

# VODAFONE- WORKSHOP



EINFÜHRUNG PYTHON

# HALLO! 🤝

# WER SEID IHR?

- Name?
- Programmiererfahrungen?
- Ziele für heute?

# HEUTIGE AGENDA

- Einführung in Python
- Variablen/Datentypen
- Kontrollstrukturen
  -  Bedingungen
  -  Schleifen
- Funktionen
- Datenstrukturen

# PYTHON-BASICS

VARIABLEN, KONTROLLSTRUKTUREN,  
FUNKTIONEN

# WAS IST PYTHON?

- interpretierte Programmiersprache
- Plattformunabhängig, Open Source
- Anwendungsbereiche insbesondere:
  - Automatisierung
  - Data Science/Machine Learning
  - Physical Computing

# VORTEILE VON PYTHON?

- ✓ Fokus auf schnellen Ergebnissen und Lesbarkeit(!)
- ✓ Vielzahl an Bibliotheken für verschiedene Anwendungsbereiche
- ✓ Große und aktive Community

Laut [TIOBE-Index](#) ist Python aktuell Platz 4 der beliebtesten Programmiersprachen hinter Java, C und C++

# ZUM VERGLEICH..

## PYTHON:

```
print("Hello World!")
```

## JAVA:

```
public class HelloWorld {  
    public static void main(String[] args){  
        System.out.println("Hello World!");  
    }  
}
```



# HELLO WORLD!

- `print()` druckt etwas auf der Standardausgabe aus

```
print("Hello World!")
```

```
print(5+7)  
print(5 < 6)
```

```
print("5^2 ist", 5**2)
```

# EINGABEN LESEN MIT `input()`

- `input()` ermöglicht das Speichern von Benutzereingaben
- Nützlich für einfache Skripte

```
name = input("Wie heißt du?")  
print("Grüß dich, ", name)
```

# VARIABLEN

- Variablen ermöglicht Daten zwischenspeichern
- **Variablenname = Wert**
- Daten besitzen einen *Datentyp*:



Integer (**int**)



Float (**float**)



Boolean (**bool**)



String (**str**)

# INTEGER

- Ganze Zahlen
- Sowohl positiv als auch negativ möglich
- Beispiele:

**128**

**-4**

**182932304**

# FLOAT

- Fließkommazahlen
- Sowohl positiv als auch negativ möglich
- **Achtung: Punkt-Schreibweise**
- Beispiele:

**-4.2**

**0.03000049**


**18.0**

# STRINGS

- Zeichenketten = Liste von Zeichen
- Ein String kann bestehen aus:
  - Zahlen    - Buchstaben    - Sonderzeichen
  - speziellen Zeichen (`\n`, `\t`)
- Beispiele:

```
"Hallo Welt"    "x3af!"    "1234999000"  
""Ich bin ein langer Satz!!1""
```

# BOOLEAN

- Wahrheitswerte: **True** oder **False**
- *Schleifen* und *Bedingungen* arbeiten mit sog. *booleschen Ausdrücken*
-  Boolescher Ausdruck :
  - Ausdruck, der entweder wahr oder falsch ist

[Video zum Thema Aussagenlogik/boolesche Algebra](#)

# ÜBERSICHT: DATENTYPEN

Strings      Zeichenketten

`"Hallo Welt"`

`'Gustav Gans'`

`"""Text`

`mit mehreren  
Zeilen"""`

---

Float      Fließkommazahlen

`5.14`

---

Integer      Ganzzahlen

`5, 99, -23`

---

Boolean      Wahrheitswerte

`True, False`



# RECHENOPERATOREN

**+** **-** **\***  
**/**

Grundrechenarten

5+7, 6\*5,  
-3.0+1.4

**\*\***

Potenz (zB. 5<sup>3</sup>)

5\*\*3

**//**

Ganzzahldivision  
("Wie oft passt X in Y?")

```
7 // 3 # ergibt 1
```

**%**

Modulo-Operator  
("Wieviel Rest ergibt  
Division?")

```
14 % 4 # ergibt 2
```



# LIVE-CODING

A man in a military uniform, wearing a green garrison cap and a dark jacket, points his right index finger directly at the camera with a stern expression. The background is a plain, light-colored wall.

**DRILL TIME!**

# KONTROLLSTRUKTUREN

- Programm läuft normal sequentiell ab
- Kontrollstrukturen: steuern Programmfluss
- 2 Arten von Kontrollstrukturen:

- Bedingungen: **if/elif/else**

"Unter welcher **Bedingung** soll dieser Code laufen?"

- Schleifen: **for/while**

"**Wie oft/wie lange** soll dieser Code laufen?"

# BEDINGUNGEN

- "Wenn X, dann Y"
- **if/elif/else**
- Code, der zu Bedingung gehört, wird eingerückt

# BEDINGUNGEN: BEISPIELE

```
alter = int(input("Wie alt bist du?: "))
if alter >= 18:
    print("Du darfst wählen gehen!")
else:
    print("Du darfst noch nicht wählen gehen!")
```

```
anzahlTage = 24
if anzahlTage <= 7:
    print("Gebühren: 1€")
elif anzahlTage <= 14:
    print("Gebühren: 5€")
elif anzahlTage <= 30:
    print("Gebühren: 10€")
else:
    print("Gebühren: 20€")
```

# VERGLEICHSOPERATOREN

< "größer"

---

<= "kleiner-gleich"

---

== "gleich"

---

>= "größer gleich"

---

> "größer"

---

!= "ungleich"

---

== "gleich"

---

# SCHLEIFEN

- dienen dazu Anweisungen wiederholt auszuführen
- 2 Arten von Schleifen
  - for-Schleife
  - while-Schleife
- Unterschied zwischen den Schleifen rein semantisch



# FOR-SCHLEIFE

Schleife, die in den meisten Fällen eine feste Anzahl Durchläufe hat

Beispiel:

```
for i in range(20):  
    print(i)
```

# WHILE-SCHLEIFE

Schleife, die solange läuft bis eine Abbruchbedingung eintritt

```
i = 0
while i < 20:
    print(i)
    i += 1
```

# DIE `range()`-FUNKTION

- Erzeugt ein Zahlenintervall
- `range(start, stop, step)`
  - `start`: Von der Startzahl..
  - `stop`: ..bis zur Endzahl
  - `step`: ..in folgenden Schritten



# LIVE-CODING

A man in a military uniform, wearing a green campaign hat and a dark jacket, points his right index finger directly at the viewer. He has a serious, stern expression. The background is a plain, light-colored wall.

**DRILL TIME!**

# FUNKTIONEN

- Gruppe von Anweisungen
  - können zusätzliche Daten übergeben bekommen (**Parameter**)
  - Kann Ergebnisse zurückliefern: **return**
- Python bringt Funktionen mit (**built-in-Funktionen**)
- Funktionen können selbst **definiert** werden

# BUILT-IN-FUNKTIONEN

<code>abs()</code>	<code>delattr()</code>	<code>hash()</code>	<code>memoryview()</code>	<code>set()</code>
<code>all()</code>	<code>dict()</code>	<code>help()</code>	<code>min()</code>	<code>setattr()</code>
<code>any()</code>	<code>dir()</code>	<code>hex()</code>	<code>next()</code>	<code>slice()</code>
<code>ascii()</code>	<code>divmod()</code>	<code>id()</code>	<code>object()</code>	<code>sorted()</code>
<code>bin()</code>	<code>enumerate()</code>	<code>input()</code>	<code>oct()</code>	<code>staticmethod()</code>
<code>bool()</code>	<code>eval()</code>	<code>int()</code>	<code>open()</code>	<code>str()</code>
<code>breakpoint()</code>	<code>exec()</code>	<code>isinstance()</code>	<code>ord()</code>	<code>sum()</code>
<code>bytearray()</code>	<code>filter()</code>	<code>issubclass()</code>	<code>pow()</code>	<code>super()</code>
<code>bytes()</code>	<code>float()</code>	<code>iter()</code>	<code>print()</code>	<code>tuple()</code>
<code>callable()</code>	<code>format()</code>	<code>len()</code>	<code>property()</code>	<code>type()</code>
<code>chr()</code>	<code>frozenset()</code>	<code>list()</code>	<code>range()</code>	<code>vars()</code>
<code>classmethod()</code>	<code>getattr()</code>	<code>locals()</code>	<code>repr()</code>	<code>zip()</code>
<code>compile()</code>	<code>globals()</code>	<code>map()</code>	<code>reversed()</code>	<code>__import__()</code>
<code>complex()</code>	<code>hasattr()</code>	<code>max()</code>	<code>round()</code>	

# EIGENE FUNKTION DEFINIEREN

- Mithilfe des Keywords **def** können eigene Funktionen definiert werden
- Parameter in runden Klammern
- Optional: Daten mit **return** zurückgeben

```
def greeting(name):  
    print("Hello", name)
```

```
def potenz(basis,exponent):  
    return basis**exponent
```



# BEISPIEL: `print()`-FUNKTION

```
print(*objects, sep=' ', end='\n', ...)
```

- Parameter
  - **\*objects**: Liste an auszugebenden Objekten
  - **sep**: Trennzeichen (Standard: " ")
  - **end**: Abschlusszeichen (Standard: "\n")

```
print("Ananas", "Kiwi", "Mango")
```

```
print("Ananas", "Kiwi", "Mango", sep="---")
```

```
print("Ananas", end="\n...\n")  
print("Kiwi", end="\n***\n")  
print("Mango", end="\n____\n")
```

# **FORTGESCHRITTENE THEMEN**

**DATENSTRUKTUREN, BIBLIOTHEKEN**

# DATENSTRUKTUREN

- Programme arbeiten oft mit großen Datenmengen
  - Tabellen
  - Ordner..
- Tausende Variablen werden unhandlich
- Lösung: *Datenstrukturen*



Listen: **list**



Tupel: **tuple**



Dictionaries: **dict**

# LISTEN

- ermöglicht Listen von Daten zu speichern
- `my_list = ["beer", 5.0, 3, True]`
- Daten können hinzugefügt/entfernt/abgefragt werden
- Kann Daten unterschiedlicher Typen enthalten, meist jedoch nur 1 Typ

# LISTEN: FUNKTIONEN

- Für eindimensionale Daten (~ Excelzeile)
- vordefinierte Funktionen zB. für:
  - Listen sortieren `sort()`
  - Kommt X vor? `contains()`
  - Wie oft kommt X vor? `count()`
  - Tipp: Autovervollständigung (`TAB`-Taste)

[Dokumentation: Listen \(Google Python Kurs\)](#)

# SLICING

# DICTIONARIES

- Datenstruktur die eine Art Tabelle darstellt
- Schlüssel-Wert-Paare
  - Schlüssel muss eindeutig sein
  - Wert kann mehrmals vorkommen

[Dokumentation: Dictionaries\(Google Python Kurs\)](#)

# LIVE-CODING

A man with a mustache and glasses, wearing an orange polo shirt, is seated at a wooden desk. He is looking at a large, light-colored CRT computer monitor and has his hands on a keyboard. The desk also holds a black printer. In the background, there is a stone fireplace and a window with white curtains. The text "LIVE-CODING" is overlaid in large white letters across the center of the image.



# TEXTADVENTURE

# TEXTADVENTURE

- Ersten Computerspiele ohne Grafik
- Benutzer navigierte mit Text durch Labyrinth/Abenteuer
  - Räume wurden ebenfalls durch Text beschrieben

# DEMO: ZORKONLINE.NET

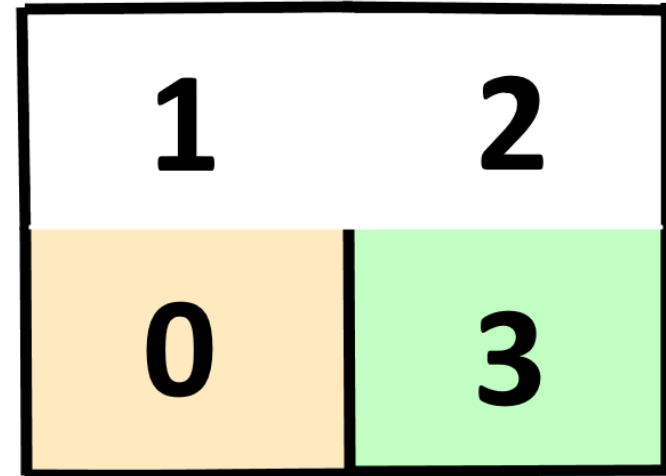
# TEXTADVENTURE PROGRAMMIEREN

- Daten:

- Start/Zielraum
- Beschreibung der Räume
- Beschreibung der Bewegungsmöglichkeiten

- Logik:

- Benutzereingaben übersetzen
- Spiel starten/beenden



A man in a military uniform, wearing a green garrison cap and a dark jacket, points his right index finger directly at the camera with a stern expression. The background is a plain, light-colored wall.

**DRILL TIME!**

