Name: Vorname:

Geburtsdatum: Matrikelnummer:

Platz:

Nachklausur
Informatische Werkzeuge in den Geistes- und Sozialwissenschaften 1

13. April 2023

	To be used for grading, do not write here										
prob.	1.1	1.2	2.1	2.2	3.1	3.2	3.3	4.1	4.2	Sum	grade
total	5	5	6	6	8	6	5	7	12	60	
reached											

Organisatorisches

Bitte lesen die folgenden Anweisungen genau und bestätigen Sie diese mit Ihrer Unterschrift.

1. Bitte legen Sie Ihren Studierendenausweis und einen Lichtbildausweis zur Personenkontrolle bereit!
2. Sie können 60 Punkte erreichen, wenn Sie alle Aufgaben vollständig lösen. Allerdings zählen 55 Punkte bereits als volle Punktzahl, d.h. 5 Punkte sind Bonuspunkte.
3. Es sind keine Hilfsmittel erlaubt außer eines handgeschriebenen "Spickzettels" von 1 Seite A4 einseitig.
4. Die Bearbeitungszeit beträgt genau 60 min.
5. Schreiben Sie die Lösungen direkt auf die ausgeteilten Aufgabenblätter. Andere Blätter werden nicht bewertet.
6. Wenn Sie die Prüfung aus gesundheitlichen Gründen abbrechen müssen, so muss Ihre Prüfungsunfähigkeit durch eine Untersuchung in der Universitätsklinik nachgewiesen werden. Melden Sie sich in jedem Fall bei der Aufsicht und lassen Sie sich das entsprechende Formular aushändigen.
7. Überprüfen Sie Ihr Exemplar der Klausur auf Vollständigkeit (12 Seiten exklusive Deckblatt und Hinweise) und einwandfreies Druckbild! **Vergessen Sie nicht, auf dem Deckblatt die Angaben zur Person einzutragen und diese Erklärung zu unterschreiben!**

Erklärung: Durch meine Unterschrift bestätige ich den Empfang der vollständigen Klausurunterlagen und die Kenntnisnahme der obigen Informationen.

Erlangen, 13. April 2023 .....
(Unterschrift)

Bitte beachten Sie die folgenden Regeln, um keine Punkte zu verlieren:

- Wenn Sie eine Antwort auf einer anderen Seite fortsetzen, geben Sie bitte die Nummer der Aufgabe auf der neuen Seite mit an und verweisen Sie auf der alten Seite auf die neue.
- Begründen Sie Ihre Aussagen, wenn angebracht (wir würden gerne Teilpunkte für unvollständige Antworten geben). Wenn nicht explizit darum gebeten, antworten Sie möglichst nicht einfach mit „Ja"', „Nein"' oder „42"'.

1 Grundlagen und Verständnis

Aufgabe 1.1 (Stellenwertsysteme) 5 Pkt

Menschen benutzen verschiedene Zahlensysteme. Manche davon sind *Stellenwertsysteme* („Positional Number Systems“). Erklären sie kurz (!), was ein Stellenwertsystem ausmacht. Nennen Sie außerdem zwei Zahlensysteme, die Stellenwertsysteme sind und ein Zahlensystem, welches keins ist.

Aufgabe 1.2 (Syntax-Hightlighting) 5 Pkt

In vielen Entwicklungsumgebungen (z.B. in JupyterLab) werden bestimmte Teile von Quellcode farblich hervorgehoben (z.B. in python Schlüsselwörter wie `return` oder `break`).

Macht dies python zu einer *Plain Text*- oder zu einer *Markup*-Sprache? Begründen Sie Ihre Antwort kurz.

2 Reguläre Ausdrücke

Aufgabe 2.1 (Regulärer Ausdruck für die UDK) 6 Pkt

Die *Universelle Dezimalklassifikation* (kurz: UDK) ist ein System, das weltweit in Bibliotheken zur Klassifizierung menschlichen Wissens benutzt wird. Eine reguläre UDK-Nummer besteht aus Gruppen von 1-3 *Dezimal*ziffern, getrennt durch Punkte zur Leserlichkeit. Je mehr Ziffern, desto präziser die Klassifizierung (so ist "1" z.B. die UDK-Nummer für den gesamten Bereich Philosophie und Psychologie, der Unterbereich Metaphysik hingegen ist "111.5") .

Hier sind ein paar weitere valide Beispiele:

```
599.312.3
316.832
629.7.025.33
37.091.322.7
404
```

Geben Sie einen regulären Ausdruck an, der auf **UDK**-Ausdrücke wie oben beschrieben matcht. Falls Sie Gruppierungen verwenden, ist es egal ob diese *capturing* oder *non-capturing* sind.

Aufgabe 2.2 (Regulärer Ausdruck für die UDK 2) 6 Pkt

Neben den regulären UDK-Nummern wie in Aufgabe 2.1 sieht der Standard etliche Konstrukte für weitere Informationen vor. Wir beschäftigen uns hier mit dem Konstrukt für Dokumententypen.

Um eine bestimmte Sorte von Dokumenten zu einem Thema zu benennen, wird der UDK-Nummer des Themas eine so genannte „Klammer“ angehängt, z.B. `(094)` für Gesetzestexte und juristisches. Diese Klammer enthält (für unsere Zwecke) 2-3 Ziffern, *immer* beginnend mit einer 0.

Hier sind ein paar valide Beispiele:

```
599.312.3(046)
316.832(094)
37.091.322.7(07)
```

Ergänzen Sie Ihren regulären Ausdruck aus Aufgabe 2.1 so, dass nun auch UDK-Ausdrücke mit Dokumententyp wie oben beschrieben matcht. Die Klammer für den Dokumententyp soll dabei optional (!) sein. Falls Sie Gruppierungen verwenden, ist es egal ob diese *capturing* oder *non-capturing* sind.

Hint: Sie werden in dieser Aufgabe nur für die neu dazugekommenen Teile des regulären Ausdrucks bewertet. Sollten Sie die vorherige Aufgabe nicht bearbeitet haben, dürfen Sie einen Platzhalter benutzen.

---

3 Digitale Dokumente

Aufgabe 3.1 (Mentales CSS) 8 Pkt

Gegeben ist der Quelltext der folgenden *HTML* Seite:

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      body { background-color: lightblue; }

      h1 {
        text-align: center;
      }

      span { background-color: yellow; }

      div {
        font-weight: bold;
        color: #FF0000;
      }
    </style>
  </head>
  <body>
    <h1 style="font-size:␣5pt;">IWGS</h1>

    <span style="background-color:␣magenta;">
      Lorem ipsum...

      <form>
        <input type="submit" value="Useless␣button!">
      </form>
    </span>
  </body>
</html>
```

Beschreiben Sie, welche Elemente auf der fertig dargestellten *HTML* Seite zu sehen sind, wie sie angeordnet sind und welche Farbe sie haben.

This page was intentionally left blank for extra space

Aufgabe 3.2 (DOM) 6 Pkt

Erklären Sie den Begriff des DOM im Kontext von digitalen Dokumenten und welche Rolle das DOM (und darauf aufbauende Programmiersprachen wie Javascript) für interaktive Webseiten spielt.

Skizzieren Sie beispielhaft (kein Quelltext notwendig, Beschreibung genügt), wie Javascript mit dem DOM in Antwort auf ein von der Nutzer*In ausgelöstes Ereignis interagiert.

Aufgabe 3.3 (XPath) 5 Pkt

Gegeben ist folgender Ausschnitt aus einer XML-Datei zur Katalogisierung von CDs:

```
<?xml version="1.0" encoding="UTF-8"?>
<CATALOG>
  <CD>
    <TITLE>Greatest Hits</TITLE>
    <ARTIST>Dolly Parton</ARTIST>
    <COUNTRY>USA</COUNTRY>
    <COMPANY>RCA</COMPANY>
    <PRICE>9.90</PRICE>
    <YEAR>1982</YEAR>
  </CD>
  <CD>
    <TITLE>The very best of</TITLE>
    <ARTIST>Cat Stevens</ARTIST>
    <COUNTRY>UK</COUNTRY>
    <COMPANY>Island</COMPANY>
    <PRICE>8.90</PRICE>
    <YEAR>1990</YEAR>
  </CD>
  <CD>
    <TITLE>Maggie May</TITLE>
    <ARTIST>Rod Stewart</ARTIST>
    <COUNTRY>UK</COUNTRY>
    <COMPANY>Pickwick</COMPANY>
    <PRICE>8.50</PRICE>
    <YEAR>1990</YEAR>
  </CD>
  <CD>
    <TITLE>The dock of the bay</TITLE>
    <ARTIST>Otis Redding</ARTIST>
    <COUNTRY>USA</COUNTRY>
    <COMPANY>Stax Records</COMPANY>
    <PRICE>7.90</PRICE>
    <YEAR>1968</YEAR>
  </CD>

  ...
</CATALOG>
```

Geben Sie einen `XPath`-Ausdruck an, der in der vollständigen Datei (mehr Einträge als hier gezeigt sind) alle Preise von CDs anwählt, die im Jahr 1990 erschienen sind.

This page was intentionally left blank for extra space

4 Programmieren in Python

Aufgabe 4.1 (Listen von Listen) 7 Pkt

Schreiben Sie eine python-Funktion `longest_string`, die aus einer *Liste von Listen von Strings* den längsten String findet und zurück gibt.

Beispiel: Angewendet auf die folgende Liste sollte Ihre Funktion `"elephant"` zurück geben:

```
[[sloth","elephant","cat"],[singing"],["banana","kiwi"]]
```

Diese Liste von Listen wird Ihrer Funktion als einziges Argument übergeben werden.

Hint: Sie können für diese Aufgabe annehmen, dass alle Längen der Strings in den Listen nur einmal vorkommen (d.h. es gibt keine zwei Strings, die gleich lang sind).

Aufgabe 4.2 (Bottle Route zum Testen regulärer Ausdrücke) 12 Pkt

Schreiben Sie eine Route `test_regexes()` für einen `bottle`-Server, die genau ein Argument (String) über die URL annimmt. Dieser String soll gegen eine Liste von regulären Ausdrücken getestet werden, welche erst aus Dateien ausgelesen werden müssen. Im Arbeitsverzeichnis liegen mehrere Dateien, deren Namen alle auf `.regex` enden. Der gesamte Inhalt einer solchen Datei stellt jeweils einen regulären Ausdruck dar.

Ihre Route soll testen, welche dieser regulären Ausdrücke auf den eingelesenen String aus der URL matchen. Am Ende soll eine Liste genau der Dateinamen zurück gegeben werden, die einen passenden Ausdruck enthalten.

Hint: Sie können für diese Aufgabe davon ausgehen, dass alle benötigten `import`-Statements bereits vorhanden sind und der Server korrekt gestartet wird. Sie müssen nur den Quelltext für die Route angeben.

This page was intentionally left blank for extra space

This page was intentionally left blank for extra space