FAU:IWGS1retake:WS2324:0

Name: Vorname:

Geburtsdatum: Matrikelnummer:

Platz:

Nachklausur
Informatische Werkzeuge in den Geistes- und Sozialwissenschaften 1

11. April 2024

	To be used for grading, do not write here											
prob.	1.1	1.2	1.3	1.4	2.1	2.2	3.1	3.2	4.1	4.2	Sum	grade
total	4	4	4	4	6	9	6	5	6	12	60	
reached												

Die „Lösungen“ der Aufgaben in diesem Dokument sollen den Studierenden als Anfangspunkt für die Beantwortung der Aufgaben dienen. Trotz aller Bemühungen kann es zu Unvollständigkeiten oder sogar Fehlern in den „Lösungen“ kommen. Da die Korrektur und Benotung der gestellten Aufgaben niemals lediglich auf einen „Vergleich mit der Musterlösung“ hinausläuft, ist dies auch nicht so schlimm. In jedem Fall sollten die Studierenden die Lösungen nachvollziehen und im Prinzip mittels des Lehrstoffs selbst verifizieren können.

Sollten Sie „Lösungen“ finden, die Sie nicht verstehen oder sogar für fehlerhaft halten diskutieren Sie diese am besten mit den Tutor*innen oder auf dem Kursforum und benachrichtigen Sie die Lehrenden; wir werden sie dann gegebenenfalls baldigst korrigieren.

1 Grundlagen und Verständnis

Aufgabe 1.1 (Dateipfade) 4 Punkte
4 Min

Was ist ein *Dateipfad*? Wie unterscheidet sich dieser von einem *file name*Dateinamen? Was bedeuten in diesem Kontext *relativ* und *absolut*?

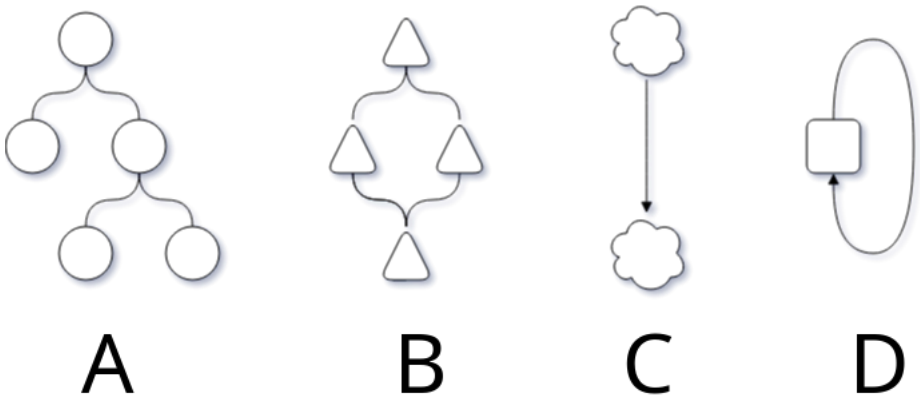
Lösung: Ein Dateipfad ist eine Zeichenfolge die eine Datei oder ein Verzeichnis in einem Dateisystem beschreibt. Ein Dateiname ist teil eines Dateipfades, enthält aber keine Informationen darüber, in welchen Verzeichnissen (oder auf welchem Laufwerk) sich die Datei befindet.

Ein absoluter Pfad beschreibt den vollständigen Pfad vom Ursprung des Dateisystems und ist eindeutig. Viele Betriebssysteme erlauben allerdings auch, relative Dateipfade zu einem anderen Dateipfad (wie dem aktuellen Verzeichnis) zu benutzen, bei dem Teile des Pfades ausgelassen werden können und vom Betriebssystem aus dem Kontext ergänzt werden.

Bewertung: Je 1 Punkt für korrekte Erklärung von Dateipfad, Dateiname, relativ und absolut.

Aufgabe 1.2 (Bäume) 4 Punkte
4 Min

Welche der folgenden Graphen zeigen korrekte Bäume im mathematischen Sinne? Für nicht-Bäume begründen Sie bitte Ihre Entscheidung kurz (!).



Lösung:

1. Baum
2. Kein Baum, Bäume erlauben es nicht, dass Elemente sich mehrere Vorfahren haben.
3. Baum
4. Kein Baum, Bäume erlauben es nicht, dass Elemente sich selbst zum Vor/Nachfahren haben.

Bewertung: Je ein Punkt pro korrektem Baum/Nicht Baum. Bei inkorrekten Begründungen jeweils einen halben Punkt Abzug.

AC1 Baum korrekt identifiziert
Punkte: +1.0
Rückmeldung: Korrekte Erkennung eines Baums

AC2 Nicht-Baum korrekt identifiziert
Punkte: +1.0
Rückmeldung: Korrekte Erkennung eines Nicht-Baums

AC3 Falsche Begründung
Punkte: -0.5
Rückmeldung: Fehler in Begründung

Aufgabe 1.3 (17 in verchiedenen Zahlensystemen) 4 Punkte
4 Min

In Stellenwert-Zahlensystemen (positional number systems) hängt der Wert einer Zahl mit mehr als einer Stelle davon ab, in welchem System wir uns genau befinden.

Geben Sie jeweils den Wert der Zahl 17_{10} in folgenden Zahlensystemen an:

1. Hexadezimalsystem (Basis 16)
2. Dezimalsystem (Basis 10)
3. Binärsystem (Basis 2)
4. Unärsystem (Basis 1)

Lösung:

1. $17_{10} = 11_{16}$
2. 17_{10} ist bereits im Dezimalsystem. :-)
3. $17_{10} = 10001_2$
4. $17_{10} = 11111111111111111_1$ (wir akzeptieren auch Antworten wie 00000000000000000 oder dergleichen.)

Bewertung: Ein Punkt pro korrekter Zahl.

AC1 Hexadec.Sys. korrekt.
Punkte: +1.0
Rückmeldung: Korrekte Umrechnung in das Hexadezimalsystem

AC2 Bin.Sys. korrekt.
Punkte: +1.0
Rückmeldung: Korrekte Umrechnung in das Binärsystem

AC3 Un.Sys. korrekt.
Punkte: +1.0
Rückmeldung: Korrekte Umrechnung in das Unärsystem

AC4 Dec.Sys. korrekt.
Punkte: +1.0
Rückmeldung: Korrekt erkannt, das Dezimal keine Umrechnung braucht

Aufgabe 1.4 (HTTP Requests) 4 Punkte
4 Min

Nennen Sie die beiden wichtigen strukturellen Bestandteile einer HTTP `POST`-Anfrage. Welchen dieser Teile hat eine HTTP `GET`-Anfrage in der Regel *nicht*? Angenommen, dass mit dieser `POST`-Anfrage eine Bilddatei auf einen Server hochgeladen werden soll, in welcher der beiden Komponenten, würden diese Daten übertragen?

Lösung: Die zwei Strukturkomponenten, nach denen hier gefragt wird, sind die *Header* und der *Body*.

Ein `GET`-Request hat im Vergleich dazu (in der Regel) keinen Body. Es wäre möglich, einen GET-Request mit Body zu versenden – der HTTP-Standard verbietet dies nicht explizit – allerdings würden die meisten Webserver diesen Body ignorieren.

Die Daten einer angehängten Bilddatei würden ebenfalls im *Body* übertragen.

Bewertung:

- Zwei Punkte für Benennung von Header und Body
- Einen Punkt für Angabe, GET-Requests keinen Body haben.
- Einen weiteren Punkt für die Angabe, dass die Daten im Body übertragen werden.

AC1 Header / Body identifiziert
Punkte: +2.0
Rückmeldung: Korrekte Nennung von Header und Body.

AC2 GET hat keinen Body
Punkte: +1.0
Rückmeldung: Korrekte Angabe dass der Body bei GET fehlt.

AC3 Body trägt die Daten.
Punkte: +1.0
Rückmeldung: Korrekte Angabe, dass die Daten im Body wären.

2 Reguläre Ausdrücke

Aufgabe 2.1 (Matchen Gegebener Regexes) 6 Punkte
6 Min

Geben Sie einen String an, der gegen alle folgende regulären Ausdrücke matcht.

- `.+apyb.+`
- `^[Cc].{7}$`
- `.a...a.a`
- `.*ara.*`

Lösung: Die beiden Strings, die hier möglich sind, sind `capybara` und `Capybara`.

Bewertung: 1.5 Punkte für jeden regulären Ausdruck, den der gegebene String erfüllt.

AC1 Regex1
Punkte: +1.5
Rückmeldung: Stringt matcht den ersten Regex

AC2 Regex2
Punkte: +1.5
Rückmeldung: Stringt matcht den zweiten Regex

AC3 Regex3
Punkte: +1.5
Rückmeldung: Stringt matcht den dritten Regex

AC4 Regex4
Punkte: +1.5
Rückmeldung: Stringt matcht den vierten Regex

Aufgabe 2.2 (RegEx für Go-Notation) 9 Punkte
9 Min

Das chinesische Strategiespiel *Go* wird (normalerweise) auf einem Brett mit 19x19 Feldern für die Figuren (gen. *Steine*) gespielt. Diese Felder sind auf der einen Seite mit den Buchstaben `A` bis `T` (der Buchstabe `I` wird ausgelassen), auf der anderen Seite mit den Zahlen 1 bis 19 gekennzeichnet.

Um den Verlauf eines Spiels später nachvollziehen zu können werden die einzelnen Züge (ähnlich wie auch beim Schach) notiert. Eine Notation besteht dabei aus einer Zahl (die theoretisch beliebig groß sein kann) für den Zug (Start bei 0) gefolgt von einem Punkt, einem beliebigen (!) Whitespace-Zeichen und einem gültigen Feldnamen. Geben Sie einen regulären Ausdruck für solche Notationen an.

Hinweis: Hier sind ein paar gültige Notationen für Züge in Go:

- `01. A1`
- `8. T19`
- `9999999. N10`

Die folgenden Notationen hingegen wären *nicht* gültig:

- `1. A100`
- `2. I10`
- `17. Z0`

Lösung: Hier ist eine Lösung (non-capturing Gruppen sind nicht verlangt):

```
\d+\.\s[ABCDEFGHJKLMNOPQRST](?:[1-9]|1[0-9])
```

---

3 Digitale Dokumente

Aufgabe 3.1 (XPath) 6 Punkte
6 Min

Beschreiben Sie für jeden der unten angegebenen XPaths, welche Elemente in einem XML-Dokument genau damit erreicht werden können.

1. `/city/library/book[@lang="de"][314]`

   Lösung: Dieser *XPath* beschreibt einen genauen Pfad durch ein Dokument. Gezielt wird auf den 314-ten `book`-Knoten mit dem Attribut `lang="de"` unter einem `library`-Knoten unter einem `city`-Knoten.

   ---

2. `//div[@id="hero"]//img`

   Lösung: Dieser Knoten zielt auf alle `img`-Knoten, die Nachfahren (Kinder, Kindeskinder, ...) eines `div`-Knotens mit der Id `"hero"` sind, egal wo im Dokument sich dieser `div`-Knoten befindet.

Aufgabe 3.2 (Mentales CSS) 5 Punkte
5 Min

Gegeben ist der Quelltext der folgenden HTML-Seite:

```
<!DOCTYPE html>
<html lang="en">
  <style>
    body { background-color: grey; }
    .foo { background-color: red; }
    .bar { background-color: black; }
    #baz { background-color: green; }
    .foo.bar { background-color: yellow; }
    .qux { background-color: pink; }
    .xuq { background-color: white; }
    .bar { background-color: blue; }
  </style>
  <body>
    <ul>
      <li class="foo">One</li>
      <li class="bar">Two</li>
      <li class="baz">Three</li>
      <li id="qux">Four</li>
      <li class="bar␣foo">Five</li>
    </ul>
  </body>
</html>
```

Beschreiben Sie, welche Elemente auf der fertig dargestellten Seite zu sehen sind und welche Hintergrundfarbe sie haben.

Lösung: Die Seite im Allgemeinen hat einen grauen Hintergrund. Die Farben der Listenelemente sind wie folgt:

1. Rot (zweite Regel)
2. Blau (letzte Regel überschreibt die dritte Regel)

3. Grau (geerbt von `body`, Regel 4 ist für Id `baz`, nicht die Klasse).
4. Grau (geerbt von `body`, Regel 6 ist für Klasse `qux`, nicht die Id).
5. Gelb (fünfte Regel, Reihenfolge der Klassen ist egal)

Die Regeln für `#baz`, `.qux` und `.xuq` finden gar keine Anwendung.

Bewertung: Es gibt einen Punkt pro Listenelement, das korrekt beschrieben wird. Wird die allgemeine Hintergrundfarbe nicht erwähnt, müssen zumindest die Hintergründe der Elemente Drei und Vier als Grau beschrieben werden.

AC1 Korrekte Farbe
Punkte: +1.0
Rückmeldung: Ein Element korrekt benannt

---

4 Programmieren in Python

Aufgabe 4.1 (Primzahlen) 6 Punkte
6 Min

Schreiben Sie eine python-Funktion namens `isPrime`, die einen Integer entgegen nimmt und einen Wahrheitswert (Boolean) zurück gibt, ob diese Zahl eine Primzahl ist.

Hinweis: Eine Primzahl ist nur durch sich selbst und 1 ganzzahlig teilbar und hat keine anderen Faktoren. Die Zahl 1 ist keine Primzahl. Die ersten Primzahlen sind also 2, 3, 5, 7, 11 ...

Lösung: Eine mögliche Lösung wäre zum Beispiel

```
def isPrime(n):
    """Takes an integer, returns if it is prime."""
    # 1 is not considered prime
    if n < 2:
        return False

    # If n has an integer divisior, it is not prime.
    for x in range(2, n):
        if n % x == 0:
            return False

    # Otherwise, it is indeed prime!
    return True
```

Bewertung:

AC1 Wrong Special case: 1
Punkte: -1.5
Rückmeldung: Der Spezialfall von 1 wird falsch behandelt (gibt 1 als prim an)

AC2 Wrong Special case: 2
Punkte: -1.5
Rückmeldung: Der “Spezialfall” von 2 wird falsch behandelt (gibt 2 als nicht prim an)

AC3 Wrong Special case: n
Punkte: -1.5
Rückmeldung: Der Spezialfall von n wird falsch behandelt (gibt n als prim an)

Aufgabe 4.2 (Bäume mit lxml) 12 Punkte
12 Min

Setzen Sie sich genau mit folgendem Schnipsel python-Code auseinander. Es wird die in der Vorlesung besprochene `lxml`-Bibliothek verwendet um iterativ (d.h. in mehreren Durchläufen) eine Baum-Datenstruktur aufzubauen, in der jeder Knoten in seinem `.text`-Attribut einen String enthält, der einer Zahl entspricht.

```
from lxml import etree

# Lege die Wurzel des Baumes an
root = etree.Element("node")
root.text = "0"
counter = 1

# Tue genau zwei Mal folgendes:
for _ in range(0,2):
    # Fuer "root" und alle seine Nachfahren tue folgendes:
    for element in root.iter():
        # Wenn der Knoten keine Kinder hat...
        if len(element) == 0:
            # ... fuege diesem Knoten (element) zwei Kinder hinzu ...
            fst_child = etree.SubElement(element, "node")
            snd_child = etree.SubElement(element, "node")
            # ... und gebe allen neuen Kindern den aktuellen counter als text.
            for child in element:
                child.text = str(counter)
                counter = counter + 1
```

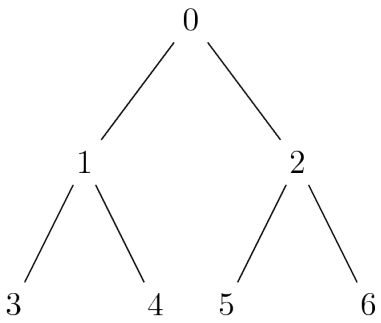
Wird der obige Code ausgeführt, so ist es möglich, den dort generierten Baum mit der Wurzel `root` wie folgt darzustellen (einmal in XML-Syntax und einmal als Grafik):

```
<node>
        0

        <node>
                1
                <node>3</node>
                <node>4</node>
        </node>

        <node>
                2
                <node>5</node>
                <node>6</node>
        </node>

</node>
```



Eine bildliche Darstellung des links beschriebenen Baumes.

Schreiben Sie nun eine python-Funktion `renameNodes`, die den Wurzelknoten eines solchen Baums als Argument bekommt. Die Funktion soll alle Knoten in dem Baum besuchen und ggf. deren `.text`-Attribut ersetzen. Dabei soll der neue Wert des Attributs `"prim!"` sein, wenn der ursprüngliche Wert des Attributs eine Primzahl darstellt. Andernfalls soll der Wert *nicht* ersetzt werden.

Hinweis: Sie können davon ausgehen, dass Ihnen eine korrekt implementierte Funktion `isPrime` wie aus der vorherigen Aufgabe zur Verfügung steht (auch wenn Sie diese Aufgabe nicht bearbeitet haben). Alle sonstigen Methoden und Funktionen, die Sie für die Beantwortung dieser Aufgabe benötigen, werden auch im obigen Codeschnippsel verwendet und werden deshalb hier nicht noch einmal aufgezählt.

Bedenken Sie, dass der Wert des `.text`-Attributs ein String ist, auch wenn er nur Ziffern enthält.

Lösung: Hier ist eine kommentierte rekursive Lösung dieser Aufgabe:

```
# Funktion definieren
def renameNodes(node):

    # Text in Zahl umwandeln.
    i = int(node.text)

    # Falls gerade...
    if isPrime(i):
        # Text-Attribut auf "prim!" setzen.
        node.text = "prim!"

    # Rekursiver Aufruf auf allen Kindern dieser node.
    for child in node:
        renameNodes(child)
```

Alternativ wäre auch zulässig gewesen, `.iter()` wie im Beispiel in der Aufgabe zu verwenden:

```
# Funktion definieren
def renameNodes(start):

    # Tue fuer alle Knoten unter start folgendes:
    for node in start.iter():

        # Text in Zahl umwandeln.
        i = int(node.text)

        # Falls prim...
        if isPrime(i):
            # Text-Attribut auf "prim!" setzen.
            node.text = "prim!"
```