FH JOANNEUM
University of Applied Sciences

Introduction to Python

Karl Kreiner

Lectures and exercises (Kreiner)

- Today – RESTful APIs for backends
- **Exercise 1 – Getting started with Python**
- Exercise 2 – Introduction to Django
- Exercise 3 – Database modelling with Django
- Exercise 4 – Creating an RESTful API and securing your application

Goals for today

- Get started with Python – write a simple script and learn the fundamentals of Python
  - Installing Python
  - Interacting with the interpreter
  - Write simple functions using basic data types and control structures
  - Simple error handling
- How this exercise is organized:
  - Part 1 – Introduction to Python (45”)
  - Coding session 1 (45”)
  - Break (15”)
  - Coding session 2 (45”)

What is Python?

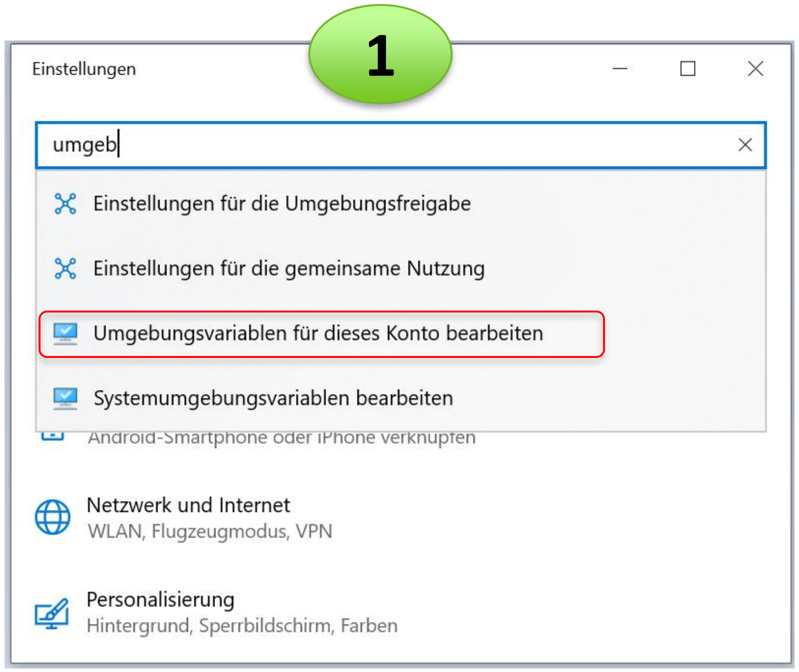
1. Python is an interpreted, high-level, general-purpose programming language.
2. Procedural, functional and object-oriented language
3. Invented by Guido van Rossum in the early 90s
4. Current version: 3.7 – Warning: Python 3 and python 2 are not compatible

→ We use Python 3.7 throughout this course.

Installing Python

1. Python is available on Windows®, Linux-based systems and Apple Macs (and more platforms)
2. Python is pre-installed on (most) Apple Mac and Linux-based systems
3. https://www.python.org/downloads/ (Python can be installed without administrative permissions on the machine)
    1. Windows: comes with Installer doing the heavy lifting
    2. single vs. multi-user mode
4. https://www.wikihow.com/Install-Python-on-Windows - good walkthrough for Windows

Make the interpreter available on Windows consoles



Einstellungen

umgeb

Einstellungen für die Umgebungsfreigabe

Einstellungen für die gemeinsame Nutzung

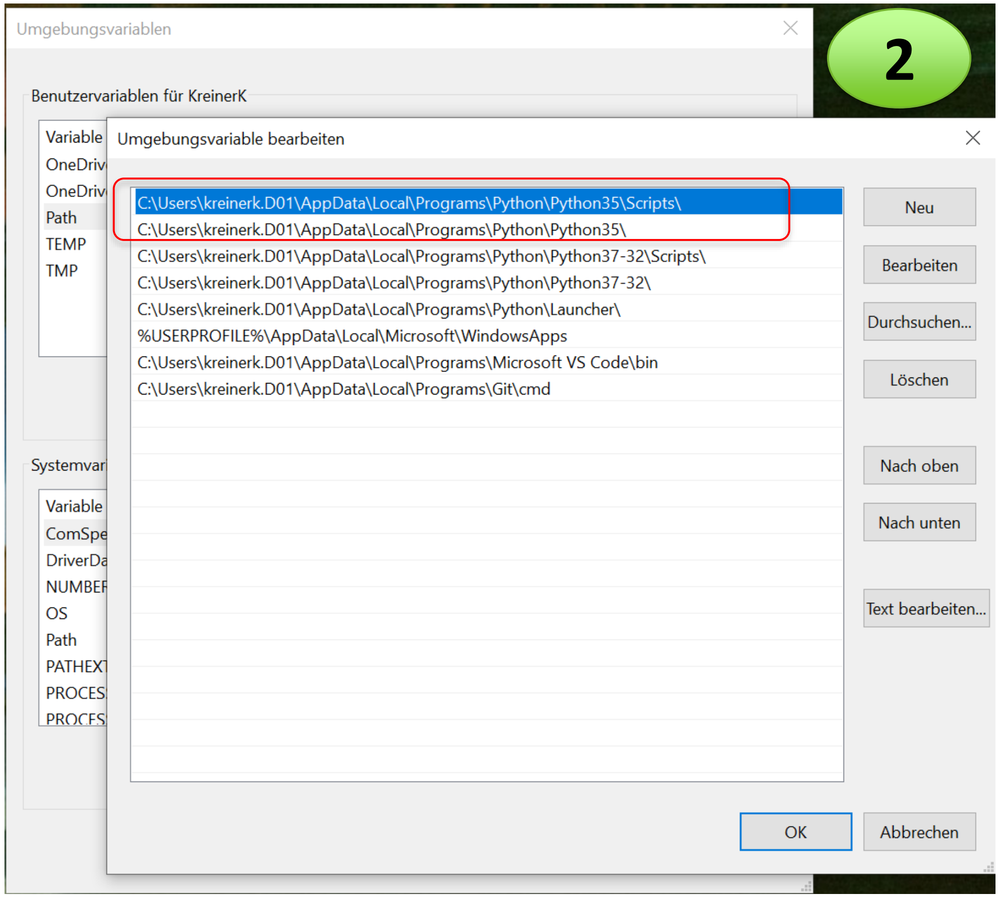
Umgebungsvariablen für dieses Konto bearbeiten

Systemumgebungsvariablen bearbeiten

Android-Smartphone oder iPhone verknüpfen

Netzwerk und Internet
WLAN, Flugzeugmodus, VPN

Personalisierung
Hintergrund, Sperrbildschirm, Farben



Umgebungsvariablen

Benutzervariablen für KreinerK

Umgebungsvariable bearbeiten

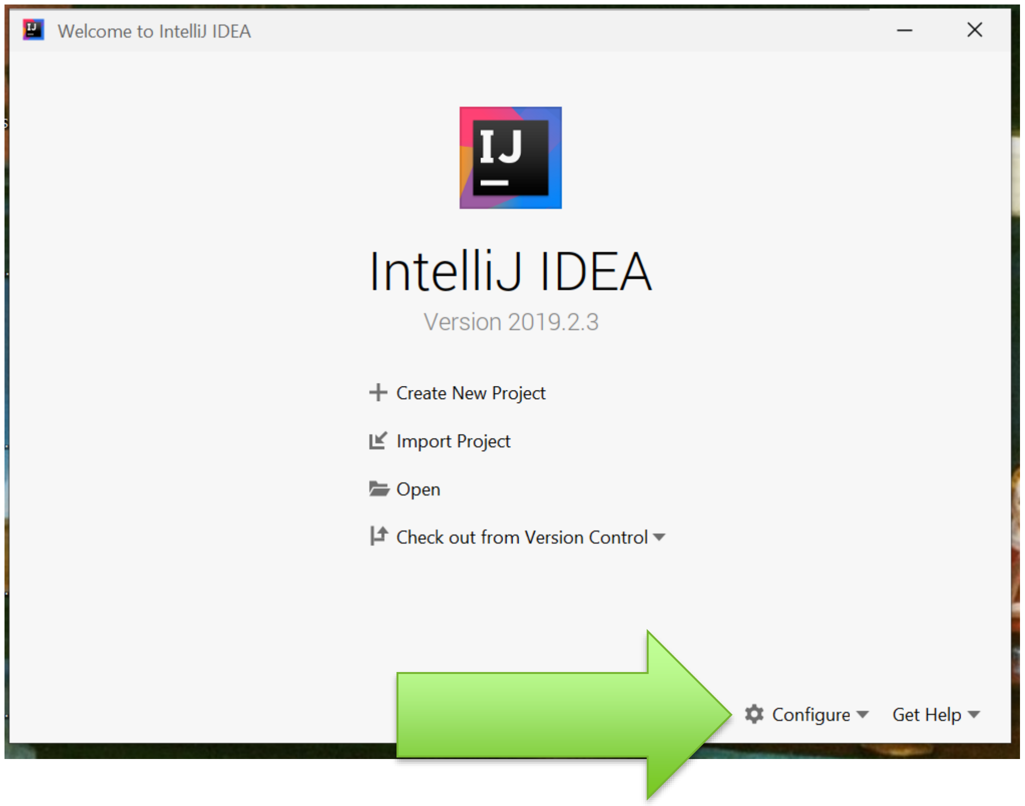
```
C:\Users\kreinerk.D01\AppData\Local\Programs\Python\Python35\Scripts\
C:\Users\kreinerk.D01\AppData\Local\Programs\Python\Python35\
C:\Users\kreinerk.D01\AppData\Local\Programs\Python\Python37-32\Scripts\
C:\Users\kreinerk.D01\AppData\Local\Programs\Python\Python37-32\
C:\Users\kreinerk.D01\AppData\Local\Programs\Python\Launcher\
%USERPROFILE%\AppData\Local\Microsoft\WindowsApps
C:\Users\kreinerk.D01\AppData\Local\Programs\Microsoft VS Code\bin
C:\Users\kreinerk.D01\AppData\Local\Programs\Git\cmd
```

The Python installer does this for you on Windows (be sure to check the flag at the beginning of the installation)

What Integrated Development Environment (IDE)?

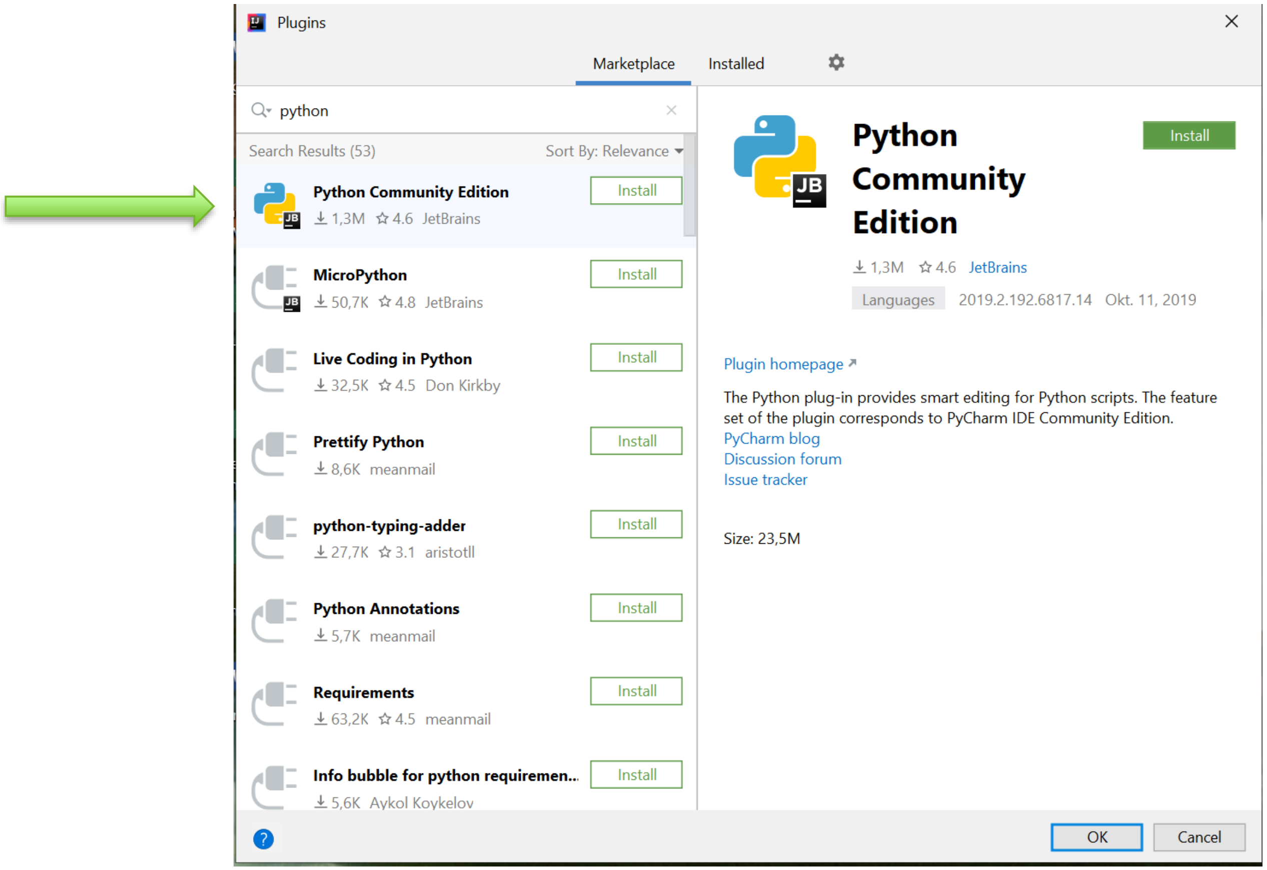
1. For purists: Notepad++, ATOM (https://atom.io/), …
2. IntelliJ IDEA (PyCharm Plugin)
3. For those who are interested: Microsoft Visual Studio Code (https://code.visualstudio.com/)

IntelliJ IDEA – Adding Python support (1/2)



FH JOANNEUM University of Applied Sciences

IntelliJ IDEA – Adding Python support (2/2)



Python Community Edition ⤓ 1,3M ☆ 4.6 JetBrains

MicroPython ⤓ 50,7K ☆ 4.8 JetBrains

Live Coding in Python ⤓ 32,5K ☆ 4.5 Don Kirkby

Prettify Python ⤓ 8,6K meanmail

python-typing-adder ⤓ 27,7K ☆ 3.1 aristotll

Python Annotations ⤓ 5,7K meanmail

Requirements ⤓ 63,2K ☆ 4.5 meanmail

Info bubble for python requiremen... ⤓ 5,6K Aykol Koykelov

Python Community Edition

⤓ 1,3M ☆ 4.6 JetBrains

Languages 2019.2.192.6817.14 Okt. 11, 2019

Plugin homepage ↗

The Python plug-in provides smart editing for Python scripts. The feature set of the plugin corresponds to PyCharm IDE Community Edition.
PyCharm blog
Discussion forum
Issue tracker

Size: 23,5M

Open example code for this exercise

1. Download "exercise1.zip" to your computer (via Moodle) and unpack to folder of your choice
2. In IntelliJ IDEA
   1. File → Open
   2. Select folder where you unpacked exercise1.zip
   3. File → Project structure → Modules
   4. Dependencies → Module SDK should point to Python 3.7

Your IDE should now be „Python-aware", giving you syntax high lightening, pointing out missing modules and giving advice on code style.

FH JOANNEUM University of Applied Sciences

Let’s get started …

Install Python on your machine (or check that your installation is working) and configure IntelliJ to support Python

FH | JOANNEUM
University of Applied Sciences

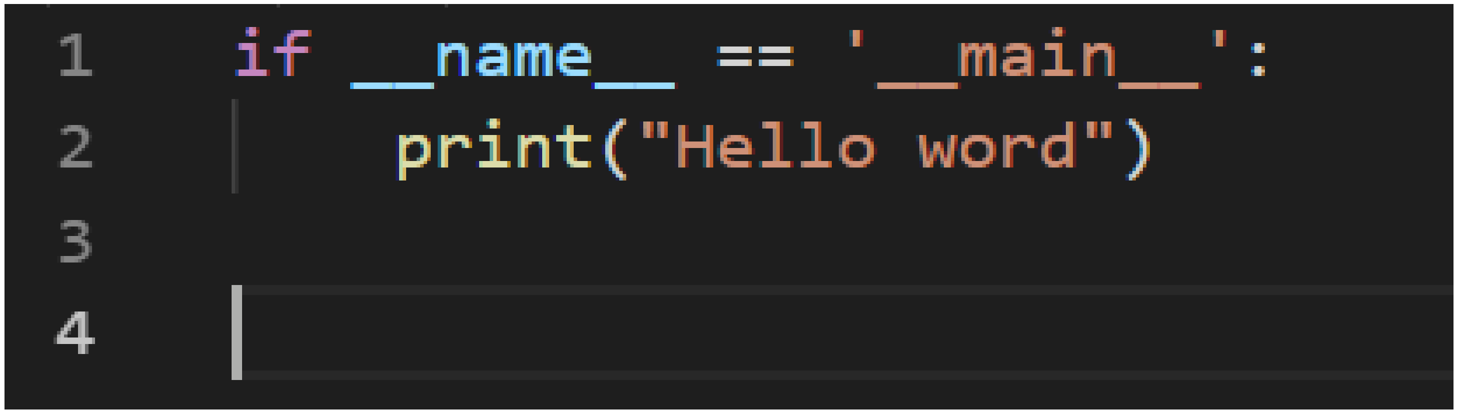
Basics

https://docs.python.org/3/tutorial/introduction.html#using-python-as-a-calculator

Karl Kreiner

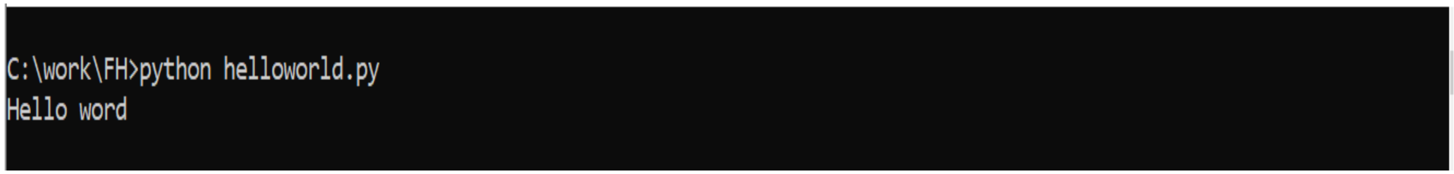
Anatomy of a simple python file

helloword.py



```python
if __name__ == '__main__':
    print("Hello word")

```

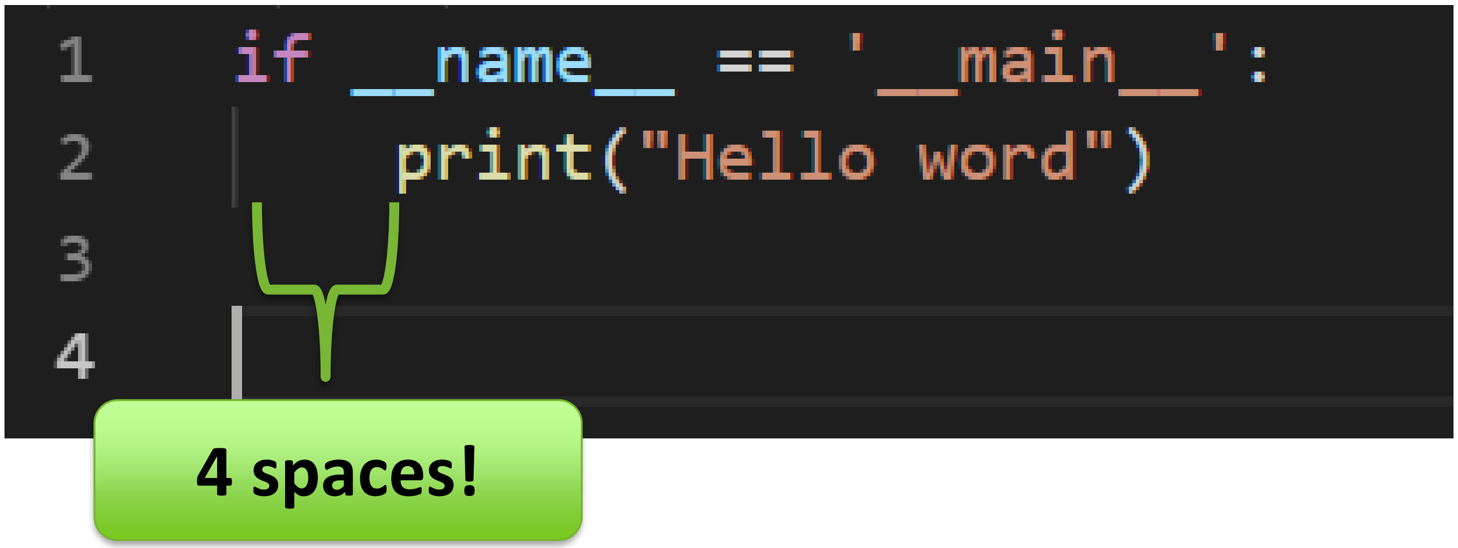


```
C:\work\FH>python helloworld.py
Hello word
```

Python can also be run interactively by just starting the interpreter

Syntax

- Python does not use brackets but **indentation** instead



```python
if __name__ == '__main__':
    print("Hello word")

```

A typical project layout

Package

- main.py
  - *helpers*
    - __init__.py
    - geo.py
    - database.py
    - admin.py
    - tests.py
  - *ui*
    - __init__.py
    - tests.py
    - widgets.py

Module

1. package and module names are always lower-case
2. Each package directory must contain a module called __init__.py (empty) → otherwise it is not recognized by Python as Python package !

Data types

https://docs.python.org/3/tutorial/introduction.html#using-python-as-a-calculator

Karl Kreiner

Basic data types (Selection)

Data type	Usage
bool	`is_active = True # use True or False – case sensitive!`
int	`score = 21`
float	`score = 21.442`
str	`name = "John Doe"`
list	`names = ["John", "Frank", "Mary", "Sue"]`
tuple	`temperature_data = ("Graz", 21)`
dict (Dictionary)	`{"Graz": 21.3 , "Vienna" : 22.4 }`
None	`name = None # counterpart to null in Java`

Python is **duck-typed**. The type of a variable is determined at runtime, when the variable is used the very first time.

See also https://docs.python.org/3/tutorial/introduction.html#using-python-as-a-calculator

Working with strings

No type needs to be declared !

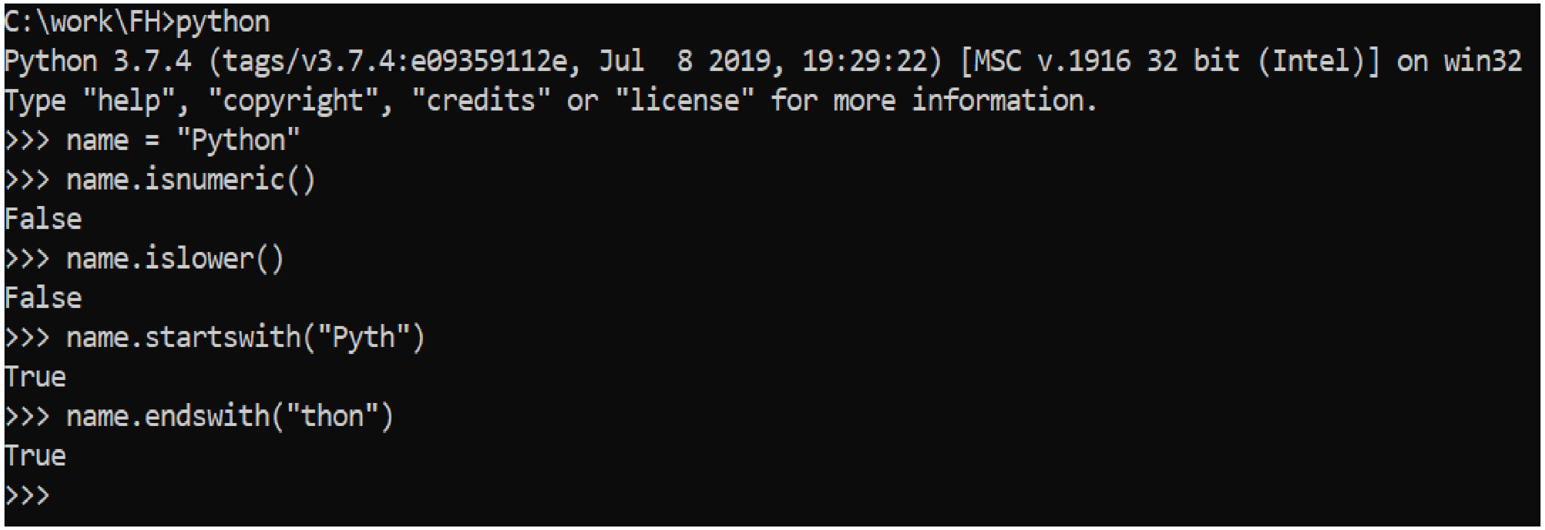
```
>>> word = 'Python'
>>> word[0]   # character in position 0
'P'
>>> word[5]   # character in position 5
'n'
```

```
>>> word[-1]   # last character
'n'
>>> word[-2]   # second-last character
'o'
>>> word[-6]
'P'
```

In Python, this syntax is called „slicing“

```
>>> word[0:2]   # characters from position 0 (included) to 2 (excluded)
'Py'
>>> word[2:5]   # characters from position 2 (included) to 5 (excluded)
'tho'
```

Strings: built-in methods



```
C:\work\FH>python
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 19:29:22) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> name = "Python"
>>> name.isnumeric()
False
>>> name.islower()
False
>>> name.startswith("Pyth")
True
>>> name.endswith("thon")
True
>>>
```

https://docs.python.org/3.7/library/stdtypes.html#string-methods
has a list of built-in string methods

Working with lists

```
>>> squares = [1, 4, 9, 16, 25]
>>> squares
[1, 4, 9, 16, 25]
```

```
>>> squares[0]  # indexing returns the item
1
>>> squares[-1]
25
>>> squares[-3:]  # slicing returns a new list
[9, 16, 25]
```

```
>>> squares + [36, 49, 64, 81, 100]
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

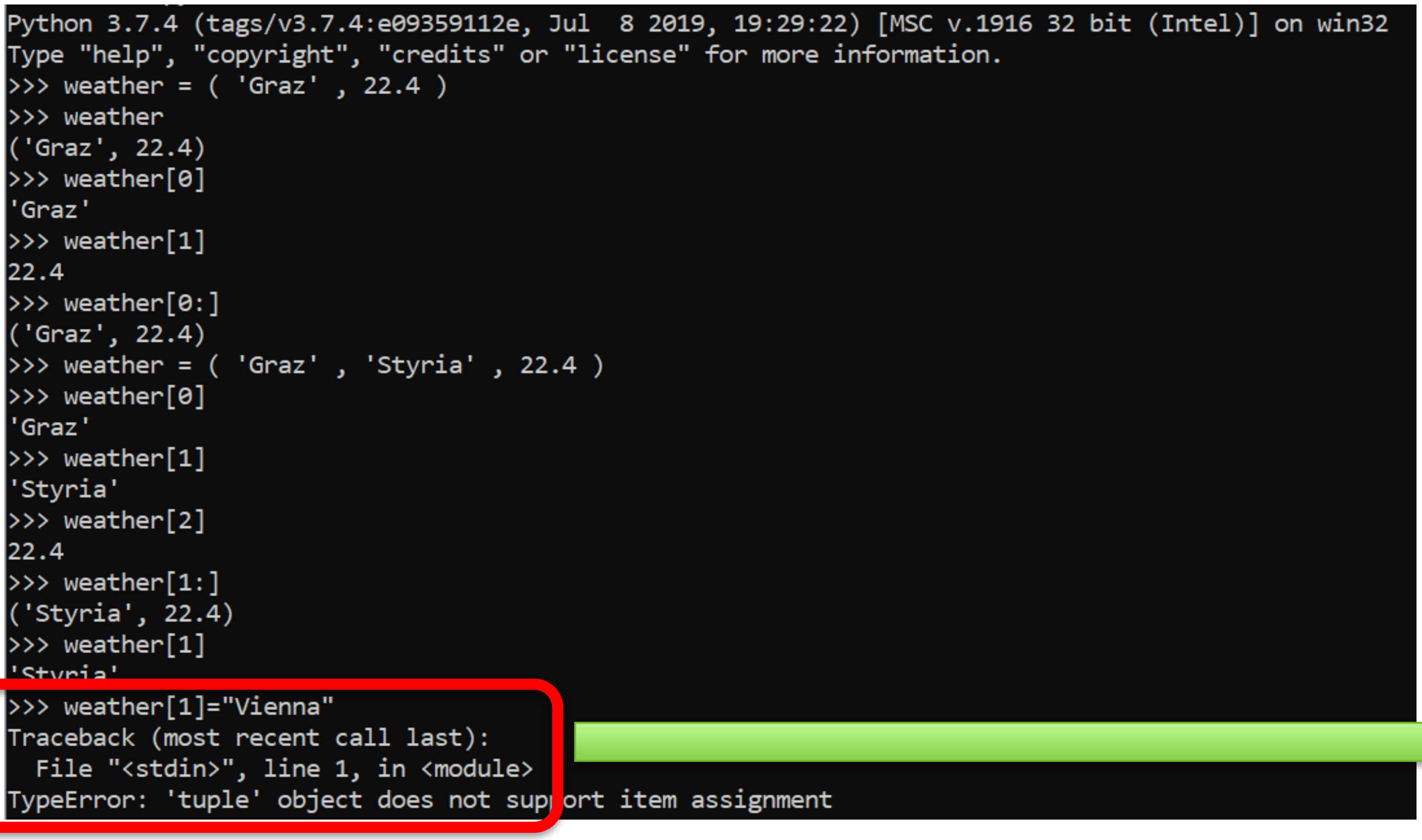
```
>>> cubes = [1, 8, 27, 65, 125]  # something's wrong here
>>> 4 ** 3  # the cube of 4 is 64, not 65!
64
>>> cubes[3] = 64  # replace the wrong value
>>> cubes
[1, 8, 27, 64, 125]
```

Lists are mutable, strings are not !

Lists can have any Python data type, not only integer values (also lists of lists)

Tuples

Tuples are similar to lists, however they are immutable and are often used to represent a mix of different data types:



```
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 19:29:22) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> weather = ( 'Graz' , 22.4 )
>>> weather
('Graz', 22.4)
>>> weather[0]
'Graz'
>>> weather[1]
22.4
>>> weather[0:]
('Graz', 22.4)
>>> weather = ( 'Graz' , 'Styria' , 22.4 )
>>> weather[0]
'Graz'
>>> weather[1]
'Styria'
>>> weather[2]
22.4
>>> weather[1:]
('Styria', 22.4)
>>> weather[1]
'Styria'
>>> weather[1]="Vienna"
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

Immutable!

Working with dictionaries

```
>>> tel = {'jack': 4098, 'sape': 4139}
>>> tel['guido'] = 4127
>>> tel
{'jack': 4098, 'sape': 4139, 'guido': 4127}
>>> tel['jack']
4098
>>> del tel['sape']
>>> tel['irv'] = 4127
>>> tel
{'jack': 4098, 'guido': 4127, 'irv': 4127}
>>> list(tel)
['jack', 'guido', 'irv']
>>> sorted(tel)
['guido', 'irv', 'jack']
>>> 'guido' in tel
True
>>> 'jack' not in tel
False
```

Values in dictionaries can be any Python data type

Control structures

https://docs.python.org/3/tutorial/controlflow.html

Karl Kreiner

If … then … else

```
x = int(input("Please enter an integer: "))

if x < 0:
    x = 0
    print('Negative changed to zero')
elif x == 0:
    print('Zero')
elif x == 1:
    print('Single')
else:
    print('More')
```

Note: you can use the keywords int, float, bool, list, … to cast variables from one type to another

Remember: no brackets (use 4 spaces instead)

For loops

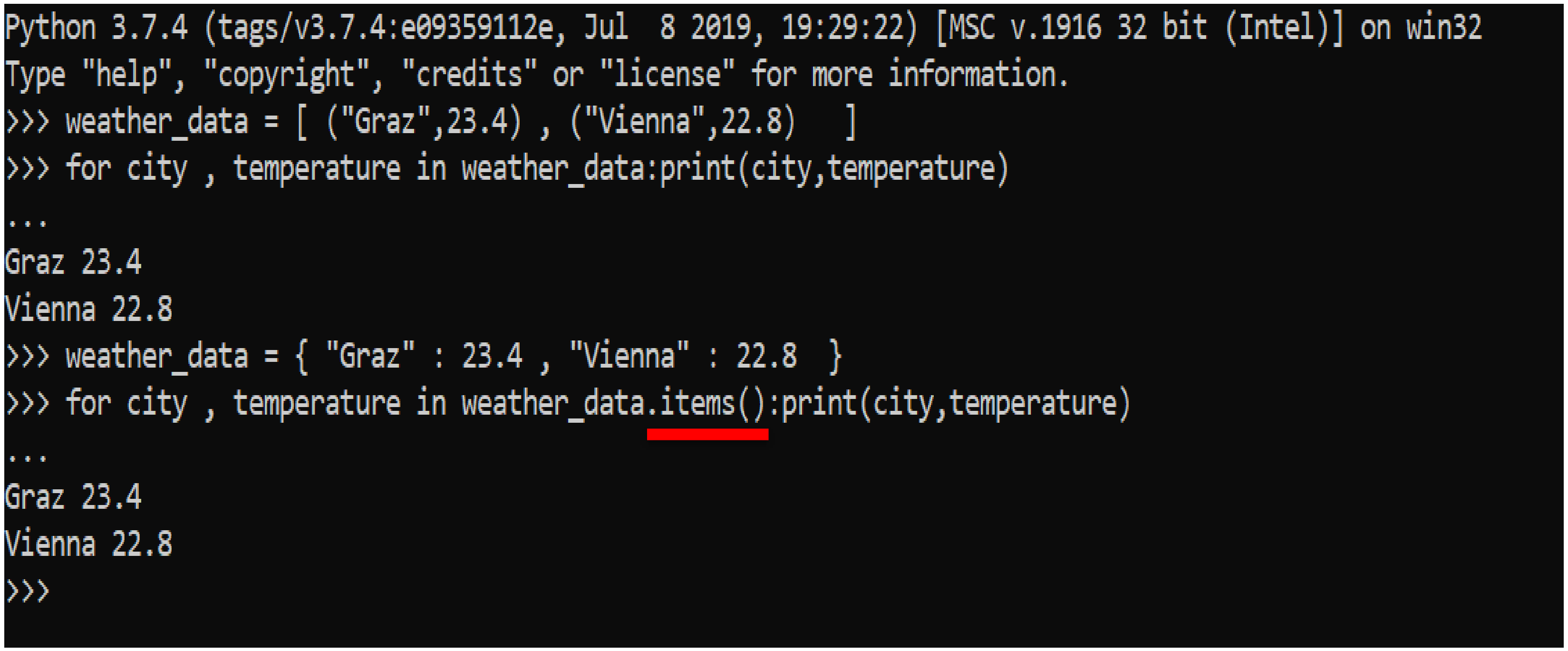
```python
# Measure some strings:
words = ['cat', 'window', 'defenestrate']
for w in words:
    print(w, len(w))
```

or …

```python
a = ['Mary', 'had', 'a', 'little', 'lamb']
for i in range(len(a)):
    print(i, a[i])
```

Remember: no brackets (use 4 spaces instead)

Looping over complex data types



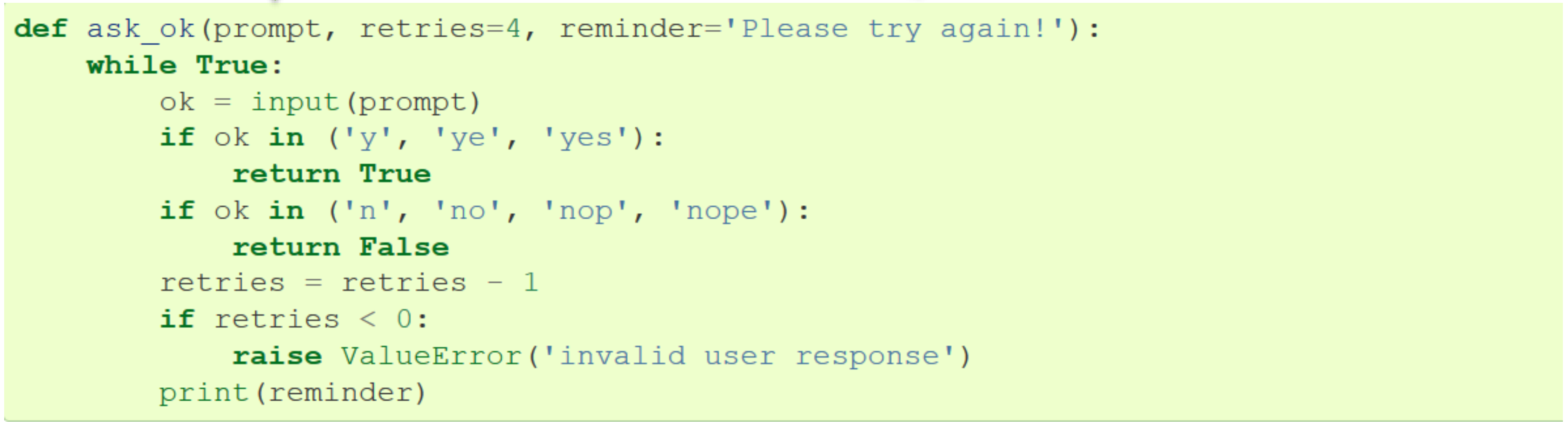
```
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 19:29:22) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> weather_data = [ ("Graz",23.4) , ("Vienna",22.8)   ]
>>> for city , temperature in weather_data:print(city,temperature)
...
Graz 23.4
Vienna 22.8
>>> weather_data = { "Graz" : 23.4 , "Vienna" : 22.8  }
>>> for city , temperature in weather_data.items():print(city,temperature)
...
Graz 23.4
Vienna 22.8
>>>
```

Functions without return values

positional argument

keyword argument

default value

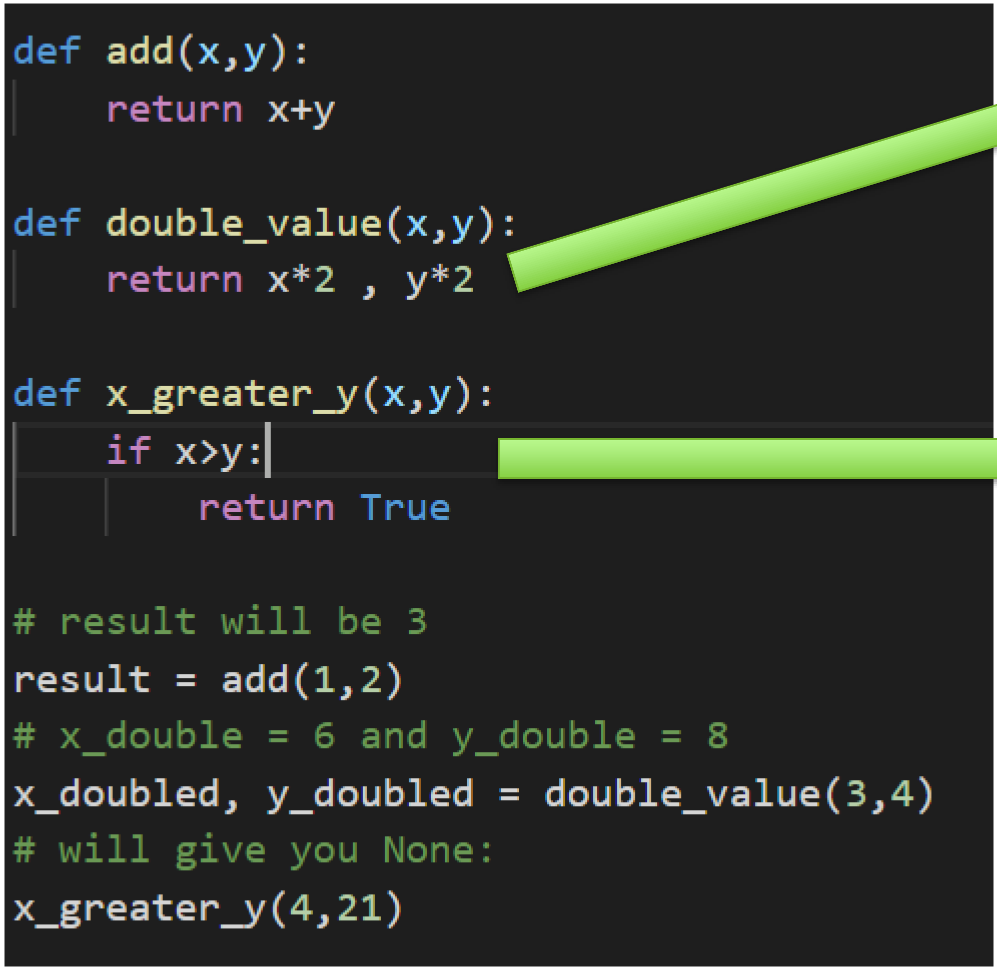


```python
def ask_ok(prompt, retries=4, reminder='Please try again!'):
    while True:
        ok = input(prompt)
        if ok in ('y', 'ye', 'yes'):
            return True
        if ok in ('n', 'no', 'nop', 'nope'):
            return False
        retries = retries - 1
        if retries < 0:
            raise ValueError('invalid user response')
        print(reminder)
```

```
ask_ok("Are you sure you want to delete this file?" )
ask_ok("Are you sure you want to delete this file?", retries=2)
ask_ok("Are you sure you want to delete this file?", reminder="Try aga
in.")
```

The default return value in Python is **None** (even if not explicitly given)

Functions with return values



```
def add(x,y):
    return x+y

def double_value(x,y):
    return x*2 , y*2

def x_greater_y(x,y):
    if x>y:
        return True

# result will be 3
result = add(1,2)
# x_double = 6 and y_double = 8
x_doubled, y_doubled = double_value(3,4)
# will give you None:
x_greater_y(4,21)
```

This functions returns a *tuple* !

Remember:
The default return value in Python is **None** (even if not explicitly given)

Handling errors

```
while True:
    try:
        x = int(input("Please enter a number: "))
        break
    except ValueError:
        print("Oops!  That was no valid number.  Try again...")
```

Python does not know the concept of „declaring exceptions“, a function might raise. If a function might raise an exception, it should be documented, e.g.

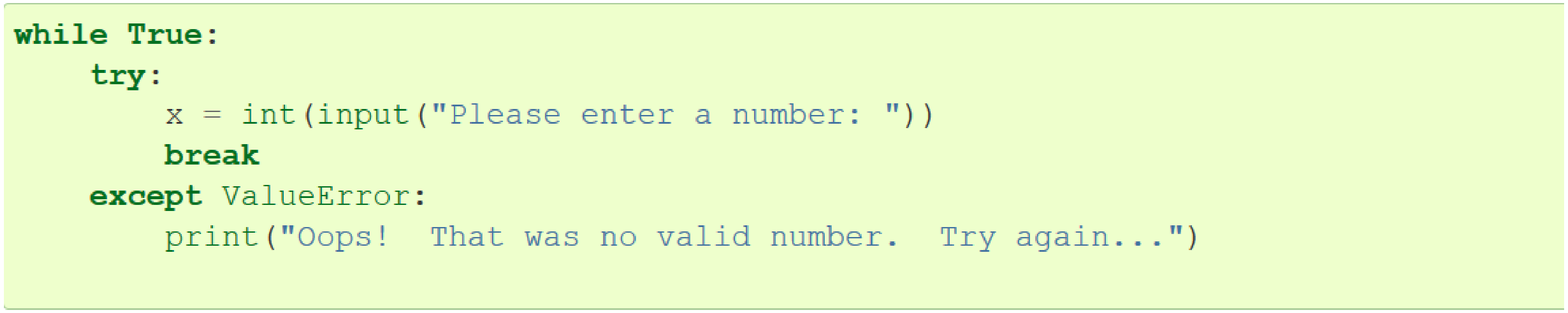
math.**factorial**(*x*)

Return *x* factorial as an integer. Raises `ValueError` if *x* is not integral or is negative.

https://www.tutorialspoint.com/python3/python_exceptions.htm

https://docs.python.org/3/tutorial/errors.html

Raising errors



```
while True:
    try:
        x = int(input("Please enter a number:  "))
        break
    except ValueError:
        print("Oops!  That was no valid number.  Try again...")
```

Try to use **built-in exceptions** where appropriate. However, you can define your exceptions as well:

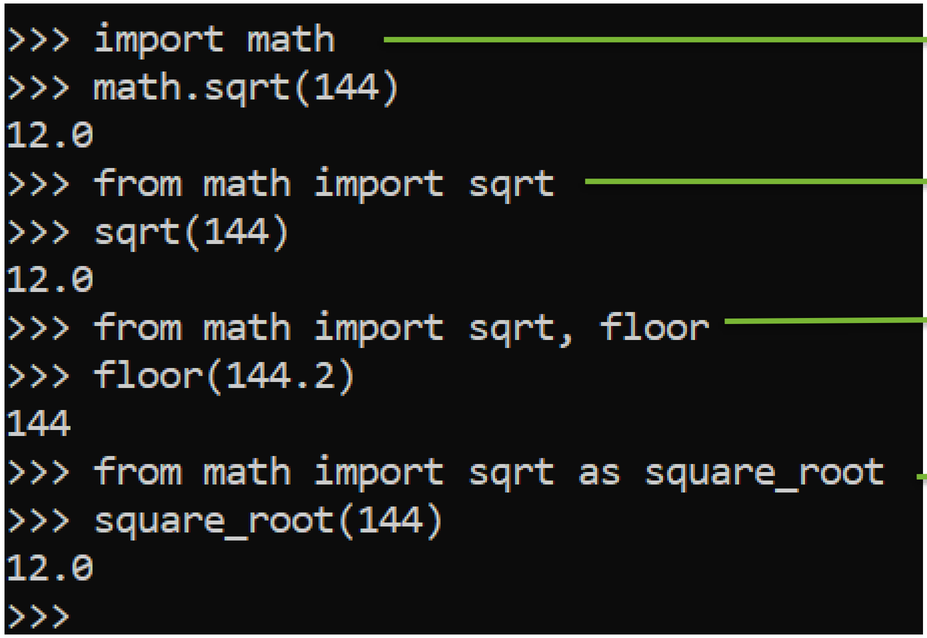
```
class DataNotFoundError(Exception):pass
```

https://www.tutorialspoint.com/python3/python_exceptions.htm

https://docs.python.org/3/tutorial/errors.html

Using the Python standard library

- https://docs.python.org/3/library/index.html
- „Helper libraries“ that are shipped with Python, e.g. helper library for mathematical functions



```
>>> import math
>>> math.sqrt(144)
12.0
>>> from math import sqrt
>>> sqrt(144)
12.0
>>> from math import sqrt, floor
>>> floor(144.2)
144
>>> from math import sqrt as square_root
>>> square_root(144)
12.0
>>>
```

Import whole library

Import function

Import more functions

Import function using a different name (aliasing)

The exercise …

- exercise1.zip contains two files: main.py and exercises.py
- python main.py will make 19 calls to functions defined in exercises.py
- You need to implement the functions in exercises.py (Hint: do it in the order as they are defined)
- python main.py will evaluate if your implementation is correct (with respect to the output) – each correct implementation scores 1 point (out of 19 possible)
- Start in exercise and finish at home as part of homework

Sources (Images)

Code examples on slides 18, 20, 22, 24 , 25, 27, 29, 30 taken from the Python tutorial found at https://docs.python.org/3/tutorial/index.html

Thank you