ORGANISATORISCHES

- Du: Sebastian
- Whatsapp: 0172 690 84 57
- 17.03. + 24.03., 31.03., 14.04., 21.04., 28.04.
- 6 Stunden **Workshop** am12.05. (19.05. fällt dafür aus)
- 6 Stunden **Hackathon** am 26.05. (09.06. fällt dafür aus)
- 6 Stunden Vertiefung und Klausurvorbereitung am 16.06. (23.06. fällt dafür aus)

Workshop und Hackathon finden in Düsseldorf Benrath statt: Benrather Schloßallee 99

Teil 1

Funktionen & Variablen

Fortsetzung

str

https://docs.python.org/3.11/library/stdtypes.html#string-methods

```python
def talk():
    def user_name():
        name = input("Wie heißt du? ")
        return name
    name = user_name().strip()
    print(f"Hallo {name}!")


talk()
```

int

Objects

- Class
- Instance
- ID (Identity)
- Type
- Value
- Methods
- Magic Methods (__XXX__) [2 x _]
- Body of Class

```python
class Dog:
    def __init__(self, name):
        self.name = name

    def bark(self):
        print("Woof!")

    def whoami(self):
        print("I am a dog. My name is " + self.name + "!")


barnie = Dog("Barnie")
barnie.bark()
barnie.whoami()
```

Name & Namespace

```python
class Dog:
    def __init__(self, name):
        self.name = name

    def speak(self):
        print("Woof!")

    def whoami(self):
        print("I am a dog. My name is " + self.name)


class Human:
    def __init__(self, name):
        self.name = name

    def speak(self):
        print("Hello!")

    def whoami(self):
        print("I am a human. My name is " + self.name)


class Cat:
    def __init__(self, name):
        self.name = name

    def speak(self):
        print("Meow!")

    def whoami(self):
        print("I am a cat. My name is " + self.name)


barnie = Dog("Barnie")
paul = Human("Paul")
felix = Cat("Felix")

barnie.speak()
barnie.whoami()
paul.speak()
paul.whoami()
felix.speak()
felix.whoami()
```

Pointer

```
a = 5
print(id(a))
a = 6
print(id(a))
b = a
print(id(b))
b = 9
print(id(b))
```

Eine Minute Übung

Was wird ausgegeben?

```
class Human:
    def __init__(self, name):
        self.name = name

    def speak(self):
        print("Hello, my name is " + self.name)

def speak():
    print("Hello, my name is ... uh")

peter = Human("Peter")
speak()
```

Noch eine Minute Übung

Welche Aussagen sind richtig?

a) Objekte haben Parameter, Typen und Werte

b) Objekte haben ID, Type & Value

c) Objekte haben Argumente, Werte und IDs

d) Objekte haben keine IDs

e) Objekte haben keine Werte

Libraries

Datatypes

- Mutable / immutable
- Alles Objekte (in Python)
- Eigene Objekte sind mutable, sofern man es nicht anders programmiert

Zahlen

immutable

Integers

- int
- Eingabe string: Umwandeln
- Unlimitert (Speicher als Grenze)
- Positiv, Negativ und Null
- Zuweisung kann mit Unterstrich passieren (z. B.: n = 1_000)
- Unterstützt alle Basis-Operatoren aus der Mathematik:
  - + (Addition), - (Subtraktion), * (Multiplikation), / ([True] Division), // (Int Division), % (Modulo), ** (Potenz)

```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more
information.
>>> 7 // 4
1
>>> -7 // 4
-2
>>>
```

Boolean

- bool
- Subclass von Integer
- True oder False
- True == 1
- False == 0
- Operatoren: and, or, not
- Upcast zu Integer

```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more
information.
>>> int(True)
1
>>> bool(1)
True
>>> bool(-42)
True
>>> not True
False
>>> True and True
True
>>> False or True
True
>>> True + 6
7
>>> False + 42
42
>>>
```

Reale Zahlen

- Float
- Begrenzt

Komplexe Zahlen

- Realer und Imaginärer Teil

Brüche und Dezimale

- from fractions import Fraction
- from decimal import Decimal
- Präzise

```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> 0.3 - 0.1 * 3 # ist nicht Null!
-5.551115123125783e-17
>>> from fractions import Fraction # wir importieren die Klasse Fraction aus der Library
fractions
>>> Fraction(10,6) # wird automatisch vereinfacht
Fraction(5, 3)
>>> Fraction(1,3) + Fraction(2,3)
Fraction(1, 1)
>>> from decimal import Decimal as D # wir importieren die Klasse Decimal aus der Library
decimal und geben ihr einen neuen Namen D
>>> D(3.14) # als float
Decimal('3.140000000000000124344978758017532527446746826171875')
>>> D('3.14') # als String
Decimal('3.14')
>>> D(0.3) - D(0.1) * D(3) # gleiches Problem
Decimal('-2.775557561565156540423631668E-17')
>>> D('0.3') - D('0.1') * D('3') # nun als String: geht!
Decimal('0.0')
>>>
```

Immutable (unveränderliche) Sequenzen

Strings, Tuples und Bytes

Strings

- Chars sind String mit einer Länge von 1
- Immutable Sequenz von Unicode
- ‘,“,“““,‘‘‘
- Escape /
- Länge der Sequenz: len(string)
- Encode / Decode (z. B. utf-8)
- bytes Objekt (z. B. text = b“ein byte objekt“)
- Indexing und Slicing, z. B. [0], [:2], [2:], [2:2], [2:2:3], [:]
- Formatierung: {}, {0}, {name}
- f-String
- Zugriff auf Position [ ]

```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> 0.3 - 0.1 * 3 # ist nicht Null!
-5.551115123125783e-17
>>> from fractions import Fraction # wir importieren die Klasse Fraction aus der Library
fractions
>>> Fraction(10,6) # wird automatisch vereinfacht
Fraction(5, 3)
>>> Fraction(1,3) + Fraction(2,3)
Fraction(1, 1)
>>> from decimal import Decimal as D # wir importieren die Klasse Decimal aus der Library
decimal und geben ihr einen neuen Namen D
>>> D(3.14) # als float
Decimal('3.140000000000000124344978758017532527446746826171875')
>>> D('3.14') # als String
Decimal('3.14')
>>> D(0.3) - D(0.1) * D(3) # gleiches Problem
Decimal('-2.775557561565156540423631668E-17')
>>> D('0.3') - D('0.1') * D('3') # nun als String: geht!
Decimal('0.0')
>>>
```

Teil 2

Konditionen & Iterationen

Flow of Control

Vorschau

Syntax

- >
- >=
- <
- <=
- ==
- !=

Achtung: = ist für Zuweisung und == ist die Prüfung, ob etwas gleich ist

if

- if
- else
- elif

Fragt nach einer Boolean-Antwort, True oder False.

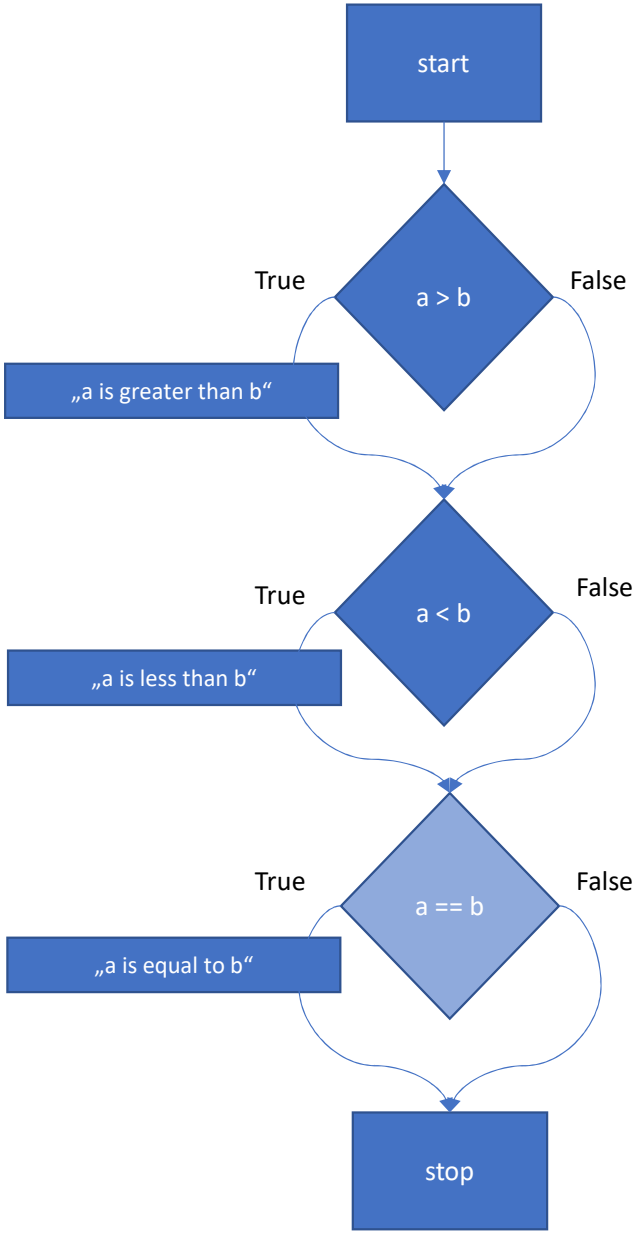
if mit Variable alleine fragt nach Existenz der Variable

Beispiel

```
a = input('Enter a number for a: ')
b = input('Enter a number for b: ')

if a > b:
    print('a is greater than b')
if a < b:
    print('a is less than b')
if a == b:
    print('a is equal to b')
```

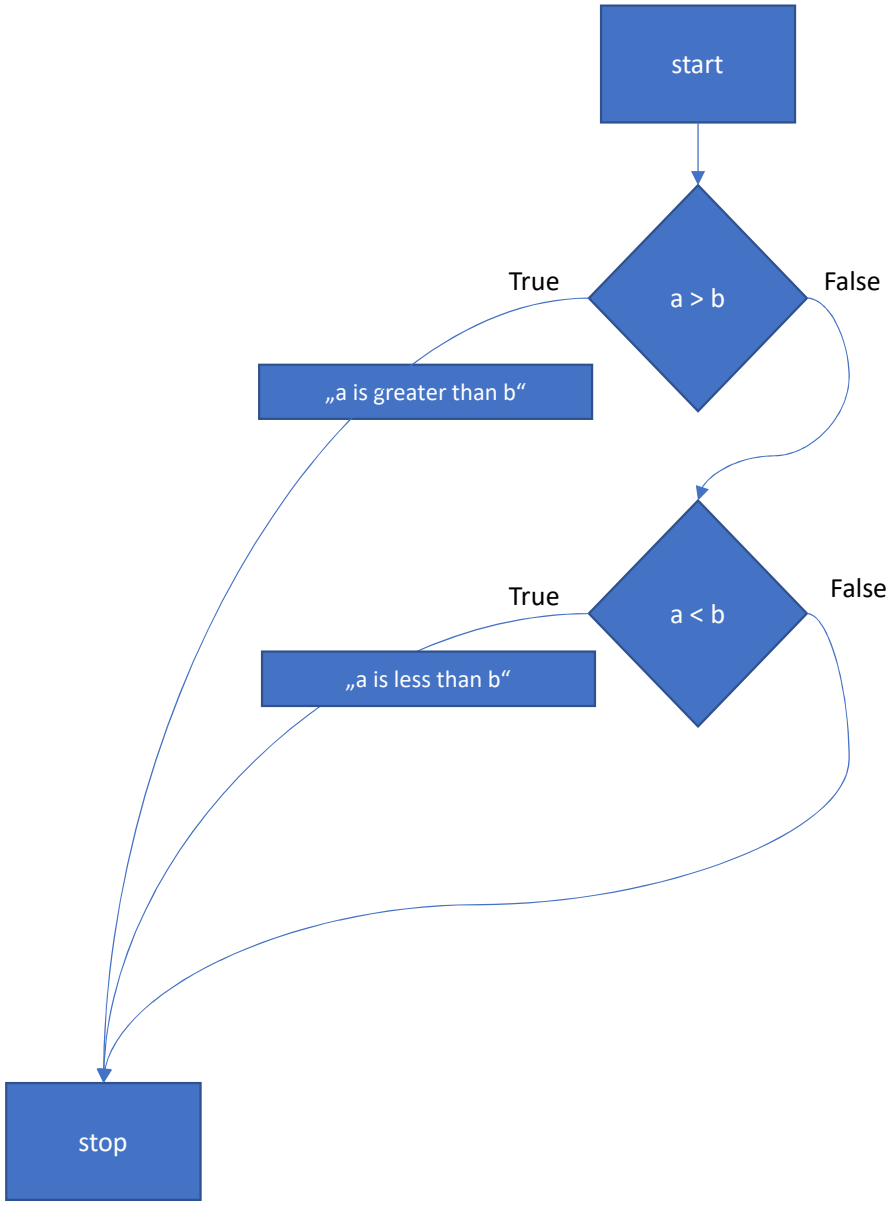
Wie geht es besser?



Design

```
a = input('Enter a number for a: ')
b = input('Enter a number for b: ')

if a > b:
    print('a is greater than b’)
elif a < b:
    print('a is less than b’)
else:
    print('a is equal to b')
```



if

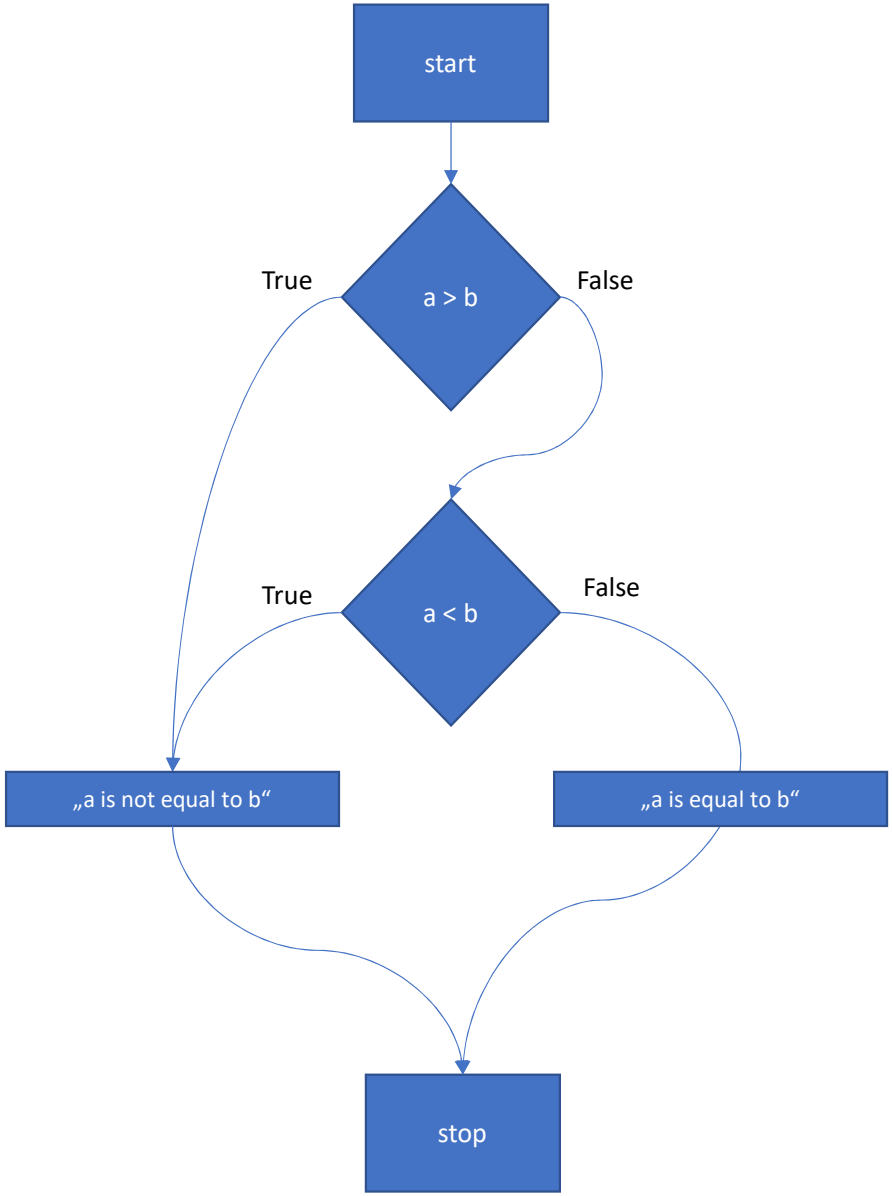
- and
- or

Beispiel

```
a = input('Enter a number for a: ')
b = input('Enter a number for b: ')

if a > b or a < b:
    print('a is not equal to b')
else:
    print('a is qual to b')
```

Wie geht es besser?

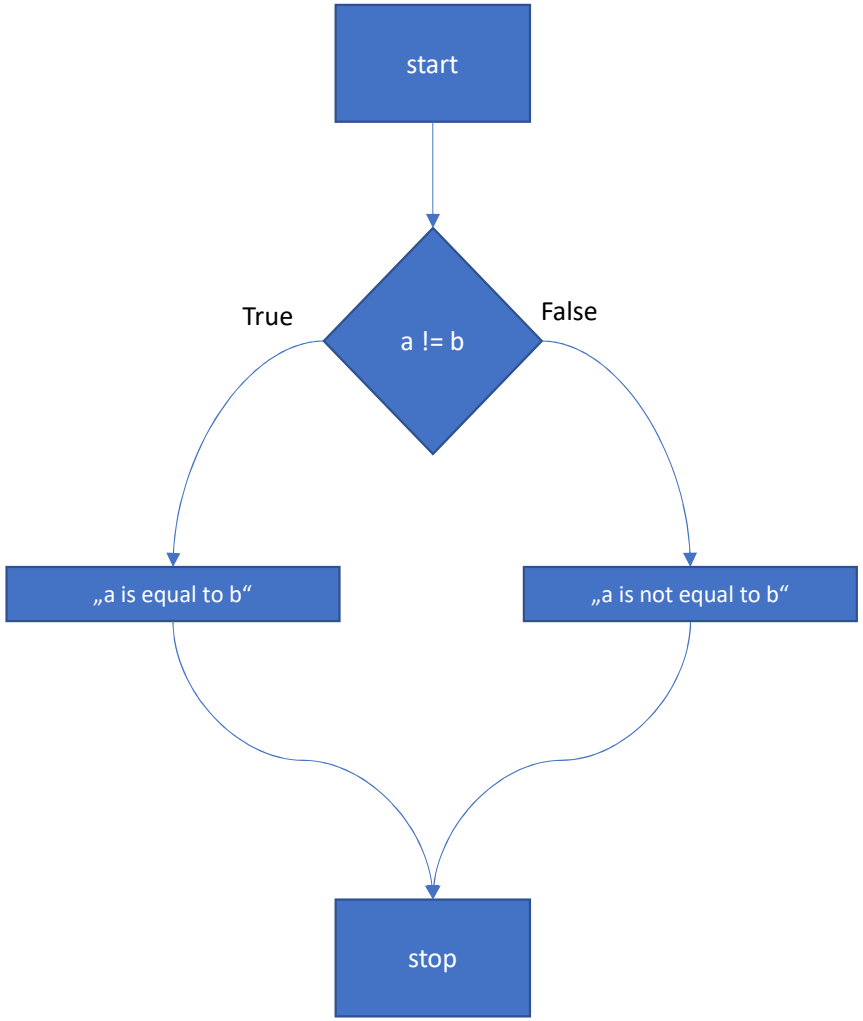


Design

```
a = input('Enter a number for a: ')
b = input('Enter a number for b: ')

if a != b:
    print('a is not equal to b')
else:
    print('a is qual to b')
```

Versuche es auch mit ==



Übungsaufgabe 3

Das Modulo 97-Verfahren

Als Benutzer:in möchte ich eine deutsche IBAN (mit Leerzeichen) eingeben und dann als Ausgabe die Prüfziffer erhalten, damit ich diese abgleichen kann.

Akzeptanzkriterien:

Bei der Eingabe “DE44 35070024 0388249600” erfolgt die Ausgabe 44.

Bei der Eingabe “DE44 35070024 7388249600” erfolgt die Ausgabe 20.

Bei der Eingabe “DE 89 37040044 0532013000” erfolgt die Ausgabe 89.

Hinweis:

Die IBAN besteht aus 22 Zeichen. Die ersten beiden beinhalten das ISO-Länderkennzeichen, das in unserem Fall immer “DE” ist. Dann folgen die zwei Prüfziffern, die wir durch unsere Berechnung nachkalkulieren wollen. Im Anschluss kommen acht Ziffern Bankleitzahl und 10 Ziffern für die Kontonummer. Fehlende Ziffern bei der Kontonummer werden von links durch Nullen aufgefüllt.

Die Prüfziffer kontrolliert die korrekte Zusammenstellung der IBAN. Sind Fehler in der Kontonummer oder der Bankleitzahl, dann passt die Prüfziffer nicht mehr.

Die Berechnung der Prüfziffer erfolgt durch die Ziffern, die dem Länderzeichen und der Prüfziffer folgen. Die 18 Ziffern wird am Ende 131400 angehangen. Die 13 und die 14 stehen für die Buchstaben, wobei ein A eine 10 wäre – hier also 13 für D und 14 für E. Dann zwei Nullen für die Prüfziffer. Die nun 24-stellige Zahl wird durch 97 geteilt und der Rest errechnet. Der Rest wird nun von der festgelegten Zahl 98 abgezogen. Das Ergebnis ist die Prüfziffer.

Tipps: 1. Erinnere dich auch an das Indexing und Slicing : Strings sind Sequenzen und es daher möglich z. B. mit name_der_variablen[0] auf das erste Zeichen des strings zuzugreifen. Versuche auch [2:4]. 2. Es gibt verschiedene Operatoren für integer. Schau sie dir genauer an.

Übungsaufgabe 4

Wie viele Tage bin ich alt?

Als Benutzer:in möchte ich meinen Geburtstag im Format TT.MM.JJJJ angeben und erhalte als Ausgabe wie viele Tage seither vergangen sind.

Akzeptanzkriterien:

Bei der Eingabe “21.06.2001” würde man am 17.03.23 die Ausgabe “7940” erhalten.

Tipp:

Schau die Library “datetime” an: https://docs.python.org/3/library/datetime.html

Übungsaufgabe 5

Quizfrage

Als Benutzer:in möchte ich eine Quizfrage gestellt bekommen und auf meine Antwort erfahren, ob diese falsch oder richtig war.

Akzeptanzkriterien:

Frage: Strings sind in Python veränderbare oder unveränderlich?

Wenn die richtige Antwort “unveränderlich” oder “unveraenderlich” angegeben wird, gibt das Programm “Richtig” aus, sonst “Falsch”.

Übungsaufgabe 6

Snackautomat

Als Benutzer:in möchte ich einen Snack auswählen und become dann den Preis angezeigt.

Akzeptanzkriterien:

Wenn “Snickers” oder “Mars” eingegeben wird, gibt das Programm “Ein Euro” aus.

Wenn “Nüsse” eingegeben wird, dann “Zwei Euro” und

bei “Apfel” kommt “50 Cent”.

Ansonsten: “Haben wir nicht.”

Übungsaufgabe 7

Dateitypen

Als Benutzer:in möchte ich einen Dateinamen angeben und erhalte bei Gif, JPG, JPEG und PNG den Media-Typ angezeigt.

Akzeptanzkriterien:

Für die Endungen .gif, .jpg, .jpeg und .png wird der korrekte MIME Typ ausgegeben, wie z. B. image/gif für .gif. Gebe ich also “auto.jpg” an, dann erhalte ich die Antwort image/jpeg.

Die Software ist case-insesitive (https://de.wikipedia.org/wiki/Case_sensitivity). Das Beispiel von oben funktioniert also auch bei “auto.JPG”.

Bei allen anderen Endungen kommt als Antwort “application/octet-stream”, was der Standard ist.

Hinweis:

Alle Browser verlassen sich auf Media Typen (vormals: MIME), um die Darstellung zu verarbeiten. Um den Typen zu bestimmen, wird die Dateiendung überprüft. Die MIME-Typen finden sich hier https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types/Common_types

Übungsaufgabe 8

Taschenrechner

Als Benutzer:in möchte ich einfache Matheaufgaben eingeben können und eine korrekte Antwort erhalten.

Akzeptanzkriterien:

Es kann eine Rechenaufgabe mit zwei Zahlen eingeben warden. Die Aufgabenoperatoren können +, -, * oder / sein.

Die Eingabe “1 + 1” wird verstanden und das Rechenergebnis ausgegeben. Ebenso diese Eingaben:

- 100 / 4
- 5 * 3
- -1 + 10

Es ist davon auszugehen, dass immer ein Leerzeichen zwischen der Zahl und dem Operator ist.

Tipp: Erinnere dich daran, dass das Objekt str verschiedene Methoden hat, die zum Beispiel split(). Split zerteilt den String in verschiedene Sequenzen, die alle zugleich zugewiesen werden können: x, y, z = eingabe.split(“ “)

Übungsaufgabe 9

Welche Zeit ist?

Als Benutzer:in möchte ich eine Zeitangabe im 24-Stunden Format angeben können und wenn es zwischen 16:00 und 22:00 ist, dann vom Programm die Meldung erhalten, dass “Programmierzeit” ist. Zwischen 8:00 bis 16:00 Uhr ist “Unizeit” und ab 22 Uhr bis 3:00 ist “Partyzeit” und 3:00 bis 8:00 ist “Schlafenszeit”.

Akzeptanzkriterien:

Wandle die String-Eingabe in ein Float um und entscheide anhand des Floats die Zeit.

Gebe ich “7:30” ein, dann erhalte ich die Antwort: “Schlafenszeit”.

Gebe ich eine z. B. “21:45” ein, dann “Programmierzeit”.

Tipp: Denk daran, dass eine Stunde aus 60 Minuten besteht ☺