Teil 2

Konditionen & Iterationen

Fortsetzung: Flow of Control

Ternärer Operator

```
def main():
    x = int(input("Nenne eine ganze Zahl: "))
    if is_even(x):
        print("Gerade")
    else:
        print("Ungerade")

def is_even(n):
    return True if n % 2 == 0 else False

main()
```

match

```
name = input("What is your name? ")

match name:
    case "Alice":
        print("Hi, Alice.")
    case "Bob":
        print("Hi, Bob.")
    case "Sebastian" | "Seb":
        print("Hi, Sebastian.")
    case _:
        print(f"Hi, {name}.")
```

Loops (Schleifen)

for & while

for

- Iteriert über eine Sequenz
- continue (geht sofort zur nächsten Iteration)
- break (unterbricht die Schleife)

```python
# Beispiel 1
a = [1, 2, 3, 4]

for _ in a:
    print(_)


# Beispiel 2
# Probiere auch mal range(10, 20) und range(10, 20, 2); erkennst du das System wieder?
for n in range(10):
    print(n)


# Beispiel 3
text = "Shorten my text for Twitter, please!"

for buchstabe in text:
    if buchstabe.lower():
        if buchstabe in "aeiou":
            continue
    print(buchstabe, end="")
```

```python
# Beispiel 4
students = {"Tom":80, "Maria":95, "Lisa":90, "Karla":75, "Paul":60, "Peter":85}

for student in students:
    if students[student] >= 90:
        print(student, "hat eine 1")
    elif students[student] >= 80:
        print(student, "hat eine 2")
    elif students[student] >= 70:
        print(student, "hat eine 3")
    elif students[student] >= 60:
        print(student, "hat eine 4")
    else:
        print(student, "hat eine 5")
```

```python
# Beispiel 5
students = [
    {"name": "Peter", "points": 90, "yob": 2001},
    {"name": "John", "points": 92, "yob": 2004},
    {"name": "Anna", "points": 94, "yob": 2003},
    {"name": "Thomas", "points": 75, "yob": None},
    {"name": "Bob", "points": 81, "yob": 2002},
    {"name": "Donald", "points": 65, "yob": 2001},
    {"name": "Maria", "points": 96, "yob": 2003},
]

for student in students:
    if student["yob"] is None:
        continue
    print(student["name"], student["points"], student["yob"], sep=", ")
```

while

- Iteriert bis eine bestimmte Kondition erreicht wird
- continue (geht sofort zur nächsten Iteration)
- break (unterbricht die Schleife)
- Infinite loops

```python
# Beispiel 1
while True:
    user_input = input("Bitte ein positive, ganze Zahl eingeben: ")
    if user_input.isdigit():
        break


# Beispiel 2
a = 0
while a < 3:
    user_input = input("Bitte dreimal was mit mindestens drei Zeichen eingeben: ")
    if len(user_input) > 3:
        a += 1 # kann man statt a = a + 1 schreiben
        print(a)
```

```python
# Fortgeschritten:
# Wandle eine Zahl in Binärcode um

n = int(input("Gib eine Zahl ein: "))
rest = [] # Liste für die Reste
while n > 0: # solange n größer als 0 ist
    rest.append(n % 2) # % ist der Modulo-Operator; das Ergebnis wird der Liste angehangen
    n = n // 2 # // ist die Ganzzahldivision

print(rest[::-1]) # [::-1] reversiert die Liste
```

Falls du nicht mehr weiß, wie eine Binärzahl aufgebaut ist, dann schau dir diesen Link an:
https://www.matheretter.de/wiki/binarzahlen

Exceptions

Fehler bei der Ausführung

Exceptions

- NameError
- ValueError
- TypeError
- KeyError
- IndexError
- ZeroDevisionError
- StopIteration
- (...)

Liste mit Exceptions: https://docs.python.org/3/library/exceptions.html#bltin-exceptions

Verschieden dazu: SyntaxError

```python
# Der nachfolgende Code setzt voraus, dass der Nutzer kooperativ ist
# und eine Zahl eingibt
x = int(input(„Was soll x sein? "))
print("x ist {zahl}".format(zahl=x))

# Zur Erinnerung: Es gibt verschiedene Varianten
print("x ist {}".format(x))
print("x ist", x)
print("x ist " + str(x))
print(f"x ist {x}")

# Gibt er allerdings kein Integer an, dann kommt es zu einem Fehler:
```

```
Was soll x sein? abc
Traceback (most recent call last):
  File "c:/Users/witzmann/test2/exceptions.py", line 1, in <module>
    x = int(input("Was soll x sein? "))
ValueError: invalid literal for int() with base 10: 'abc'
```

Tracebacks

Abfangen von Exceptions

- try
- except
- else
- finally

Ausgeben eigener Exceptions

- raise

Fortgeschritten: Eigene Exceptions definiere und damit Einblicke in den Code verhindern.

```
# Iteration T0
try:
    x = int(input("Was soll x sein? "))
    print("x ist {zahl}".format(zahl=x))
except ValueError:
    print("Bitte gib ein Integer an.")

# Achtung: Der folgende Code führt zu einem NameError
try:
    x = int(input("Was soll x sein? "))
    print("x ist {zahl}".format(zahl=x))
except ValueError:
    print("Bitte gib ein Integer an.")

print(x)
```

```python
# Iteration T1
try:
    x = int(input("Was soll x sein? "))
    print("x ist {zahl}".format(zahl=x))
except ValueError:
    print("Bitte gib ein Integer an.")
else:
    print(f"{x} ist ein Integer.")

# Iteration T2
while True:
    try:
        x = int(input("Was soll x sein? "))
        print("x ist {zahl}".format(zahl=x))
    except ValueError:
        print("Bitte gib ein Integer an.")
    else:
        print(f"{x} ist ein Integer.")
        break
```

```python
# Iteration T1
try:
    x = int(input("Was soll x sein? "))
except ValueError:
    print("Bitte gib ein Integer an.")
else:
    print(f"{x} ist ein Integer.")

# Iteration T2
while True:
    try:
        x = int(input("Was soll x sein? "))
    except ValueError:
        print("Bitte gib ein Integer an.")
    else:
        print(f"{x} ist ein Integer.")
        break

# Iteration T3
while True:
    try:
        x = int(input("Was soll x sein? "))
        break
    except ValueError:
        print("Bitte gib ein Integer an.")

print(f"{x} ist ein Integer.")
```

```python
# Iteration T4
def main():
    x = get_int()
    print(f"{x} ist ein Integer.")


def get_int():
    while True:
        try:
            return int(input("Was soll x sein? "))
        except ValueError:
            print("Bitte gib ein Integer an.")

main()


# Iteration T5
def main():
    x = get_int("x")
    y = get_int("y")
    print(f"{x} + {y} = {x + y}")


def get_int(varName="x"):
    while True:
        try:
            return int(input(f"Was soll {varName} sein? "))
        except ValueError:
            print("Bitte gib ein Integer an.") # Alternativ: pass?


main()
```

Libraries

Beispiel: random

- randint
- choice
- shuffle
- (…)

https://docs.python.org/3/library/random.html

https://docs.python.org/3/library/random.html

random.**shuffle**(x)

Shuffle the sequence x in place.

To shuffle an immutable sequence and return a new shuffled list, use `sample(x, k=len(x))` instead.

Note that even for small `len(x)`, the total number of permutations of x can quickly grow larger than the period of most random number generators. This implies that most permutations of a long sequence can never be generated. For example, a sequence of length 2080 is the largest that can fit within the period of the Mersenne Twister random number generator.

Deprecated since version 3.9, removed in version 3.11: The optional parameter random.

```python
import random

cards = ["7", "8", "9", "10", "Bauer", "Königin", "König", "Ass"]
random.shuffle(cards)
print(cards[0])

# Variante
from random import shuffle

shuffle(cards)
print(cards[0])

# Noch eine Variante
from random import shuffle as mix

mix(cards)
print(cards[0])
```

Beispiel: sys

- argv (Liste der Argumente, die dem Script beim Aufruf übergeben wurden; [0] ist der Skriptname: command-line arguments)
- exit (Fehlermeldung [string] kann als Argument übergeben werden oder z. B. auch ein Exitcode)
- (…)

https://docs.python.org/3/library/sys.html

```python
import sys

for arg in sys.argv:
    print(arg)

# Variante?
if sys.argv[1]:
    print(sys.argv[1])

# Variante?
i = 1
while i < len(sys.argv):
    print(sys.argv[i])
    i += 1

# Probiere dies mit einem Argument und ohne Argument aus.
# Was ist hier passiert?
try:
    print(sys.argv[1])
except IndexError:
    sys.exit("Bitte geben Sie mindestens ein Argument an.")
```

Packages

https://pypi.org/

Package Manager pip

pip install

Virtual Environment

Versuche mal

- `cowsay`

`(Weitere folgen …)`

```
import cowsay

print(cowsay.get_output_string('trex‘, „WOW“))
```

APIs

Modul: requests

https://requests.readthedocs.io/en/latest/

JSON (JavaScript Object Notation)

Programmiersprachen unabhängig

Versuche mal

`https://api.currently.io/v2.0/gsi/marketdata?zip=40597`

`https://world.openfoodfacts.org/api/v2/search?code=3263859883713`

```python
import requests
import random
from prompt_toolkit import print_formatted_text, HTML

# https://opentdb.com/api_config.php
url = "https://opentdb.com/api.php?amount=10&category=15"

response = requests.get(url)
result = response.json()

for i in range(10):
    print_formatted_text(HTML(result["results"][i]["question"]))
    if result["results"][i]["type"] == "multiple":
        choice = []
        for j in range(3):
            choice.append(result["results"][i]["incorrect_answers"][j])
        choice.append(result["results"][i]["correct_answer"])
        random.shuffle(choice)
        for j in range(4):
            print_formatted_text(HTML(f"{j+1}: {choice[j]}"))
    input("See the answer: ")
    print_formatted_text(HTML(result["results"][i]["correct_answer"]))
    print("")
```