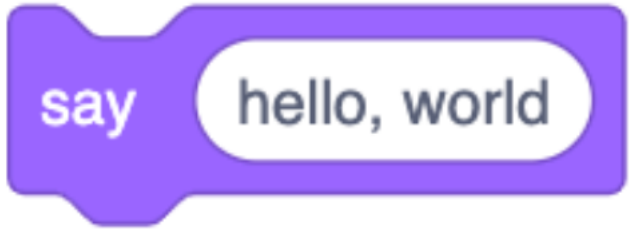
Challenge 04: Grundlagen der Programmierung mit Python

Prof. Dr. Markus Heckner

Bisher: Elemente einer Programmiersprache in Scratch



- Funktionen (mit Parametern und Rückgabewerten)
- Verzweigungen
- Boolsche Ausdrücke
- Schleifen
- Variablen
- ...

Eine neue Programmiersprache Python

- Alle Elemente in Scratch finden sich auch in Python wieder
- Python etwas weniger nutzerfreundlich – alles muss getippt werden
- Programmierer müssen das Vokabular der Programmiersprache sprechen (am Anfang nur wenige Wörter)
- Syntax am Anfang verwirrend – Je mehr Übung, desto leichter das Verständnis für das Lesen und Schreiben von Code

Abstraktion

- Zu Beginn versteht man nicht alle Details des Codes
- Zulassen, dass man nicht alles versteht und sich auf die Funktion bestehender Programmbestandteile verlassen
- Nur so kommt man weiter

Qualität von Quellcode

- **Korrektheit**, d.h. ob der Code die korrekte Lösung für ein Problem liefert
- **Design**, d.h. wie gut lesbar der Code ist und wie effizient er das Problem löst.
- **Stil**, d.h. ob der Code visuell ansprechend formatiert ist (d.h. Abstände, Einrückungen, etc.)

Hello World!

```
print("Hello world!")
```

Wo schreibe ich meinen Code?

- Theoretisch mit jedem Texteditor,
  aber…

Entwicklungsumgebungen
Integrated Development Environments (IDEs)

Programme schreiben erleichtert durch:

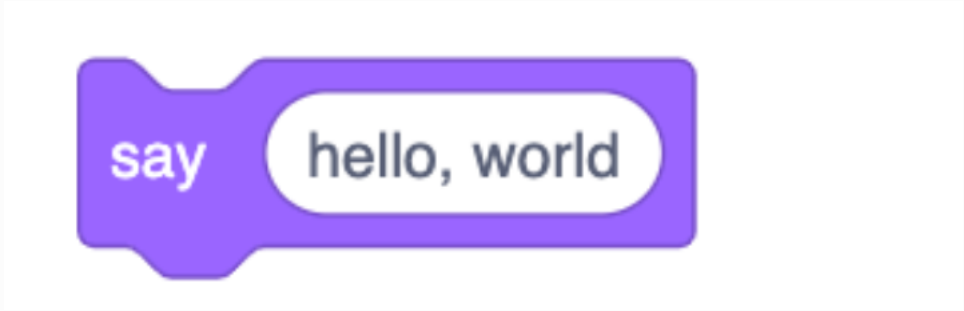
- Syntax-Highlighting
- Code-Vervollständigung
- Unterstützung Einrückung
- …

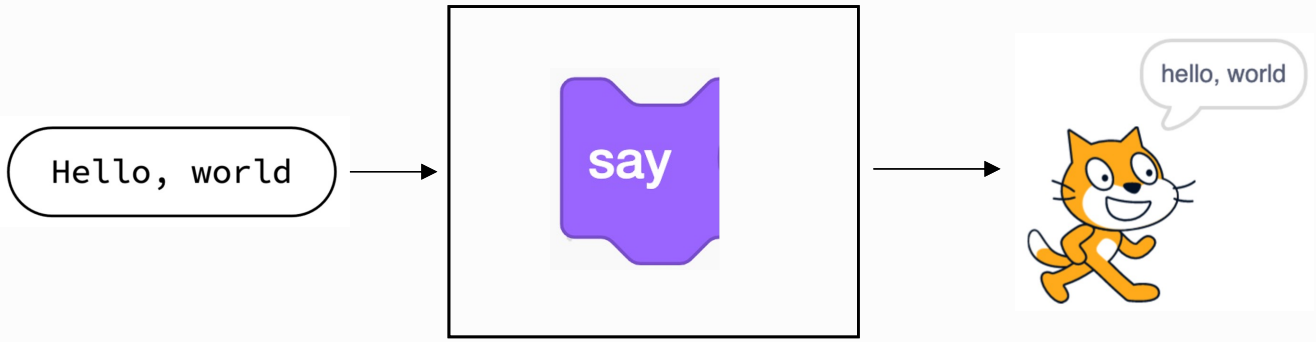
Wir verwenden die browserbasierte IDE
replit

Demo replit und hello.py

Funktionen, Parameter und Ausgaben

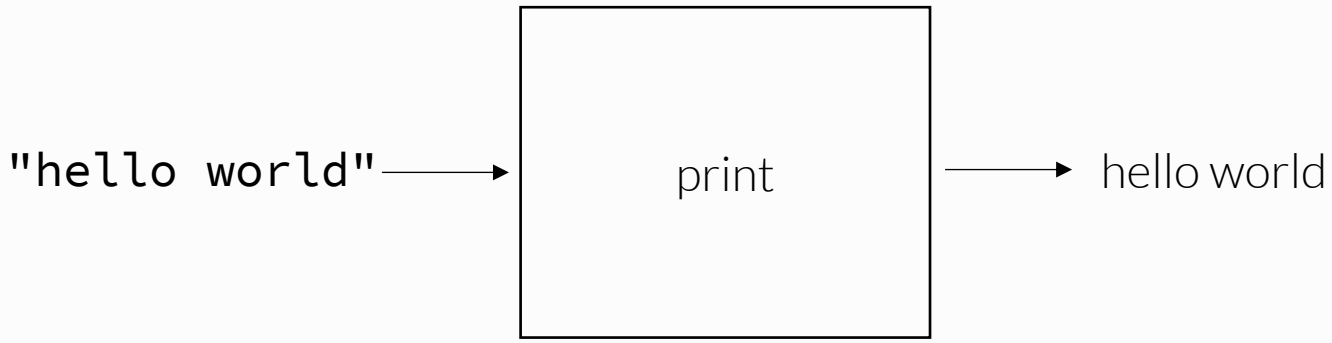
Funktionen, Parameter und Ausgaben



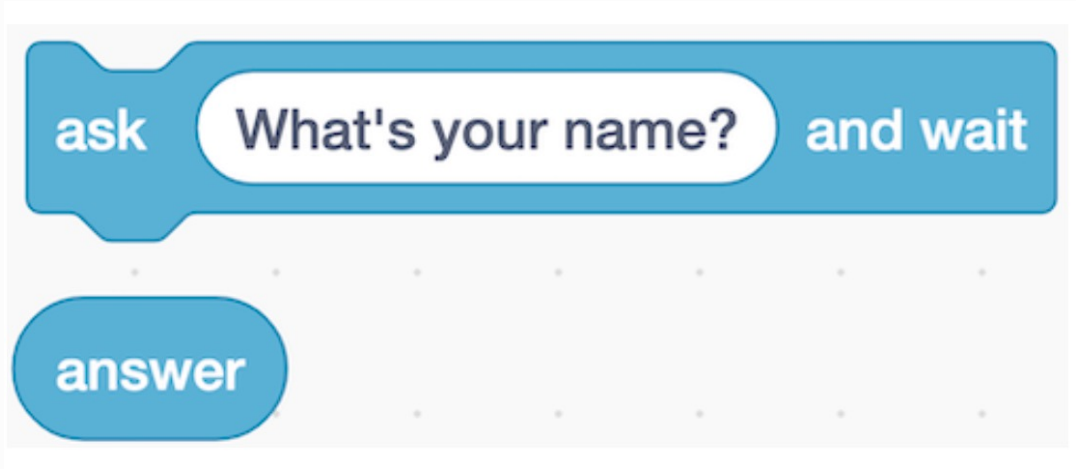


Funktionen, Parameter und Ausgaben

```
print("Hello world!")
```



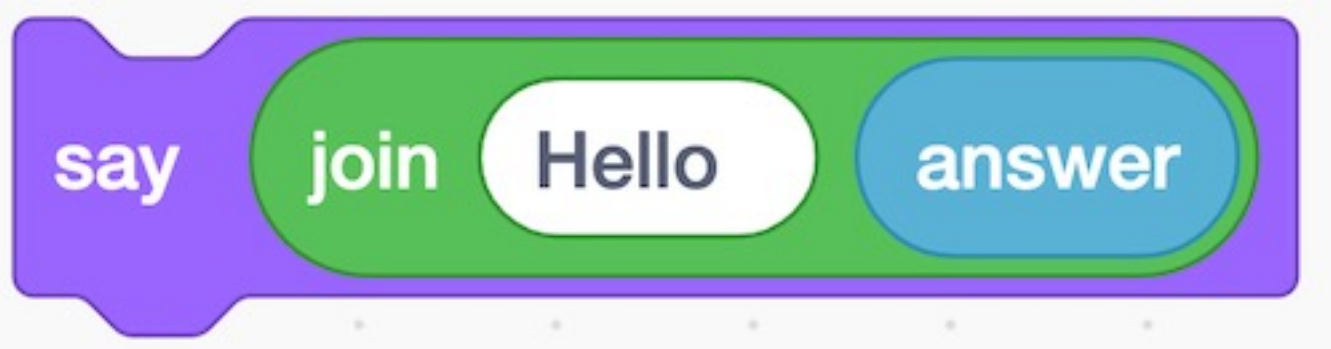
Funktionen, Parameter und Rückgaben




```
answer = get_string("What's your name? ")
```

Demo hello1.py

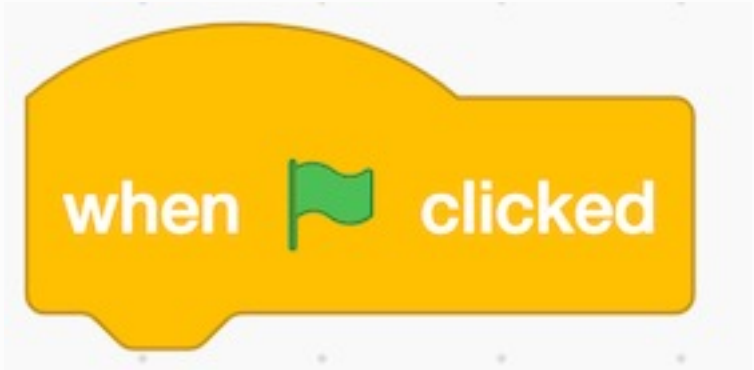
Konkatenierung, d.h.Zusammenfügen eines Strings in Scratch und Python




```
print(f"Hello {answer}")
```

Demo hello1.py continued

main als Start des Programms




```
from cs50 import get_string

def main():
    answer = get_string("What's your name?")
    print(f"Hello {answer}")
```

Demo hello2.py

Shell Kommandos

- `cd` (*change directory*) - wechselt das aktuelle Verzeichnis (= Ordner)
- `cp` (*copy*) - kopiert Dateien und Ordner
- `ls` (*list*) - zeigt die Dateien in einem Ordner an
- `mkdir` (*make directory*) - erstellt einen Ordner
- `mv` (*move*) - verschiebt Dateien und Ordner
- `rm` (*remove*) - löscht Dateien
- `rmdir` (*remove directory*) - löscht Ordner
- `touch` – legt eine neue Datei an
- ...

Variablen können unterschiedliche Werte annehmen

Datentypen

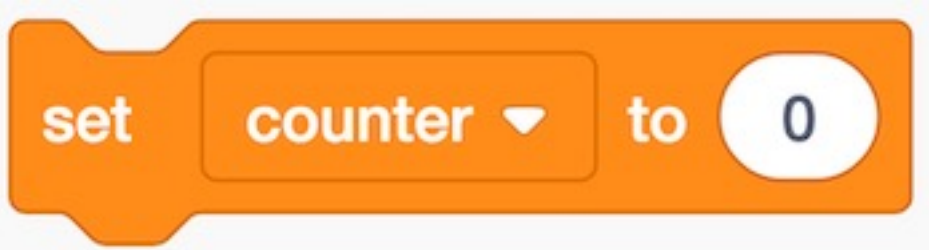
- `bool` - Kann entweder `true` (wahr) oder `false` (falsch) sein
- `str` - Zeichenketten
- `float` - Fließkommazahlen
- `int` - Ganzzahlen

Demo Anzeige von Datentypen hello3.py

Mathematische Operatoren

- +
- –
- *
- /
- % für Ermittlung des Rests bei ganzzahliger Division

Variablen und “syntactic sugar“



counter = 0

counter = counter + 1

counter += 1

Demo calculator.py

Verzweigungen
`if`


```
if x < y:
    print("x ist kleiner als y")
```

Verzweigungen
`if else`


```
if x < y:
    print("x ist kleiner als y")
else:
    print("x ist nicht kleiner als y")
```

Verzweigungen
`else if`


```
if x < y:
    print("x ist kleiner als y")
elif x > y:
    print("x ist größer als y")
elif x == y:
    print("x ist gleich y")
```

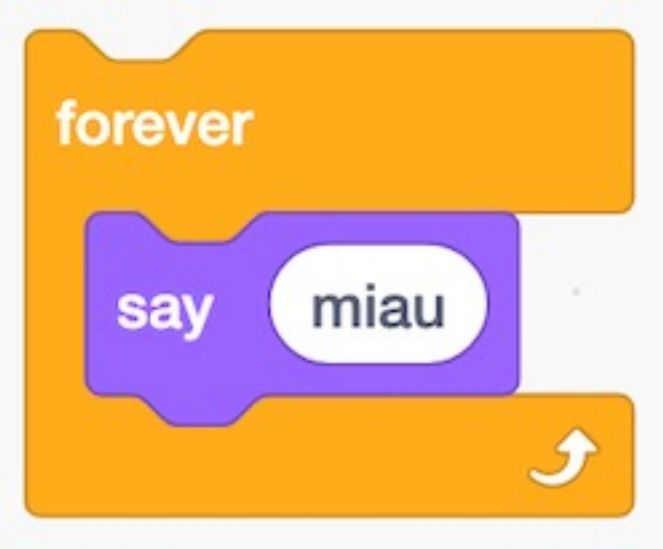
Demo conditionals.py, points0.py, parity.py

Dinge öfter tun

```
def main():
    print("miau")
    print("miau")
    print("miau")
```

Demo miau0.py

Schleifen – Dinge öfter tun endlos




```
while True:
    print("miau")
```

Demo miau endlos

Schleifen – Dinge öfter tun n-mal

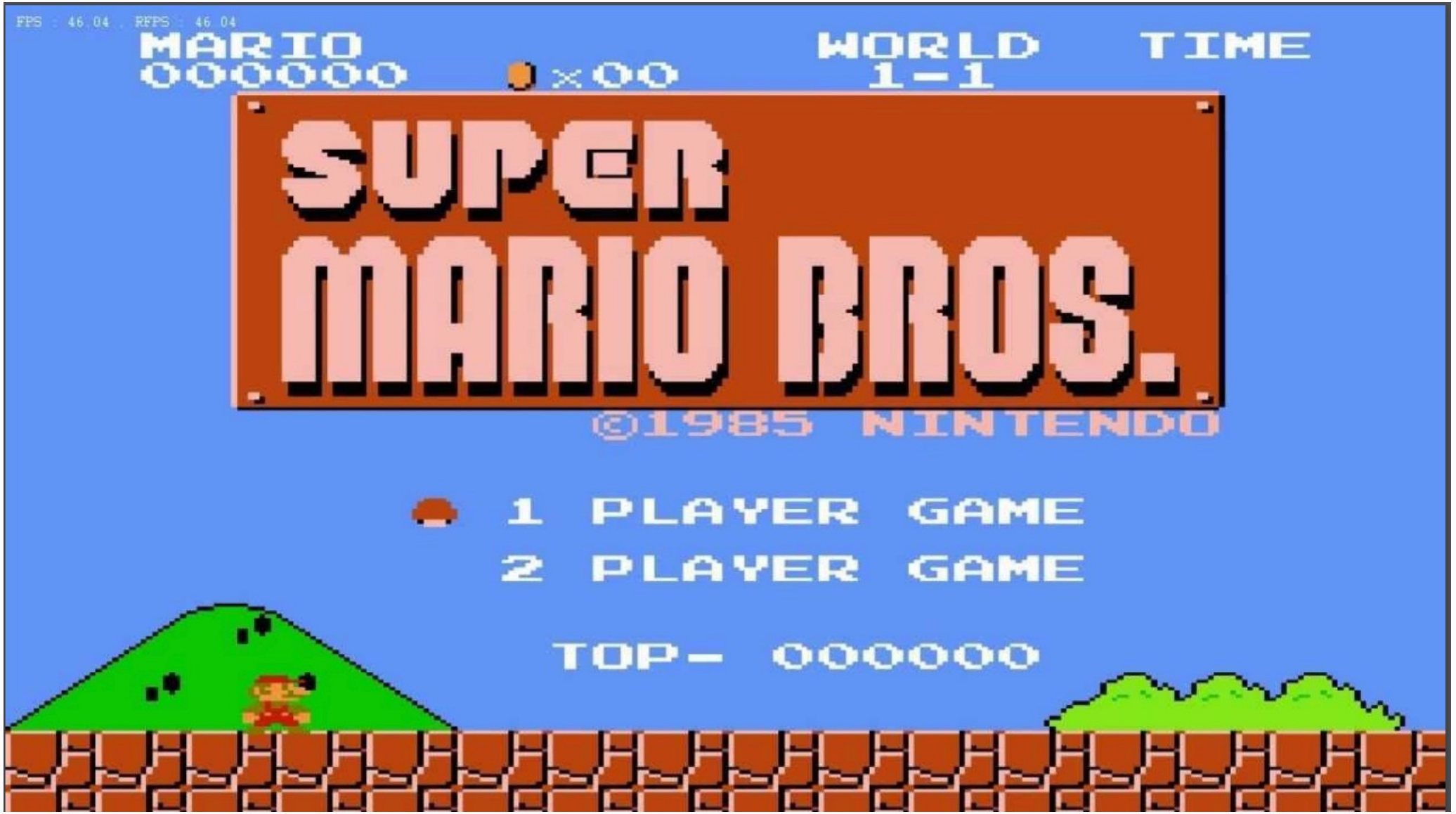

```
counter = 0

while counter < 3:
    print("miau")
    counter = counter + 1
```

Besser:

```
for i in range(3):
    print("miau")
```

Demo miau1.py


FPS : 46.04 . RFPS : 46.04
MARIO
000000
×00
WORLD
1-1
TIME
SUPER
MARIO BROS.
©1985 NINTENDO
1 PLAYER GAME
2 PLAYER GAME
TOP- 000000

```
$ python3 mario0.py
?
?
?
?
```

Demo mario0.py, mario1.py

```
$ python3 mario3.py
Height: 4
#
#
#
#
```


```
$ python3 mario4.py
Size: 4
####
####
####
####
```


Funktionen verbessern Lesbarkeit und ermöglichen das Zerlegen in Teilprobleme

```
def main():
    while True:
        n = get_int("Size: ")
        if n > 0:
            break

    for i in range(n):
        for j in range(n):
            print("#", end="")
        print()
```

Demo mario4-function.py