Python

ASE Übung

10.05.2019

Python

- Einfach
- Abstrakt
- Umfangreich
- In Wissenschaften bewährt
- `https://www.python.org/`

Programmablauf

- Skripte / Module
- Anweisungen zeilenweise ausgeführt
- Kontrollfluss
- Python Interpreter (Kommandozeile)

$\Rightarrow$ `python some_python_script.py`

Interaktiver Modus

- Direkte verarbeitung von Anweisungen
- Start des Interpreters ohne Argument `python`

```
Python 3.6.8 |Anaconda, Inc.| (default, Dec 30 2018, 01:22:34)
[GCC 7.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Variablen

```
some_integer_variable = 12
some_floating_point_variable = 12.345
some_string_variable = "This is free text."
some_boolean_variable = True
another_boolean_variable = False
```

Arithmetik

```
>>> a = 2
>>> b = 3
>>> a + b
5
>>> a * 4
8
>>> 12 / b
4.0
```

Relationen

```
>>> a == b
False
>>> (a + 1) != b
False
>>> a < b
True
>>> b >= 3
True
>>> b > 3
False
```

Relationen (cont.)

```
>>> "text" == "text"
True
>>> "text" == "test"
False
```

Aufrufen von Funktionen

```
>>> some_string = "abcde"
>>> length = len(some_string)
>>> length == 5
True
>>> print(some_string)
abcde
```

Indexierung

```
>>> some_string = "abcde"
>>> print(some_string[2])
c
```

Bedingte Anweisungen und Verzweigungen

```
>>> a = 10
>>> b = 20
>>> if a < b:
...     print("a is smaller than b.")
... else:
...     print("a is larger than b.")
...
a is smaller than b.
```

Schleifen

```
>>> some_string = "abcde"
>>> i = 0
>>> length = len(some_string)

>>> while i < length:
...     print(some_string[i])
...     i = i + 1
...
a
b
c
d
e
```

Schleifen (cont.)

```
>>> i = 0
>>> while True:
...     i = i + 1
```

Definition eigener Funktionen

```
>>> def get_smaller_number(a, b):
...     if a < b:
...             return a
...     else:
...             return b
...
>>> get_smaller_number(10, 20)
10
```

Sequenzidentität

```
ACGTACGTAC
||  ||   |||
ACTTAGCTAC
```

$$\Rightarrow 100\% \times \frac{7}{10} = 70\%$$

Skript

```
mkdir ase_uebung_04
cd ase_uebung_04
cp /home/olzhabaev/Public/seq_identity.py .
```

`compute_sequence_identity(...)`

1. Längen der Eingabesequenzen ermitteln
2. Längen vergleichen
3. Wenn ungleich, Text ausgeben
4. Anderenfalls Matches zählen
   - 4.1 Match-Zählervariable (z.B. `num_matches`) anlegen
   - 4.2 `while` Schleife mit Indexvariable (z.B. `i`) erzeugen
   - 4.3 Pro Iteration `i`: Sind Sequenzen an Position `i` gleich? Wenn ja, Zählervariable inkrementieren
   - 4.4 Indexvariable inkrementieren
5. Matches normalisieren, zurück geben

Testdaten

```
python seq_identity.py
```

```
   |    0    1    2    3
---+--------------------
0  |  100
1  |   45  100
2  |   20   25  100
3  |   15   30   25  100
```