Lösungen / Solutions: AINF1166, Informatik I (V+Ü) (Informatics I)

Questions 1

Gefragt / Asked: 2

Inferring Types / Typeninferenz

Punkte / Points: 1 Kprim

Frage / Question

EN: Given the following code, what types can variable 'b' take?

DE: Gegeben folgenden Code, welche Typen kann die Variable 'b' haben?

Snippet

```
a = input('Please enter your age')
b = a.isdigit() if not a else a
```

Antworten / Answers

- `bool`

  Correct

- `int`

  False

- `NoneType`

  False

- `str`

  Correct

Binary boolean expressions / Binärer boolsche Ausdrücke

Punkte / Points: 1 Kprim

Frage / Question

EN: Which of the values for 'a' and 'b' evaluate the following expression to True?

DE: Mit welchen Werten für 'a' und 'b' wird die folgende Expression als True evaluiert?

Snippet

```
bool(a or b)
```

Antworten / Answers

- `a = True, b = 0`

  Correct

- `a = False, b = 0`

  False

- `a = not -2, b = 0`

  False

- `a = 'Something', b = not None`

  Correct

Boolean expressions / Boolsche Ausdrücke

Punkte / Points: 1 Kprim

Frage / Question

EN: For which of the following values 'a' and 'b' does this expression evaluate as True?

DE: Für welche der folgenden Werte Werte 'a' und 'b' evaluiert diese Expression als True?

Snippet

```
bool(4 <= a < 40 and b)
```

Antworten / Answers

- `a = 16**0.5, b = 1`

  Correct

- `a = 'not (1 == 1)', b = -2`

  False

- `a = 39, b = (not not (1 == 1))`

  Correct

- `a = 4, b = True or not True`

  Correct

Compound Types / Compound Types

Punkte / Points: 1 Kprim

Frage / Question

EN: which of the following are compound types?

DE: Welche der folgenden Typen sind Compound Types?

Antworten / Answers

- `str`

  Correct

- `NoneType`

  False

- `int`

False

- `complex`

  Correct

List and tuple operations / Listen- und Tupel-Operationen

Punkte / Points: 1 Kprim

Frage / Question

EN: Which of the following are valid expression for the given 'lst' or 'tpl' variables?

DE: Welche der folgenden Expressions sind für die gegebenen Variablen 'lst' und 'tpl' gültig?

Snippet

```
lst = ['a', 1, var1]
tpl = (2,)
```

Antworten / Answers

- `x = del(lst[3])`

  False

- `x = del(lst[0])`

  False

- `lst.append(tpl[0])`

  Correct

- `tpl[0] = lst[0]`

  False

Programming 1

Gefragt / Asked: 2

all_falsy

Punkte / Points: 7

Frage / Question

DEUTSCH

Implementieren sie eine Funktion namens `all_falsy`, die eine Liste von beliebigen Werten `values` annimmt. Die Funktion soll bestimmen, ob alle Werte in `values` als `False` evaluieren. Wenn ja, soll die Funktion `True` zurückgeben, aber falls irgendein Wert als `True` evaluiert, soll `False` zurückgegeben werden. Für eine leere Liste soll `True` zurückgegeben werden.

Sie können davon ausgehen, dass die Funktion nur mit gültigen Parametern aufgerufen wird.

ENGLISH

Implement a function `all_falsy`, which takes a list of arbitrary values as a parameter `values`. The function should determine if all values in `values` evaluate as `False`. If yes, it should return `True`, but if any of the values evaluate as `True`, it should return `False`. For an empty list, the function should return `True`.

You can assume that the function is only called with valid parameters.

Fügen Sie ihre Lösung in folgendes Textfeld ein / Paste your solution into the following text field.

```
def all_falsy(values):
    pass

#assert all_falsy([False, "", 0, []])
#assert (not all_falsy([False, "no", 0.1]))
#assert all_falsy([[]])
```

Lösung / Solution

```
#!/usr/bin/env python3

def all_falsy(values):
    # the traditional way:
    for v in values:
        if v:
            return False
    return True
```

count_characters

Punkte / Points: 7

Frage / Question

DEUTSCH

Implementieren Sie eine Funktion `count_characters`, welche einen einzelnen String `text` als einzigen Parameter annimmt. Die Funktion soll ein Dictionary in der Form `{char: count}` zurückgeben, welches angibt, wie oft jeder Buchstabe in `text` vorkommt. Beachten Sie die Assertions als Beispiele für die Anwendung von `count_characters`.

ENGLISH

Implement a function `count_characters`, which accepts a single String `text` as the only paramter. The function should return a dictionary in the form `{char: count}` which represents how many times each character appears in `text`. Consider the assertions as examples for how `count_characters` can be used.

Fügen Sie ihre Lösung in folgendes Textfeld ein / Paste your solution into the following text field.

```
def count_characters(text):
    pass

# assert(count_characters("hello") == {"h": 1, "e": 1, "l": 2, "o": 1})
# assert(count_characters("Прогресс") == {"П": 1, "р": 2, "о": 1, "г": 1, "е": 1, "с": 2})
# assert(count_characters("<3") == {"<": 1, "3": 1})
# assert(count_characters("") == {})
```

Lösung / Solution

```
#!/usr/bin/env python3

def count_characters(text):
    res = {}
    for c in text:
        if c not in res:
            res[c] = 0
        res[c] += 1
    return res
```

duplicates

Punkte / Points: 7

Frage / Question

DEUTSCH

Implementieren Sie eine Funktion `duplicates_of`, welche zwei Listen `lst` und `xs` annimmt und einen Integer zurückgibt, welcher angibt, wie of die Elemente in `xs` in `lst`. vorkommen. Beachten Sie, dass sowohl `lst` als auch `xs` Werte von verschiedenen Typen enthalten können, und dass beide Listen Duplikate enthalten können. Es ist nicht nötig, Duplikate zu entfernen oder besonders zu behandeln. Beachten Sie die Assertions als Beispiele für die Anwendung von `duplicates_of`.

ENGLISH

Implement a function `duplicates_of`, which accepts two lists `lst` and `xs` and returns an integer representing how many times the items in `xs` appear in `lst`. Please note both `lst` and `xs` can contains values of various types, and that both lists can contain duplicates. It is not necessary to remove duplicates or treat them specially. Consider the assertions as examples for how `duplicates_of`. can be used.

Fügen Sie ihre Lösung in folgendes Textfeld ein / Paste your solution into the following text field.

```
def duplicates_of(lst, xs):
    pass

# assert(duplicates_of([], []) == 0)
# assert(duplicates_of([1], [1]) == 1)
# assert(duplicates_of([True, True], [True]) == 2)
# assert(duplicates_of([1, 1, 'hello', 3], [1, 'hello', 1]) == 5)
```

Lösung / Solution

```
#!/usr/bin/env python3

def duplicates_of(lst, xs):
    # For this task we assume lst and xs are always a non-null list.
    sum = 0
    for n in lst:
        for x in xs:
            if x == n:
                sum += 1
    return sum
    # or using a list comprehenion:
    #return sum([x == n for x in xs for n in lst])
```

greetings

Punkte / Points: 7

Frage / Question

DEUTSCH

Implementieren Sie eine Funktion `greetings`, welche eine Liste von Strings (Namen von Personen) `names`, annimmt und einen String zurückgibt, um alle Personen in der Liste zu grüssen. Falls die Liste leer ist, soll `greetings`, den String `Hello?`. zurückgeben. In allen anderen Fällen soll die Funktion einen String zurückgeben, der mit `Hello`, beginnt, gefolgt von den Namen in `names`, kommagetrennt, mit Ausnahme der letzten zwei Namen, welche mit dem Wort `and`. getrennt werden sollen. In diesen Fällen soll der String mit einem Punkt enden. Sie

können annehmen, dass `names` immer eine Liste von Strings ist. Beachten Sie die Assertions als Beispiele für die Anwendung von `greetings`.

ENGLISH

Implement a function `greetings`, which accepts a list of strings `names`, of people's names, and returns a string greeting all of the people in the list. If the list of names is empty, then `greetings`, returns the string `Hello?`. If all other cases, the function returns a string starting with `Hello`, followed by the names in `names`, as given, all separated by a comma, with execption of the last two names which are separated by the word `and`. In this last scenario, the string computed always terminates with a dot. You can assume that `names` is always a list of strings. Consider the assert statements given below as examples for `greetings`.

Fügen Sie ihre Lösung in folgendes Textfeld ein / Paste your solution into the following text field.

```
def greetings(names):
    pass

# assert(greetings([]) == 'Hello?')
# assert(greetings(['Josh']) == 'Hello Josh.')
# assert(greetings(['Josh', 'Alice']) == 'Hello Josh and Alice.')
# assert(greetings(['Josh', 'Alice', 'Marie']) == 'Hello Josh, Alice and Marie.')
```

Lösung / Solution

```
#!/usr/bin/env python3

def greetings(names):
    # For this task we assume names is always a non-null list.
    g = 'Hello?'
    if names:
        g = 'Hello ' + ', '.join(names[:-1])
        g += ' and ' if len(names) > 1 else ''
        g += f"{names[-1]}."
    return g
```

is_sub_collection

Punkte / Points: 7

Frage / Question

DEUTSCH

Implementieren Sie eine Funktion `is_sub_collection`, welche zwei Listen `a` und `b` annimmt und `True` zurückgibt, wenn alle Elemente in `b` auch in `a`. vorkommen. Ansonsten soll die Funktion `False` zurückgeben. Ausserdem soll die Funktion immer `True` zurückgeben, falls `b` eine leere Liste ist, egal was der Inhalt von `a`. ist. Sie können annehmen, dass die Werte in `a` und `b` immer Listen sind. Beachten Sie die Assertions als Beispiele für die Anwendung von `is_sub_collection`.

ENGLISH

Implement a function `is_sub_collection`, which two lists of items `a`, and `b`, and returns `True` if all of the elements found in `b` can also be found in `a`. The function returns `False` otherwise. Furthermore, if `b` is an empty list, then the function always returns `True` regardless of the content of `a`. You can assume the values for `a` and `b` are always non-null lists. Consider the assert statements given below as examples for `is_sub_collection`.

Fügen Sie ihre Lösung in folgendes Textfeld ein / Paste your solution into the following text field.

```
def is_sub_collection(a, b):
    pass

# assert(is_sub_collection([], []))
# assert(not is_sub_collection([], [True]))
# assert(is_sub_collection([1, 2, 3, None], [None]))
```

Lösung / Solution

```python
#!/usr/bin/env python3

def is_sub_collection(a, b):
    # For this task we assume a b are always non-null lists.
    for i in b:
        if i not in a:
            return False
    return True
    # or using a list comprehension, note that all([]) == True:
    # return all([i in a for i in b])
```

no_unlucky

Punkte / Points: 7

Frage / Question

DEUTSCH

Implementieren sie eine Funktion namens `no_unlucky`, die zwei Parameter annimmt: Eine Liste von ganzen Zahlen `values`, und eine ganze Zahl `unlucky`.

Die Funktion soll die Summe jener Werte in `values` berechnen, die nicht durch `unlucky` teilbar sind. Falls `unlucky` `0` ist, sollen alle Werte in `values` aufaddiert werden. Die Funktion soll die resultierende Summe zurückgeben.

Sie können davon ausgehen, dass die Funktion nur mit gültigen Parametern aufgerufen wird.

ENGLISH

Implement a function `no_unlucky`, which takes two parameters: a list of integers `values`, and an integer `unlucky`.

The function should sum up all those values in `values` which are not divisible by `unlucky`. If `unlucky` is `0` all values in `values` should be summed up. Return the resulting sum from the function.

You can assume that the function is only called with valid parameters.

Fügen Sie ihre Lösung in folgendes Textfeld ein / Paste your solution into the following text field.

```python
def no_unlucky(values, unlucky):
    pass

#assert(no_unlucky([10, 24, 1], 13) == 35)
#assert(no_unlucky([13, 25], 13) == 25)
#assert(no_unlucky([13, 26], 13) == 0)
```

Lösung / Solution

```python
#!/usr/bin/env python3

def no_unlucky(values, unlucky):
    sum = 0
    for v in values:
        if unlucky == 0 or not v % unlucky == 0:
            sum += v
    return sum
```

shout

Punkte / Points: 7

Frage / Question

DEUTSCH

Implementieren sie eine Funktion namens `shout`, die einen String als parameter `text` annimmt. Die Funktion soll diesen String in Grossbuchstaben zurückgeben, und dabei ein oder mehrere Ausrufezeichen anhängen. Wenn `text` keine Ausrufezeichen enthält, soll ein Ausrufezeichen angehängt werden. Wenn `text` ein oder mehrere Ausrufezeichen enthält, soll dieselbe Anzahl Ausrufezeichen zusätzlich angehängt werden.

Sie können davon ausgehen, dass die Funktion nur mit gültigen Parametern aufgerufen wird.

ENGLISH

Implement a function `shout`, which takes a string as parameter `text`. The function should return this string in upper case with one or more exclamation marks added to the end. If `text` contains no exclamation marks, one exclamation mark should be appended. If `text` contains one or more exclamation marks, the same number of exclamation marks should be added to the end.

You can assume that the function is only called with valid parameters.

Fügen Sie ihre Lösung in folgendes Textfeld ein / Paste your solution into the following text field.

```
def shout(text):
    pass

#assert(shout("Hello, God") == "HELLO, GOD!")
#assert(shout("Hello, World!") == "HELLO, WORLD!!")
#assert(shout("Hello! World?") == "HELLO! WORLD?!")
```

Lösung / Solution

```
#!/usr/bin/env python3

def shout(text):
    exclams = text.count("!")
    if exclams == 0: exclams = 1
    return text.upper() + exclams * "!"
```

Questions 2

Gefragt / Asked: 2

Creating strings / Strings erstellen

Punkte / Points: 2 Kprim

Frage / Question

EN: Which of the following expressions produces the following string: 'Josh bought a 1 gallon bottle of milk.'

DE: Welche der folgenden Expressions generieren folgenden String: 'Josh bought a 1 gallon bottle of milk.'

Antworten / Answers

- `f'{'Josh'} bought a {1} gallon bottle of {'milk'}.'`

  False

- `'Josh bought' + 'a' + '1' + 'gallon bottle of milk.'`

  False

- `'%s bought a %d gallon %s of milk' % ('Josh', 1, 'bottle')`

  Correct

- `'Josh bought {} {} {} bottle of milk.'.format('a', 1, 'gallon')`

  Correct

Length of lists / Listenlänge

Punkte / Points: 2 Kprim

Frage / Question

EN: Which of the following lists has a length of 4?

DE: Welche der folgenden Listen haben die Länge 4?

Antworten / Answers

- `[name, 'bb', age]`

  False

- `list(range(x - 10, x, 3))`

  Correct

- `[[1, 1, 1], [1, 1], 1] + [1]`

  Correct

- `[[foo, ('b','a','r'), [baz,[42]]]]`

  False

List operations / Listenoperationen

Punkte / Points: 2 Kprim

Frage / Question

EN: Which of the following code snippets can be used to compute the length of any list?

DE: Welche der folgenden Codesnippets können benutzt werden um die Länge einer beliebigen Liste zu berechnen?

Antworten / Answers

- ```
  def func(l):
      return 2 + len(l[2:])
  ```

  False

- ```
  def func(l):
      return 1 + func(l[1:]) if l else 0
  ```

  Correct

- ```
  def func(l):
      def inner(l, n):
          return n + 1 if not l else inner(l[1:], n + 1)
      return inner(l, 0)
  ```

  False
  ```

- ```
  def func(l):
      x = 0
      for _ in l:
          x += 1
      return x
  ```

  Correct

Procedural and recursive / Prozedural und rekursiv

Punkte / Points: 2 Kprim

Frage / Question

EN: Which of the following Statements are correct?

DE: Welche der folgenden Aussagen sind korrekt?

Antworten / Answers

- EN: To solve the same problem, many different algorithms can exist.

  DE: Um dasselbe Problem zu lösen, kann es viele unterschiedliche Algorithmen geben.

  Correct

- EN: For any recursive algorithm there always exists a procedural routine that can carry out the same calculations.

  DE: Für jeden rekursiven Algorithmus gibt es einen prozeduralen Weg, der dieselben Berechnungen durchführen kann.

  Correct

- EN: Python allows for infinite recursive calls for any recursive function.

  DE: In Python kann man beliebig viele rekursive Aufrufe einer rekursiven Funktion machen.

  False

- EN: Python programs cannot consist solely of recursive functions.

  DE: Ein Pythonprogramm kann nicht ausschliesslich aus rekursiven Funktionen bestehen.

  False

Reading code / Code lesen

Punkte / Points: 2 Kprim

Frage / Question

EN: Which of the following are correct statements about the implementation given below?

DE: Welche der folgenden Aussagen treffen auf die gegebene Implementation zu?

Snippet

```
lst = []
lst.append((1, 'a', True))
lst.extend([(2,), (3, 'c', False)])
lst + [(4, 'd', True and False)]
x = 0
for (n, s, b) in lst:
   x += n
```
```

```
print(f'x = {x}')
```

Antworten / Answers

- EN: At the end of the script, the length of 'lst' is equal to 4.

  DE: Am ende des scripts ist die Länge von 'lst' 4

  False

- EN: Instead of constructing the list 'lst' in multiple steps, it could have been declared and assigned a value in a single statement.

  DE: Anstatt die Liste 'lst' in mehreren Schritten zu konstruieren, hätte man sie in einem einzigen statement deklarieren und ihr einen Wert zuweisen können.

  Correct

- EN: When run, the script prints: x = {x}.

  DE: Wenn man das script ausführt wird x = {x} geprintet.

  False

- EN: The for loop assumes the list carries tuples of the same size.

  DE: Der for loop geht davon aus, dass die Liste Tupel der gleichen grösse enthält.

  Correct

Programming 2

Gefragt / Asked: 1

encoder

Punkte / Points: 18

Frage / Question

DEUTSCH

Implementieren Sie eine Funktion `is_valid_encoding`, welche drei Parameter annimmt: zwei Strings `a` und `b`, sowie ein dictionary `mapping`, welches chars als Keys und beliebig lange Strings als Werte enthält. Die Funktion soll `True` zurückgeben, wenn die Zeichensequenz in `a` mithilfe der Zuweisungen in `mappings` in den String in `b` umgewandelt werden kann. Ansonsten soll die Funktion `False` zurückgeben. Falls die ersten zwei Parameter beide `None` oder beides leere Strings sind, soll ebenfalls `True` zurückgegeben werden. Beachten Sie die Assertions als Beispiele für die Anwendung von `can_derive`.

ENGLISH

Implement a function `is_valid_encoding`, which takes three parameters: two strings `a` and `b`, and a dictionary `mapping` which has characters as keys and arbitrarily long strings as values. The function should return `True` if it is possible to translate the string in `a` to the string in `b` `b`. using the mappings in `mapping` If the first two parameters are both `None` or both empty strings, the function should also return `True`. Consider the assert statements given below as examples for `can_derive`.

Fügen Sie ihre Lösung in folgendes Textfeld ein / Paste your solution into the following text field.

```
def is_valid_encoding(a, b, mapping):
    pass
```

```
# assert(is_valid_encoding(None, None, {}))
# assert(is_valid_encoding(None, None, {'i': "don't care"}))
# assert(not is_valid_encoding(None, '', {}))
# assert(not is_valid_encoding('', None, {}))
# assert(is_valid_encoding('cat', 'dog', {'c': 'd', 'a': 'o', 't': 'g'}))
# assert(is_valid_encoding('a', 'archer', {'a': 'archer'}))
# assert(is_valid_encoding('no', 'yes', {'n': 'y', 'o': 'es', 'z': 'z'}))
```

Lösung / Solution

```
#!/usr/bin/env python3

def is_valid_encoding(a, b, mapping):
    # If they're both exaclty None or empty, true.
    if a == b and (a == None or a == ''):
        return True
    # If one is None and the other empty, false.
    elif (a == None and b == '') or (a == '' and b == None):
        return False
    # Otherwise check bindings.
    else:
        tmp = ''.join([mapping.get(i, i) for i in a])
        return tmp == b
```

min_max_if_num

Punkte / Points: 18

Frage / Question

DEUTSCH

Implementieren Sie eine Funktion `min_max_of`, welche eine Liste von beliebigen Werten `values`, annimmt und ein Tupel zurückgibt, welches den maximalen und minimalen numerischen Wert enthält, sofern solche Werte in der Liste vorkommen. Falls keine entsprechenden numerischen Werte vorkommen, soll ein Tupel zurückgegeben werden, wo beide Werte `None` sind. Falls die minimalen und maximalen Werte identisch sind, soll nur der Minimalwert im Tupel vorkommen und das zweite Tupelelement soll `None` sein. Beachten Sie die Assertions als Beispiele für die Anwendung von `min_max_of`.

ENGLISH

Implement a function `min_max_of`, which takes a list of arbitrary values `values`, and returns the tuple of the minimum and maximum numerical values in the list, if any exist. If the minimum and maximum values are equal, only the minimum value should appear in the tuple, while the second tuple element should be `None` Consider the assert statements given below as examples for `min_max_of`.

Fügen Sie ihre Lösung in folgendes Textfeld ein / Paste your solution into the following text field.

```
def min_max_of(values):
    pass

# assert(min_max_of(None) == (None, None))
# assert(min_max_of([]) == (None, None))
# assert(min_max_of([True]) == (None, None))
# assert(min_max_of([0, 0.0]) == (0, None))
# assert(min_max_of([0, 0.0]) == (0.0, None))
# assert(min_max_of([-1, 0, 1]) == (-1, 1))
# assert(min_max_of(['a', 'A', 1]) == (1, None))
```

Lösung / Solution

```
#!/usr/bin/env python3

def min_max_of(values):
    # Initialise return values to base assumption.
    l, u = None, None
    # Ensure list is not null.
    for n in values if values else []:
        # Filter elements of supported number types.
```

```
        if isinstance(n, (int, float)) and not isinstance(n, bool):
            # Update lower and upper values.
            l = n if not l or n < l else l
            u = n if not u or n > u else u

    # Equal min and max: report min only.
    u = None if l == u else u
    return l, u
```

multiples_of

Punkte / Points: 18

Frage / Question

DEUTSCH

Implementieren Sie eine Funktion `multiples_of`, welche zwei Zahlen als Parameter annimmt: eine beliebige Zahl `n` und eine weitere Zahl `count`. Die Funktion soll `n` mit jedem Integer von `1` bis `count` multiplizieren und ein Dictionary zurückgeben, wo jeder Key ein Multiplikator und jeder zugehöriger Value `n` mal dem Multiplikator entspricht. Sehen Sie sich die Assertions für konkrete Beispiele an.

Wenn `count` null ist, soll die Funktion ein leeres Dictionary zurückgeben.

Wenn `count` kleiner als null oder kein Integer ist, soll die Funktion `False` zurückgeben.

Sie können davon ausgehen, dass die Funktion nur mit zwei Zahlen aufgerufen wird.

ENGLISH

Implement a function `multiples_of`, which takes two numbers as parameters: an arbitrary number `n` and another number `count`. The function should multiply `n` with each integer number from `1` to `count` and return a dictionary where each key is the multipier and each corresponding value is `n` times the multiplier. See the assertions below for concrete examples.

If `count` is zero, the function should return an empty dictionary.

If `count` is smaller than zero or not an integer, the function should return `False`.

You can assume that the function is only called with two numbers as parameters.

Fügen Sie ihre Lösung in folgendes Textfeld ein / Paste your solution into the following text field.

```
def multiples_of(n, count):
    pass

#assert(multiples_of(3, 4)  == {1: 3, 2: 6, 3: 9, 4: 12})
#assert(multiples_of(2, 0)  == {})
#assert(multiples_of(0, 3)  == {1: 0, 2: 0, 3: 0})
#assert(multiples_of(-2, 3) == {1: -2, 2: -4, 3: -6})
```

Lösung / Solution

```
#!/usr/bin/env python3

def multiples_of(n, count):
    if not isinstance(count, int) or count < 0:
        return False
    # manually:
    res = {}
    for i in range(1, count+1):
        res[i] = i*n
    return res
    # using a dict comprehension:
    return {i: n*i for i in range(1,count+1)}
```

recursive_format

Punkte / Points: 18

Frage / Question

DEUTSCH

Implementieren Sie eine Funktion `recursive_format`, welche eine Liste `tree` als einzigen Parameter annimmt. `tree` kann Strings enthalten, aber auch eingebettete Listen, die wiederum Strings und tiefer eingebettete Listen enthalten. Die Funktion soll einen einzigen String basierend auf `tree` wie folgt zurückgeben:

- Strings in `tree` sollen aneinandergereiht werden, separiert durch einzelne Leerzeichen.
- Eingebettete Listen sollen mittels einem rekursiven Aufruf von `recursive_format` genau gleich verarbeitet werden, wie `tree` und der resultierende Rückgabewert soll dem Ergebnis in Klammern hinzugefügt werden.

Beachten Sie die Assertions als Beispiele für die Anwendung von `recursive_format`.

ENGLISH

Implement a function `recursive_format`, which takes a list `tree` as the only parameter. `tree` can contain strings but also nested lists, which in turn again can contain both strings and more deeply nested lists. The function should return a single string constructed from `tree` as follows:

- Strings in `tree` should be strung together separated by a single space.
- Nested lists should be processed by a recursive call to `recursive_format` exactly the same way as `tree` itself and the resulting return value should be added to the result, enclosed by parantheses.

Consider the assert statements given below as examples for `recursive_format`.

Fügen Sie ihre Lösung in folgendes Textfeld ein / Paste your solution into the following text field.

```
def recursive_format(tree):
    pass

# assert(recursive_format([]) == "")
# assert(recursive_format([[]]) == "()")
# assert(recursive_format(["Hi", "there", ["Jack", "and", "Bobby"], "!"])
#                       == "Hi there (Jack and Bobby) !")
# assert(recursive_format(["What", "is", ["going", ["on"], "here?"]])
#                       == "What is (going (on) here?)")
```

Lösung / Solution

```
#!/usr/bin/env python3

def recursive_format(tree):
    res = ""
    elems = []
    for node in tree:
        if isinstance(node, str):
            elems.append(node)
        else:
            elems.append(f"({recursive_format(node)})")
    return " ".join(elems)
```

string_derivation

Punkte / Points: 18

Frage / Question

DEUTSCH

Implementieren Sie eine Funktion `can_derive`, welche einen String `s` und ein Dictionary `chars`.

annimmt, wo die Keys vom Typ char und die Werte vom Typ int sind. Das Dictionary beschreibt, wie oft jeder Buchstabe benutzt werden kann. Somit soll die Funktion einen bool zurückgeben der angibt, ob genügend Buchstaben in `chars` vorhanden sind, um den String in `s` zu konstruieren. Wenn ja, soll `True` zurückgegeben werden, ansonsten `False`. Beachten Sie die Assertions als Beispiele für die Anwendung von `can_derive`.

ENGLISH

Implement a function `can_derive`, which takes a string `s`, and a dictionary `chars` where the keys are of type char and the values are of type int. The dictionary describes, how many times each character can be used. Hence, the function should return a bool, stating whether there are sufficient characters to construct the given string `s`. If yes, the function should return `True`, `False` otherwise. Consider the assert statements given below as examples for `can_derive`.

Fügen Sie ihre Lösung in folgendes Textfeld ein / Paste your solution into the following text field.

```
def can_derive(s, chars):
    pass

# assert(can_derive('', {}))
# assert(not can_derive('hello', {}))
# assert(can_derive('aabb', {'a': 2, 'b': 2}))
```

Lösung / Solution

```
#!/usr/bin/env python3

def can_derive(s, chars):
    check = True
    # Keep track of chars already visited.
    checked = []
    # For each char not already visited.
    for c in s:
        if not c in checked:
            # Mark as visited.
            checked.append(c)
            # Count how many times this is found in the string.
            n_occs = sum([c == i for i in s])
            # Check we have enough of such chars in the dictionary.
            if n_occs > chars.get(c, 0):
                check = False
                break
    return check
```

Questions 3

Gefragt / Asked: 2

Scopes / Scopes

Punkte / Points: 2 Kprim

Frage / Question

EN: Which of the following statements are true about the Python script given below?

DE: Welche der folgenden Aussagen treffen auf dieses Python script zu?

Snippet

```
lst = [[2]]

def func1(lst):
   lst = lst + [1, 2]
   print(lst)
```

```
def func2(lst):
   lst.append([1, 2])
   print(lst)
```

Antworten / Answers

- EN: Invoking 'func1' on 'lst' (declared in line 1) at the end of the script, would result in 'lst' (declared in line 1) being of length 3.

  DE: Wenn man 'func1' mit 'lst' (deklariert auf Zeile 1) aufruft, wird 'lst' (deklariert auf Zeile 1 am Ende des Scripts die Länge 3 haben.

  False

- EN: Invoking 'func2' on 'lst' (declared in line 1) at the end of the script, would result in 'lst' (declared in line 1) being of size 2.

  DE: Wenn man 'func2' mit 'lst' (deklariert auf Zeile 1) aufruft, wird 'lst' (deklariert auf Zeile 1 am Ende des Scripts die Länge 2 haben.

  Correct

- EN: Invoking 'func1' on '[]' at the end of the script, would result in 'lst' (declared in line 1) being of size 2.

  DE: Wenn man 'func1' mit '[]' aufruft, wird 'lst' (deklariert auf Zeile 1) am Ende des Scripts die Länge 2 haben.

  False

- EN: Invoking 'func2' on '[]' at the end of the script, would result in 'lst' (declared in line 1) being of size 1.

  DE: Wenn man 'func2' mit '[]' aufruft, wird 'lst' (deklariert auf Zeile 1) am Ende des Scripts die Länge 1 haben.

  Correct

Scopes and assignments / Scopes und Zuweisungen

Punkte / Points: 2 Kprim

Frage / Question

EN: Which of the following statements are true about the python script given below?

DE: Welche der folgenden Aussage treffen auf dieses Pythonscript zu?

Snippet

```
a = (1, [])

def func(a, lst):
   a[0] = lst + lst
   a += [(3, 4), lst]
   return a
```

Antworten / Answers

- EN: Execution would crash if 'func(a, a)' is called at the end of the script.

  DE: Das Script stürzt ab wenn man 'func(a, a)' am Ende des Scripts ausführen würde.

  Correct

- EN: Invoking 'func([a], a)' at the end of the script would return a list of length 3.

  DE: Ein Aufruf von 'func([a], a)' am Ende des Scripts würde eine Liste der Länge 3 zurückgeben.

  Correct

- EN: Execution would crash if 'func(a, True) is called at the end of the script.

  DE: Das Script stürzt ab wenn man 'func(a, True) am Ende des Scripts ausführen würde.

  False

- EN: Invoking 'func((a), (a,))' at the end of the script would return a list of length 3.

  DE: Ein Aufruf von 'func((a), (a,))' am Ende des Scripts würde eine Liste der Länge 3 zurückgeben.

  False

Scopes and recursion / Scopes und Rekursion

Punkte / Points: 2 Kprim

Frage / Question

EN: Which of the following statements are true about the python script given below?

DE: Welche der folgenden Aussagen treffen auf dieses Pythonscript zu?

Snippet

```
a = 1
b = 0
c = []
def func(a):
   if a < 0:
       b = -1
   elif a == 0:
       c.append(-2)
   else:
       a = func(a - 1) - 3
   return a
func(x)
```

Antworten / Answers

- EN: Assigning the value '-9' to 'x' and running the script above leads to variable 'b' (declared in line 2) carrying the value '0'.

  DE: Wenn man 'x' den Wert '-9' zuweist und das Script ausführt, wird b (deklariert auf Zeile 2) den Wert '0' haben.

  Correct

- EN: Assigning the value '3' to 'x' and running the script above leads to variable 'a' (declared in line 1) carrying the value '-9'.

  DE: Wenn man 'x' den Wert '3' zuweist und das Script ausführt, wird a (deklariert auf Zeile 1) den Wert '-9' haben.

  False

- EN: Assigning the value '2' to 'x' and running the script above leads to variable 'a' (declared in line 1) carrying the value '1'.

  DE: Wenn man 'x' den Wert '2' zuweist und das Script ausführt, wird a (deklariert auf Zeile 1) den Wert '1' haben.

  Correct

- EN: The values assigned to the variables 'a', 'b', 'c' in lines 1, 2 and 3 respectively can never change when executing the script, regardless of the value carried by 'x'.

  DE: Die Werte der Variabeln 'a', 'b' und 'c' auf Zeilen 1, 2 und 3 können sich nie ändern wenn man das Script ausführt, egal welchen Wert 'x' hat.

  False

Manipulating sequences recursively / Sequenzen rekursiv manipulieren

Punkte / Points: 2 Kprim

Frage / Question

EN: Which of the following statements are true about the python script given below?

DE: Welche der folgenden Aussagen treffen auf dieses Pythonscript zu?

Snippet

```
def func(a, b):
   return func(a[1:], b) + [b(a[0])] if a else []
```

Antworten / Answers

- EN: With the definition of the function 'square' as given below, invoking 'func([1, 2, 3], square())' returns the list '[1, 4, 9]'. def square(x): return x*x

  DE: Mit der oben angegebenen definition von 'square' gibt ein Aufruf von 'func([1, 2, 3], square())' die Liste '[1, 4, 9]' zurück.

  False

- EN: With the definition of the function 'id' as given below, for any list 'lst', 'func' can be used to' create another list containing the elements of 'lst' in reverse order. def id(x): return x

  DE: Mit der oben angegebenen definition von 'id' kann man 'func' benutzen um für eine beliebige Liste 'lst' eine neue Liste zu generieren welche die Elemente von 'lst' in umgekehrter Reihenfolge enthält.

  Correct

- EN: With the definition of the function 'func1' as given below, invoking 'func(['a', 'b', 'c'], todo)' returns the list '[None, None, None]. def todo(x): pass

  DE: Mit der oben angegebenen definition von 'todo' gibt ein Aufruf von 'func(['a', 'b', 'c'], todo)' die Liste '[None, None, None]' zurück.

  Correct

- EN: The function 'func' is guaranteed to crash if 'a' is a tuple.

  DE: Die Funktion 'func' wird garantiert abstürzen wenn 'a' ein Tupel ist.

  False

Using dictionaries. / Verwendung von Dictionaries.

Punkte / Points: 2 Kprim

Frage / Question

EN: Which of the following statements are true about the python script given below?

DE: Welche der folgenden Aussagen treffen auf dieses Pythonscript zu?

Snippet

```
str_freq = {}
def func(str_list):
    str_freq = {}
```

```
    for s in str_list:
        str_freq[s] = str_freq[s] + 1
    return str_freq
```

Antworten / Answers

- EN: Function 'func' contains a bug guaranteed to lead to a crash for any non empty instance of 'str_list'.

  DE: Die Funktion 'func' enthält einen Bug, der für beliebige nicht-leere Werte für 'str_list' garantiert zu einem crash führt.

  Correct

- EN: Function 'func' contains a bug guaranteed to lead to a crash for any instance of 'str_list'.

  DE: Die Funktion 'func' enthält einen Bug, der für beliebige Werte für 'str_list' garantiert zu einem crash führt.

  False

- EN: Every time function 'func' is invoked the value of 'str_freq' (declared in line 1) is overwritten.

  DE: Wann immer die Funktion 'func' aufgerufen wird, wird der Wert von 'str_freq' (deklariert auf Zeile 1) überschrieben.

  False

- EN: Function 'func' when invoked as is on any value of 'str_freq', can never modify the value of 'str_freq' (declared in line 1).

  DE: Egal mit welchem Wert für 'str_freq' die Funktion 'func' aufgerufen wird, sie kann niemals den Wert von 'str_freq' (deklariert auf Zeile 1) verändern.

  Correct

Programming 3

Gefragt / Asked: 1

parsing

Punkte / Points: 18

Frage / Question

DEUTSCH

Implementieren Sie eine Funktion `parse` welche einen einzigen String als einzigen Parameter `text` annimmt. Dieser String kann folgendes enthalten:

- Worte (Sequenzen von ASCII Buchstaben A bis Z und a bis z)
- Integer, und
- Floats,

getrennt durch eine beliebige anzahl Leerzeichen. Die Funktion soll eine Liste zurückgeben, die alle individuellen Element aus `text` in der folgenden Reihenfolge enthält: Erst alle Worte, dann alle Integer, dann alle Floats. Innerhalb dieser drei Gruppen sollen die Elemente in der natürlichen Reihenfolge sortiert sein, wie sie Python mittels den Funktionen `sorted()` und `.sort()` anbietet (d.h. Sie müssen keinen Sortieralgorithmus selber implementieren). Sie können davon ausgehen, dass `text` der `parse` übergeben wird, nur Zeichen wie oben spezifiziert enthält. Sie müssen nicht überprüfen, ob ungültige Zeichensequenzen vorkommen. Beachten Sie die Assertions als Beispiele für die Anwendung von `parse`.

ENGLISH

Implement a function `parse` which takes a single String `text` as the only parameter. This string can contain:

- Words (sequences of ASCII characters A through Z and a through z)
- Integers, and
- Floats,

separated by any number of spaces. The function should return a list containing all the individual elements from `text` in the following order: first all the words, then all the integers, then all the floats. Within each of these three groups, the elements should be sorted in the natural order that Python provides via the `sorted()` and `.sort()` functions (i.e. you do not need to manually implemented a sorting algorithm). You may assume that any `text` passed to `parse` will only contain characters as described above. You do not need to perform input validation. Consider the assert statements given below as examples for `parse`.

Fügen Sie ihre Lösung in folgendes Textfeld ein / Paste your solution into the following text field.

```
def parse(text):
    pass

# assert(parse("") == [])
# assert(parse("B A 2.5 1 5.5 C 2") == ["A", "B", "C", 1, 2, 2.5, 5.5])
# assert(parse("B A 1 2") == ["A", "B", 1, 2])
# assert(parse("  2.5        5.5 2 abc Abcd 1 Abc  ") == ["Abc", "Abcd", "abc", 1, 2, 2.5, 5.5])
```

Lösung / Solution

```
#!/usr/bin/env python3

def parse(text):
    strings = []
    ints = []
    floats = []
    tokens = text.split()
    for token in tokens:
        if "." in token:
            floats.append(float(token))
        elif token.isdigit():
            ints.append(int(token))
        else:
            strings.append(token)
    res = []
    res.extend(sorted(strings))
    res.extend(sorted(ints))
    res.extend(sorted(floats))
    return res
```

repair_bill

Punkte / Points: 18

Frage / Question

DEUTSCH

Implementieren Sie eine Funktion `bill`, welche folgende Parameter annimmt:

1. `per_hour`: eine Gleitkommazahl die den Stundensatz eines Arbeiters angibt und
2. `parts_prices_hours`: ein Dictionary in der Form `{part: (price, hours)}`, wo `part` der Name eines Bauteils ist, `price` den Preis des Bauteils repräsentiert, und `hours` die Anzahl Stunden als Gleitkommazahl repräsentiert, die benötigt werden, um das Bauteil zu installieren. `price` und `hours` können für gewisse Einträge in `parts_prices_hours` `None` sein.

Die Funktion soll ein Tupel mit zwei Elementen zurückgeben, wo das erste Element die totalen Kosten für den Kauf und die Installation aller Teile angibt, und das zweite Element ein boolean ist, welcher angibt, ob irgendwelche Informationen für die Berechnung fehlten. Die Kosten für die Installation eines einzelnen Teil soll wie folgt berechnet werden: part_price + (parts_hours *

per_hour) und die totalen Kosten berechnen sich aus der Summe der Kosten für die Installation aller Teile. Wenn `price` eines Teils `None`, ist, soll ein Preis von 0 für dieses Teil angenommen werden, und wenn `hours`, für ein Teil `None` ist, soll eine Stunde angenommen werden. Beachten Sie die Assertions als Beispiele für die Anwendung von `bill`.

ENGLISH

Implement a function `bill`, which takes as parameters:

1. `per_hour`: a float representing the hourly rate of a worker, and
2. `parts_prices_hours`: a dictionary of the form `{part: (price, hours)}`, where `part` is the name of a part, `price` is a float representing the price of that part, and `hours` is a float representing the total amount of hours needed to install that part. `price` and `hours` can be `None` for some entries in `parts_prices_hours`.

The function should return a tuple with two items, where the first item represents the total cost of buying and installing all of the parts and the second item is a boolean, reporting whether some information was missing whilst computing the total cost. The cost for installing each individual part should be computed as: part_price + (parts_hours * per_hour) and the total cost is the sum of installing all parts. If the `price` of a part is `None`, a zero cost should be assumed for that part, and if `hours`, of a part is `None` one hour should be assumed. Consider the assert statements given below as examples for `bill`.

Fügen Sie ihre Lösung in folgendes Textfeld ein / Paste your solution into the following text field.

```
def bill(per_hour, parts_prices_hours):
    pass

# assert(bill(0, {'Door': (23.33, 2.50)}) == (23.33, False))
# assert(bill(0, {}) == (0, False))
# assert(bill(0, {'Door': (None, 3.0)}) == (0, True))
# assert(bill(50, {'Door': (None, 3.0)}) == (150, True))
# assert(bill(50, {'Door': (10, None)}) == (60, True))
# assert(bill(10.5, {'Door': (None, 3.0), 'Window': (22.22, 0.5)}) == (58.97, True))
# assert(bill(10.5, {'Window': (22.22, 0.5)}) == (27.47, False))
```

Lösung / Solution

```
#!/usr/bin/env python3

def bill(per_hour, parts_prices_hours):
    total = 0
    missing = False
    for cost, hours in parts_prices_hours.values():
        if cost == None:
            cost = 0
            missing = True
        if hours == None:
            hours = 1
            missing = True
        total += (hours * per_hour) + cost
    return (total, missing)
```

shopping

Punkte / Points: 18

Frage / Question

DEUTSCH

Implementieren Sie eine Funktion `organise`, die eine Liste `records` annimmt, welche null oder mehrere Kaufereignisse enthält. Diese Liste enthält Tupel in der Form `(username, shop_name, day_of_week)`, wobei: `username` ein String ist, der den Namen eines Kunden enthält (z.b. 'Tom'), `shop_name` ein String ist, der den Namen eines Ladens enthält (z.b. 'Denner'), und `day_of_week` eine Zahl zwischen 1 und (einschliesslich) 7 ist, welche den Wochentag zwischen Montag und Sonntag repräsentiert. Semantisch gibt jeder Eintrag an, dass ein gewisser Kunde in einem bestimmten Laden an einem bestimmten Wochentag eingekauft hat. Somit soll `organise` die Einträge verarbeiten, um ein Profil des Kaufverhaltens jedes Kunden zu erstellen. Das Kaufverhalten soll in einem Dictionary abgebildet werden, welches wie folgt strukturiert ist:

`{username: {shop_name: {day_of_week: counter}}}`. Das bedeutet, dass für jeden Kunden (`{username: ...}`), ein Dictionary erstellt wird, welches wiederum für jeden Laden (`{shop_name: ...}`) ein Dictionary enthält, welches die Anzahl Besuche (`counter`) für jeden Wochentag angibt (`{day_of_week: counter}`). Ihre Implementation von `organise`, soll alle Einträge kommentarlos ignorieren, wo `username` oder `shop_name` None oder leere Strings sind, und ebenfalls wenn `day_of_week` ausserhalb der oben angegebenen Reichweite ist. Beachten Sie die Assertions als Beispiele für die Anwendung von `organise`.

ENGLISH

Implement a function `organise`, which takes a list of zero or more purchase events `records`. This is a list of tuples where each tuple represents a purchase and follows the pattern `(username, shop_name, day_of_week)`, where: `username` is a string carrying the identifier of a user (eg: 'Tom'), `shop_name` is a string carrying the identifier of a shop (eg: 'Denner'), and `day_of_week` is an integer between 1 and 7 (included) representing the day of the week from Monday to Sunday. Semantically, each record indicates that a certain user visited a certain shop on a certain day of the week. Hence, `organise` processes these records to create a concise profile of the buying habits for every user. These buying habits should be represented in a dictionary structured as: `{username: {shop_name: {day_of_week: counter}}}`. This means that for each user (`{username: ...}`), a dictionary is created which in turn for each shop (`{shop_name: ...}`) creates a nested dictionary carrying the number of times (`counter`) the user visited on a certain day of the week (`{day_of_week: counter}`). Your implementation of `organise`, should quietly ignore all tuples in the provided list where `username` or `shop_name` are None or an empty string, and also if `day_of_week` is outside the range specified above. Consider the assert statements given below as examples for `organise`.

Fügen Sie ihre Lösung in folgendes Textfeld ein / Paste your solution into the following text field.

```
def organise(records):
    pass

# assert(organise([]) == {})
# assert(organise([('Tom', '', 5), ('Tom', 'Aldi', 4)]) == {'Tom': {'Aldi': {4: 1}}})
# assert(organise([('Tom', 'Aldi', 5), ('Tom', 'Aldi', 5)]) == {'Tom': {'Aldi': {5: 2}}})
# assert(organise([('Tom', 'Aldi', 1), ('Tom', 'Migros', 4), ('Jack', 'Aldi', 5)]) == {'Jack':
        {'Aldi': {5: 1}}, 'Tom': {'Aldi': {1: 1}, 'Migros': {4: 1}}})
```

Lösung / Solution

```
#!/usr/bin/env python3

def organise(records):
    # { user: {shop -> {day -> counter}}}
    res = {}
    for person, shop, day in records:
        if person and shop and (0 < day < 8):
            if person not in res:
                res[person] = {}
            if shop not in res[person]:
                res[person][shop] = {}
            if day not in res[person][shop]:
                res[person][shop][day] = 0
            res[person][shop][day] += 1
    return res
```