Quiz 1

Für diese Folie kann folgendes angenommen werden:

```
s = [1, 2, 3, 5, 8, 13, 21, 34, 55]
```

Was ist der Output des folgenden Befehls?

```
s[3::5]
```

[5, 55]

Wie würdest du die Sequenz s aufteilen, um folgenden Output zu produzieren?

```
[34, 8, 2]
```

s[7::-3], s[7:0:-3], s[-2:-9:-3]

Slicing: Quiz 2

```
s[start:stop:step]
s[start:stop] #step = 1
s[:stop:step] #start = 0
s[start::step] #stop = len(s)
```

Es sei folgende Sequenz s gegeben:

```
s = ['A','B','C','D','E','F','G','H','I','J','K','L','M','N','O']
```

Wie würdest du diese Sequenz aufteilen um die folgenden Listen zu erzeugen?

```
['E','F','G','H','I']
['L','K','J','I']
['C','H']
['O','L','I','F']
```

s[4:9]
s[11:7:-1], s[-4:-8:-1]
s[2:8:5],..., s[2:12:5]
s[14:4:-3], s[14:3:-3], s[14:2:-3], s[-1:-11:-3],
s[-1:-12:-3], s[-1:-13:-3]

Range

Eine Sequenz, die bei **start** beginnt, **bevor stop** endet mit der Schrittgrösse **step** :

```
range(start,stop,step)
range(start,stop) #step = 1
range(stop) #start = 0, step = 1
```

Range wird häufig in for-loops verwendet:

Python

```
for i in range(a, b, c):
    do something
```

C++

```
for(int i=a; i<b; i+=c)
    do something;
```

Tupel

Eine generell unveränderliche Sequenz

```
t = () #leeres Tupel
t = (1,) #Tupel mit einem Element
t = (1,2) #Tupel mit zwei Elementen
t = tuple(range(6)) #Tupel mit Range
```

Trick: Swap mit Tupel:

```
x = 6
y = 7
x,y = y,x # (x,y) = (y,x)
print(x,y) # prints 7 6
```

Range: Quiz 1

Was ist der Output des folgenden Befehls?

```
tuple(range(3,15,4))
```

(3,7,11)

Was ist der Output des folgenden Befehls?

```
tuple(range(19,2,-2)[2:7:3])
```

(15,9)

Wie würdest du den folgenden Output mit einem range-Befehl generieren? Fällt dir ein anderer range-Befehl ein, der das gleiche Ergebnis erzielt? Wie viele solcher Möglichkeiten gibt es?

```
(2019, 2023, 2027)
```

range(2019, 2028, 4)

Range: Quiz 2

Wie würdest du den folgenden Output mit einem range-Befehl generieren?

```
[-12, -6, 0, 6, 12]
```

range(-12,13,6),…range(-13,18,6)

Wie würdest du den folgenden Output mit einem range-Befehl generieren?

```
[8, 4, 0, -4]
```

range(8,-5,-4),…range(8,-8,-4)

Wie würdest du **range(15,-15,-3)** aufteilen, um den folgenden Output zu generieren?

```
[-9, -3, 3, 9]
```

range(15,-15,-3)[8:1:-2]
range(15,-15,-3)[8:0:-2]
range(15,-15,-3)[-2:1:-2]
range(15,-15,-3)[-2:-9:-2]

2. Lists

List Operations

Element ändern

```
l[i] = val
```

Element anhängen

```
l.append(val)
```

Element entfernen

```
del l[i]
```

Liste umkehren (inplace)

```
l.reverse()
```

Liste mit Länge k und Werten val

```
l = [val] * k
```

List Operations: Quiz 1

What does list l look like after each step?

```
l = [0] * 4
l[1] = 3
l.append(5)
l.reverse()
del l[3]
```

[0,0,0,0]
[0,3,0,0]
[0,3,0,0,5]
[5,0,0,3,0]
[5,0,0,0]

3. Exceptions

Exceptions

- Exceptions werden ausgelöst, wenn das Programm syntaktisch korrekt ist, der Code jedoch zu einem Fehler geführt hat.
- Einige der am häufigsten vorkommenden Exceptions umfassen **IndexError, ImportError, IOError, ZeroDivisionError, TypeError**, and **FileNotFoundError**
- Beispiel: **ZeroDivisionError**, da wir versuchen durch null zu teilen.

```
a = 1000
b = a / 0
print(b)
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
Cell In[1], line 2
      1 a = 1000
----> 2 b = a / 0
      3 print(b)

ZeroDivisionError: division by zero
```

Behandlung von Exceptions

- Wir können **Try-** und **Except**-Klauseln verwenden, um Exceptions zu behandeln.
- Eine **Try** Anweisung kann verschiedene **Except**-Klausel haben, um verschiedene Fehler zu beheben.
- Trotzdem wird maximal ein **Except** ausgeführt

```
a = [1, 2, 3]
try:
    print("Second element = %d" %(a[1]))
    #error, since array only has 3 elements
    print("Fourth element = %d" %(a[3]))
except IndexError:
    print("An error occurred!")
```

Output:

```
Second element = 2
An error ocurred!
```

Behandlung von Exceptions

Um alle Art von Exceptions zu händeln benutze: **except Exception**

```
a = [1, "Hello World", 3]
try:
   print(a[1])
   #error, since array only has 3 elements
   print("Fourth element = %d" %(a[3]))
except Exception:
   print("An error occurred!")
```

Hello World
An error occured!

Finally

Finally definiert Code, der immer nach einem Try-and-Except-Block ausgeführt wird, unabhängig davon, ob eine Exception ausgelöst wird oder nicht.

```
try:
    k = 5//0 #causes a divide by zero exception
    print(k)
except ZeroDivisionError:
    #this block handles zerodivision exception
    print("Can't divide by zero")
finally:
    #this block is always executed, regardless
    #of exception generation
    print("This is always executed")
```

Output:

```
Can't divide
by zero
This is always
executed
```

Fragen?

Sonst bitte Email an: kvaratharaja@ethz.ch

4. Inclass-Exercise

Lesen von User-Input

- Dieser Befehl schreibt den Text **"Enter a word :"** in die Konsole und wartet auf User-Input.

```
word = input("Enter a word : ")
```

- Nachdem der User Text eingegeben hat, wird dieser in der Variable **word** als String gespeichert.

Lesen von User-Input in einer Schleife

- Dieser Code liest sequentiell String des Benutzers und verarbeitet sie.

```
word = input("Enter a word : ")
again = True
while again:
    #do something with word…
    word = input("Enter a word : (or <Enter> to stop) ")
    again = len(word)>0
```

- Wenn der Nutzer einen leeren String eingibt, wird das Programm beendet.

Palindrome Checker

Ein Palindrome ist ein Wort, welches rückwärts gleich geschrieben wird wie vorwärts

Tipp 1: Ein String ist eine Sequenz, d.h. alle Sequenzoperationen können verwendet werden

Tipp 2: mit slicing kann man einen string leicht umkehren

Werte über dem Durschnitt zählen

Optionale Aufgabe, falls ihr genug Zeit habt:

Schreibe ein Python-Programm mit dem folgenden In- und Output:
Input: Eine Liste s, die Zahlen enthält.
Output: Die Anzahl der Zahlen in der Liste s, die streng grösser sind als der Durchschnittswert aller Elemente.

Example: s = [1,1,2,3,4,1]
Der Durchschnittswert in der Liste s ist gleich 2. Es gibt zwei Zahlen in s, die grösser sind als 2: 3 und 4. Daher sollte der Output der Funktion 2 sein.

Tipp: Ints in eine Liste einlesen: lst = [int(x) for x in input().split()]

5. Homework

Übung 1: Python 1

On: https://expert.ethz.ch/enrolled/SS25/mavt2/exercises

- Crops & Dictionaries
- Sum and Maximum
- Bergprofil

Due Date: Montag 3.3.2025, 20:00 CET

NO HARDCODING

Feedback?

Zu schnell? Zu langsam? Weniger Theorie, mehr Aufgaben?
Dankbar für Feedback am besten mir direkt sagen oder Mail schreiben

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/