Tutorium 03 - 2024-10-31

Blatt 02, Workflow, Blatt 03

Blatt 02

Welche Fehler wurden gemacht?

Hochladen

- Probleme beim Hochladen
  - Binary statt Textdatei
  - Tutorial: Beispiel Workflow mit Git

keine korrekte Ausgabe

- Expectation:

```
p = -4.0
q = -5.0
x1 = 5.0
x2 = -1.0
```

- Reality:

```
Geben Sie nun einen Wert für p an:
> -4.0
Geben Sie nun einen Wert für q an:
> -5.0
Die Lösungsmenge ist L := {5.0, -1.0}
```

- Gebt bitte alles exakt wie vorgeschrieben in der Console aus!

One-Liner in 2.4

```
print((((((float(input())) * 2) + 3) ** 2) - (float(input()))) / 4)
```

Falsch, weil `input` zwei mal aufgerufen wird

One-Liner in 2.4

```
print((((((float(x := input())) * 2) + 3) ** 2) - (float(x))) / 4)
```

Das wäre korrekt, aber `:=` kam noch nicht in der Vorlesung vor

One-Liner in 2.4

Durch umformen konnte man auch mit den bekannten mitteln eine einzige Zeile haben. Wurde aber nicht gefraget

```
print((float(input()) + 11 / 8) ** 2 - 121 / 64 + 9 / 4)
```

Blatt 02 - Vorrechnen

Recap Vorlesung

Funktionen, Loops

Funktionen

- Ihr habt schon die ganze Zeit mit Funktionen gearbeitet
  - `print(...)`
  - `int(...)`
  - ...
- Funktionen haben ihren eigenen *Scope*
- Funktionen sollten auch nur lokale Objekte verwenden

```python
def my_func(param1: type, param2: type) -> type:
  ...

if __name__ == '__main__':
  my_func(...)
```

```python
def add(a: int, b: int) -> int:
  return a + b

if __name__ == '__main__':
  print(add(1, 2)) # 3
```

Was ist `if name == 'main'?

- Damit bestimmter Code auch nur ausgeführt wird wenn wir die Datei als Hauptdatei ausführen
- Beim importieren von anderen Python-Datein möchten wir nicht dass irgend etwas Ausgegeben wird/Eingaben erfragt werden

Iteration - for

```
for some_item in some_iterable:
  ...
```

- Wenn wir für jedes *Element* in einer *Menge* machen wollen

```python
def print_even_numbers(max: int):
  for num in range(0, max + 1):
    print(num)
```

Iteration - while

```
while condition:
  ...
```

- Wenn wir eine Bendingung haben die vor jedem Durchlauf gelten muss

```python
def find_next_prime(n: int) -> int:
  next_prime = n + 1
  while !is_prime(next_prime):
    next_prime += 1
  return next_prime
```

Blatt 03