Computergrundlagen 2025

Blatt 6: LaTeX and Python

- Abgabetermin für die Lösungen: **30.11.2025, 20 Uhr/ für Montagsgruppe: 28.11.2025, 12 Uhr**
- Bei Fragen wendet euch bitte an eure/n Tutor/in:
  - Mo 11:30: Stephan Haag: `st170833@stud.uni-stuttgart.de`
  - Di 09:45: Julian Hoßbach: `julian.hossbach@icp.uni-stuttgart.de`
  - Mi 14:00: Julian Peters: `julian.peters@icp.uni-stuttgart.de`
  - Do 09:45: Rebecca Stephan: `rebecca.stephan@icp.uni-stuttgart.de`
  - Fr 09:45: Jonas Höpker: `st182335@stud.uni-stuttgart.de`
- Die Übungsaufgaben sollen in der Regel in **Zweiergruppen** bearbeitet werden. Nur in **begründeten Ausnahmefällen** sind Dreiergruppen möglich.
- Die Abgabe der Übungsblätter erfolgt über Ilias.
- Mit Abgabe der Lösungen erklärt Ihr, dass Ihr die Lösung euren Mitstudierenden im Rahmen der Übungsbesprechung vorstellen könnt. Um dies zu überprüfen, muss mindestens zweimal von jedem Teilnehmenden vorgetragen werden. Wenn Ihr das nicht könnt, werden euch die Punkte für die entsprechenden Aufgaben wieder abgezogen.
- **Befehle, die nicht in der Vorlesung besprochen wurden, müssen gegebenenfalls recherchiert werden.**
- **Alle erstellen Skripte (`.py` and `.ipynb`) sowie ein mit markown oder Latex erstellter Report (`.pdf`) sind Teil der Abgabe**

  Tipp: Für Abgaben in Python kann der Editor https://jupyter.org/ hilfreich sein. Hier könnt ihr euren Code *interactive* ausführen sowie mit Markdown Zellen ausführlich kommentieren. Die `*.ipynb*` Dateien können ebenfalls als Lösungen abgegeben werden und sind leicht zu validieren.

LaTeX/ Markdown Report (1 Punkt)

Ab diesem Übungsblatt sollen alle Abgaben mit LaTeX oder Markdown erstellt werden und zu einer `.pdf` kompiliert werden. Insbesondere sollen die Lösungen der Aufgaben strukturiert dargestellt und Fragen ausführlich beantwortet werden. Mögliche Abbildungen oder Tabellen (in zukünftigen Abgaben) sollen eingebunden und entsprechend beschriftet werden. Auf diesem Blatt gibt es für das Erstellen der Abgabe mit LaTeX oder Markdownnoch einen Bonuspunkt, ab der nächsten Abgabe wird dies vorausgesetzt. Zusätzlich zur `.pdf` sollen alle erstellten Skripte Teil der Abgabe sein.

Listen (2 Punkte)

Gebt an, wie man in einer Python-Liste wie

```
a = [2, "d", 5, 8, 233, "dx", 54, "we", "g", ..., 72, 23, "g"]
```

auf folgende(s) Element(e) zugreifen kann: - Das vierte Element - Das vorletzte Element - Das dritte bis dritt letzte Element - Jedes zweite Element, beginnend ab dem vierten - Jedes dritte Element in umgekehrter Reihenfolge, beginnend ab dem vorletzten Wie kann man das siebte Element entfernen?

Datentypen und Ausdrücke (1 Punkt)

Sagt vorher, welches Ergebnis (Wert und Datentyp, oder eine Fehlermeldung) folgende Ausdrücke in Python produzieren. Begründet für Eure Abgabe, warum die Ausgabe einen bestimmten Typ hat/in bestimmter Weise ausgegeben wird. - 3 + 5 - 3 + 5.0 - “3” + “5” - “3” * 5 - 3//2 - 3/2 - int(2.71828) - round(2.71828) - “hallo” + “Welt”

Gerade / Ungerade (2 Punkte)

Ergänzt die folgende Funktion, welche angeben soll, ob eine gegebene natürliche Zahl eine gerade Zahl ist. Was passiert, wenn du der Funktion keine positive, `int` Zahl übergibst? > Der folgenden Codeblock enthält einen `docstring` im numpy Format. Dieser wird unterhalb einer Funktion geschrieben und beginnt / endet mit `"""`. Dieser stellt die häufigste Art der Code Dokumentation in Python dar. Ein Docstring ist dabei nicht zu verwechseln mit einem Kommentar, welcher innerhalb des Codes mit einem `#` beginnt.

```
def is_even(number):
  """Check, if a given number is even.
  Parameters
  ----------
  number: int
      the number to be checked.

  Raises
  ------
  AssertionError: if the input is not 'int' and not positiv.

  Returns
  -------
  bool:
     'True' if the number is even.
  """
  assert isinstance(number, int) and number >= 1, "Die gegebene Zahl ist keine positive,\
natürliche Zahl."
  ...  # TODO: Überprüfe ob die Zahl gerade ist.

assert not is_even(9)
assert is_even(10)
```

Summation (2 Punkte)

Schreibt eine Funktion, die mittels einer Schleife die Quadratzahlen von 1^2 bis zu einer gegebenen Zahl N^2 aufsummiert. In Formelschreibweise heißt dies:

$$S(N) := \sum_{k=1}^{N} k^2 = 1^2 + 2^2 + \cdots + N^2.$$

Hinweis: testet immer euren Code (auch für die anderen Aufgaben!). In diesem Fall kann man testen, indem man mit der in der Mathematik bekannten Kurzform $S(N) = \frac{N(N+1)(2N+1)}{6}$ vergleicht.

```python
def get_summation_sq(number):
    """Summiere Quardatzahlen bis N.

    Parameters
    ----------
    number: int
        maximale Zahl N.

    Returns
    -------
    result: int
       aufsummierte Quadratzahlen bis N.
    """
    assert isinstance(number, int) and number >= 1, "Die gegebene Zahl ist keine positive,\
natürliche Zahl."
    result = 0
    ...  # TODO
    return result

assert get_summation_sq(...) == ...
```

Fakultät (2 Punkte)

Schreibt eine Funktion, die die Fakultät von n berechnet. Für alle natürlichen Zahlen n ist die Fakultät

$$n! := \prod_{k=1}^{n} k = 1 \cdot 2 \cdot ... \cdot n$$

als das Produkt der natürlichen Zahlen von 1 bis n definiert. Weiterhin gilt $0! = 1$.

```python
def fak(n):
    """Berechne das Produkt aller natürlichen Zahlen von 1 bis n.

    Parameters
    ----------
    n: int

    Returns
    -------

    result: int
       Produkt aller natürlichen Zahlen von 1 bis n.
    """

    assert isinstance(n, int) and n >= 1, "Die gegebene Zahl ist keine positive,\
natürliche Zahl."
    result = 0

    ...  # TODO
```

```
  return result

assert fak(...) == ...
```

Nutzt diese Funktion, um die Zahl e mittels der Potenzreihe auszurechnen. Die Exponentialfunktion ist definiert als

$$e^x = \sum_{k=1}^{\infty} \frac{x^k}{k!}.$$