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

Warmup Questions

Gefragt / Asked: 2

Computing

Punkte / Points: 2 Kprim

Frage / Question

EN: Entscheiden Sie, welche der folgenden Aussagen wahr sind und kreuzen Sie diese an.

DE: For each of the following statements, check the box if the statement is true.

Antworten / Answers

- EN: Binärzahlen im Computer haben eine fixe Bedeutung: es ist unmöglich, dieselbe Zahlensequenz auf unterschiedliche Arten zu interpretieren.

  DE: Binary numbers in a computer have a fixed meaning: it is not possible, to interpret the same sequence of numbers in different ways.

  False

- EN: Per Definition muss ein Algorithmus zwingend irgendwann stoppen.

  DE: An algorithm must, by definition, halt eventually.

  Correct

- EN: Für jedes Problem gibt es genau einen Algorithmus als Lösung.

  DE: For every problem, there exists exactly one algorithm to solve it.

  False

- EN: Die input() Funktion gibt immer einen String zurück.

  DE: The input() function always returns a string.

  Correct

Git

Punkte / Points: 2 Kprim

Frage / Question

EN: Entscheiden Sie, welche der folgenden Aussagen wahr sind und kreuzen Sie diese an.

DE: For each of the following statements, check the box if the statement is true.

Antworten / Answers

- EN: Gegeben ist folgende Sachlage: zwei Personen A und B klonen dieselbe Git repository und erstellen beide einen lokalen Commit. Dann pusht A ihren Commit. Entscheiden Sie ob folgende Aussage wahr ist: B muss erst A's Änderungen pullen, bevor sie ihre eigenen

pushen kann.

DE: Consider the following sequence of events: two persons A and B clone the same Git repository and both create a local commit. Then, A pushes their commit. Decide whether the following statement is true: B will first have to pull A's changes before being able to push his own.

Correct

- EN: Um Änderungen von einer lokalen Git repository auf eine entfernte Git repository hochzuladen, verwendet mann den Befehl `git commit`.

  DE: To upload changes from the local Git repository to a remote Git repository, the `git commit` command is used.

  False

- EN: Wenn zwei Branches Änderungen enthalten, die nicht-überlappende Teile des Quellcodes betreffen, kann Git die Änderungen automatisch in einen einzigen Branch zusammenführen.

  DE: If two branches contain changes that affect disjoint parts of the source code, Git can automatically merge the changes into a single branch.

  Correct

- EN: Commit messages sind optional.

  DE: Commit messages are optional.

  False

Inheritance

Punkte / Points: 2 Kprim

Frage / Question

EN: Entscheiden Sie, welche der folgenden Aussagen wahr sind und kreuzen Sie diese an.

DE: For each of the following statements, check the box if the statement is true.

Antworten / Answers

- EN: Eine Klasse kann von mehreren anderen Klasse erben.

  DE: A class can inherit from multiple other classes.

  Correct

- EN: Um zu erzwingen, dass eine Subklasse eine bestimmte Methode der Superklasse überschreibt, muss diese mit `@staticmethod` dekoriert werden.

  DE: To enforce that subclasses override and implement a given method, it must be decorated with `@staticmethod`.

  False

- EN: Wenn eine Klasse C von einer Klasse B erbt, welche wiederum von einer Klasse A erbt, dann erbt C die Funktionalität von A, wo diese nicht von B überschrieben wurde.

  DE: If a class C inherits from a class B, which in turn inherits from a class A, then C inherits functionality from class A where it has not been overridden by class B.

  Correct

- EN: Wenn eine Subklasse B den Konstruktor einer Superklasse A überschreibt und ein Objekt vom Typ B instanziert wird, dann wird der Konstruktor von A auch aufgerufen, egal wie der Konstruktor von B implementiert wurde.

  DE: If a subclass B overrides the constructor of a superclass A, and an object of type B is instantiated, the constructors of A is also called, no matter how the constructor of B is

implemented.

False

Object-oriented

Punkte / Points: 2 Kprim

Frage / Question

EN: Entscheiden Sie, welche der folgenden Aussagen wahr sind und kreuzen Sie diese an.

DE: For each of the following statements, check the box if the statement is true.

Antworten / Answers

- EN: Wenn man eine Klasse definiert muss man zwingend auch einen Konstruktor definieren.

  DE: When writing a class definition, it is mandatory to include a constructor definition.

  False

- EN: Eine statische Methode erhält keine Referenz zu `self`.

  DE: A static method does not receive a reference to `self`.

  Correct

- EN: Standardmässig werden zwei Instanzen derselben Klasse als equivalent angesehen, wenn alle deren Instanzattribute equivalent sind.

  DE: By default, two distinct instances of the same class are considered equal if all their instance attributes are equal.

  False

- EN: Wenn man die `__str__`-Methode einer Klasse überschreibt bewirkt das, dass eine aussagekräftige Repräsentation dieser Instanzen angezeigt wird, wenn man eine Liste von Instanzen mittels der print-Funktion ausgibt.

  DE: Overriding the `__str__` method of a class makes it so that printing a list of instances of that class will show a meaningful representation of the elements.

  False

Recursion

Punkte / Points: 2 Kprim

Frage / Question

EN: Entscheiden Sie, welche der folgenden Aussagen wahr sind und kreuzen Sie diese an.

DE: For each of the following statements, check the box if the statement is true.

Antworten / Answers

- EN: Eine rekursive Funktion kann mehr als eine Abbruchbedingung beinhalten.

  DE: A recursive function can have more than one termination condition.

  Correct

- EN: Eine rekursive Funktion kann mehr als einen rekursiven Aufruf enthalten.

DE: A recursive function can have more than one recursive call.

Correct

- EN: Die Kette von rekursiven Aufrufen einer Funktion kann beliebig lang sein.

  DE: The chain of rekursive calls of a function can be arbitrarily long.

  False

- EN: Für jede rekursive Implementation ist es möglich eine iterative Alternative zu implementieren.

  DE: For every recursive implementation, it is possible to implement an iterative alternative.

  Correct

Errors

Punkte / Points: 2 Kprim

Frage / Question

EN: Entscheiden Sie, welche der folgenden Aussagen wahr sind und kreuzen Sie diese an.

DE: For each of the following statements, check the box if the statement is true.

Antworten / Answers

- EN: Der Python-Interpreter kann semantische Programmierfehler erkennen.

  DE: The Python interpreter can detect semantic errors.

  False

- EN: Ein Programm kann auch dann ausgeführt werden, wenn es syntaktische Fehler enthält.

  DE: A program can be run, even if it contains syntactic errors.

  False

- EN: Ein Programm kann auch dann ausgeführt werden, wenn es semantische Fehler enthält.

  DE: A program can be run, even if it contains semantic errors.

  Correct

- EN: Wenn ein Programm einen NameError verursacht, enthält dieses Programm syntaktische Fehler.

  DE: If a program causes a NameError, then it contains syntactic errors.

  False

Debugging and Testing

Punkte / Points: 2 Kprim

Frage / Question

EN: Entscheiden Sie, welche der folgenden Aussagen wahr sind und kreuzen Sie diese an.

DE: For each of the following statements, check the box if the statement is true.

Antworten / Answers

- EN: Ein Debugger kann Bugs in Quellcode automatisch erkennen und fixen.

  DE: A debugger can automatically detect and fix bugs in source code.

  False

- EN: Namen von Testmethoden für einen unit test müssen mit `test_` beginnen.

  DE: Names of test methods for a unit test must start with `test_`.

  Correct

- EN: Es ist möglich, ein Programm normal weiterlaufen zu lassen, auch wenn man es zuvor mit einem Debugger pausiert hat.

  DE: It's possible to resume normal execution even after having paused a program using a debugger.

  Correct

- EN: Idealerweise sollte ein Unit test nur eine Sache testen und deswegen genau eine Assertion beinhalten.

  DE: Ideally, a unit test should test only one thing and, therefore, contain exactly one assertion.

  Correct

Warmup Programming

Gefragt / Asked: 2

Warmup: calc

Punkte / Points: 8

Aufgabe / Task

DEUTSCH

Implementieren Sie eine Funktion `calc`, die einen String `expression` annimmt, welche einen einfachen mathematischen Ausdruck in prefix-notation `operator x y` (eine Addition, Subtraktion, Multiplikation oder Division) enthält. Die Funktion soll ganze und Gleitkommazahlen verarbeiten können und das Ergebnis als Zahl zurückgeben. Wenn durch 0 dividiert wird, soll ein ValueError verursacht werden. Beachten Sie die Assertions als Beispiele für die Anwendung von `calc`.

ENGLISH

Implement a function `calc`, which takes a string `expression` that contains a simple mathematical expression in prefix notation `operator x y` (an addition, subtraction, multiplication or division). The function should be able to process integers and floats, and return the result as a number. If dividing by zero, a ValueError should be raised. Consider the assertions given below as examples for using `calc`.

```
def calc(expression):
    pass

# DO NOT SUBMIT THE LINES BELOW!
#assert calc("+ 1 2") == 3
#assert calc("- 1 2") == -1
#assert calc("* 1 2") == 2
#assert calc("/ 1 2") == 0.5
#assert calc("* 1 -2") == -2
#assert calc("* 10.5 2") == 21
```

```
#assert calc("* -10.5 -2") == 21
```

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

Lösung / Solution

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

def calc(expression):
    op, left, right = expression.split()
    left = float(left)
    right = float(right)
    if op == "+":
        return left + right
    if op == "-":
        return left - right
    if op == "/":
        if right == 0:
            raise ValueError
        return left / right
    if op == "*":
        return left * right
```

Warmup: count_flat_lists

Punkte / Points: 8

Aufgabe / Task

DEUTSCH

Implementieren Sie eine **rekursive** Funktion `count`, die eine Liste `l` annimmt, welche beliebige Elemente enthalten kann, einschliesslich verschachtelte Listen. Die Funktion soll die Anzahl Elemente in `l` zählen, wobei jede verschachtelte Liste besonders behandelt werden soll: anstatt sie einfach als einzelnes Element zu zählen, soll sie in einem rekursiven Aufruf an `count` übergeben werden, damit auch die Elemente in der verschachtelten Liste einzeln mitgezählt werden. Die Liste selbst soll dabei nicht mitgezählt werden. Beachten Sie die Assertions als Beispiele für die Anwendung von `calc`. Falls Ihre Lösung nicht rekursiv ist, erhalten Sie höchstens die Hälfte der erreichbaren Punkte.

ENGLISH

Implement a **recursive** function `count`, which takes a list `l` that contains arbitrary Elements, including nested lists. The function should count the number of elements in `l` while specially treating each nested list: instead of simply counting it as single element, it should be passed to `count` in a recursive call, so that the nested Elements are counted individually. The list itself should not be counted. Consider the assertions given below as examples for using `count`. If your solution is not recursive, you will receive at most half of the attainable points.

```
def count(l):
    pass

# DO NOT SUBMIT THE LINES BELOW!
#assert count([]) == 0
#assert count([[],[]]) == 0
#assert count([1, "", [{}], [[True], 4]]) == 5
```

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

Lösung / Solution

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

def count(l):
    res = 0
    for e in l:
        if isinstance(e, list):
            res += count(e)
        else:
```

```
            res += 1
    return res
```

Warmup: count_palindromes

Punkte / Points: 8

Aufgabe / Task

DEUTSCH

Implementieren Sie eine Funktion `count_palindromes` die einen String als Parameter `sentence` annimmt. Die Funktion soll die Anzahl Palindrome in `sentence` als Integer zurückgeben. Ein Palindrom ist ein Wort, das vorwärts wie rückwärts gleich gelesen werden kann. Ein Palindrom muss mindestens 3 Zeichen lang sein. Die Funktion soll Gross- und Kleinschreibung sowie Sonderzeichen in Worten ignorieren. Beachten Sie die Assertions als Beispiele für die Anwendung von `count_palindromes`.

ENGLISH

Implement a function `count_palindromes`, which takes a string as parameter `sentence`. The function should return the number of palindromes in `sentence` as an integer. A palindrome is a word that can be read the same forwards and backwards. A palindrome must be at least 3 characters long. The function should be case-insensitive and ignore special characters in words. Consider the assertions given below as examples for using `count_palindromes`.

```
def count_palindromes(sentence):
    pass

# DO NOT SUBMIT THE LINES BELOW!
#assert count_palindromes("Having fun!") == 0
#assert count_palindromes("Bob and otto") == 2
#assert count_palindromes("Where's Dad?") == 1
#assert count_palindromes("Otto is my dad.") == 2
#assert count_palindromes("I don't like pop music") == 1
```

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

Lösung / Solution

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

def count_palindromes(sentence):
    tokens = sentence.split()
    res = 0
    for t in tokens:
        t = "".join([c for c in t if c.isalpha()])
        if len(t) < 3: continue
        if t.lower() == t[::-1].lower():
            res += 1
    return res
```

Warmup: currency_converter

Punkte / Points: 8

Aufgabe / Task

DEUTSCH

Implementieren Sie eine Funktion `currency_converter` die drei Parameter annimmt: zwei Strings `src` und `dst`, welche die Namen der Ausgangs- und Zielwährungen repräsentieren, sowie eine Zahl `rate`, welche den Umrechnungskurs darstellt. Die Funktion

`currency_converter` soll eine Funktion zurückgeben, welche den gewünschten Geldbetrag in der Ausgangswährung als einzigen Parameter annimmt, und einen String zurückgibt in der Form `X SRC is Y DST`. Der berechnete Umrechnungswert soll auf 2 Nachkommastellen gerundet werden. Beachten Sie die Assertions als Beispiele für die Anwendung von `currency_converter`.

ENGLISH

Implement a function `currency_converter`, which takes three parameters: two strings `src` and `dst`, which represent the names of the source and destination currencies, and a number `rate` which is the conversion rate. The function `currency_converter`, should return a function that takes the desired amount in the source currency as the only parameter and returns a string like `X SRC is Y DST`. The converted value should be rounded to two decimal places. Consider the assertions given below as examples for using `currency_converter`.

```
def currency_converter(src, dst, rate):
    pass

# DO NOT SUBMIT THE LINES BELOW!
#assert currency_converter("EUR", "CHF", 1.1)(5) == "5 EUR is 5.5 CHF"
#chf_to_jpy = currency_converter("CHF", "JPY", 123)
#assert chf_to_jpy(1) == "1 CHF is 123 JPY"
#assert chf_to_jpy(2) == "2 CHF is 246 JPY"
#assert currency_converter("Peanuts", "Pinecones", 0.2)(50) == "50 Peanuts is 10.0 Pinecones"
#assert currency_converter("Blemflarcks", "Coins", 0.0021)(333.3) == "333.3 Blemflarcks is 0.7 Coins"
```

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

Lösung / Solution

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

def currency_converter(src, dst, rate):
    # using a nested function
    def convert(n):
        res = round(rate * n, 2)
        return f"{n} {src} is {res} {dst}"
    return convert
    # or using a lambda
    return lambda n: f"{n} {src} is {round(rate * n, 2)} {dst}"
```

Warmup: intersperse

Punkte / Points: 8

Aufgabe / Task

DEUTSCH

Implementieren Sie eine Funktion `intersperse`, die zwei Parameter annimmt: einen nicht-leeren String `s` und eine nicht-leere Liste von Strings `l`. Die Funktion soll den String `s` zurückgeben, wobei die Elemente aus `l` eines nach dem anderen zwischen die Buchstaben in `s` eingefügt werden sollen. Wurde das letzte Element aus `l` eingefügt, so soll wieder mit dem ersten begonnen werden. Beachten Sie die Assertions als Beispiele für die Anwendung von `intersperse`.

ENGLISH

Implement a function `intersperse`, which takes two parameters: a non-empty string `s` and a non-empty list of strings `l`. The function should return the string `s` while the elements from `l` should be inserted one by one in-between the characters in `s`. Once the last element in `l` has been inserted, the selection should return to the beginning. Consider the assertions given below as examples for using `intersperse`.

```
def intersperse(s, l):
    pass

# DO NOT SUBMIT THE LINES BELOW!
#assert intersperse("H", [',']) == "H"
#assert intersperse("Hello", [',']) == "H,e,l,l,o"
```

```
#assert intersperse("Hello", [',', '.']) == "H,e.l,l.o"
#assert intersperse("Hello", ['', '.']) == "He.ll.o"
#assert intersperse("Hello", ['-o-', '_o_']) == "H-o-e_o_l-o-l_o_o"
#assert intersperse("Hello", [',', '.', '-']) == "H,e.l-l,o"
```

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

Lösung / Solution

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

def intersperse(s, l):
    res = s[0]
    delim = 0
    for c in s[1:]:
        res += l[delim % len(l)]
        res += c
        delim += 1
    return res
```

Warmup: is_isogram

Punkte / Points: 8

Aufgabe / Task

DEUTSCH

Implementieren Sie eine Funktion `is_isogram` die einen Parameter `sentence` annimmt. Die Funktion soll feststellen, ob `sentence` ein Isogramm ist. Ein Isogramm ist eine Zeichensequenz, wo jeder Buchstabe gleich oft vorkommt. Wenn `sentence` ein Isogramm ist, soll `True` zurückgegeben werden, ansonsten `False`. Die Gross- und Kleinschreibung soll dabei keine Rolle spielen, und Sonderzeichen sollen ignoriert werden. Wenn `sentence` kein String ist, soll ein TypeError verursacht werden. Wenn `sentence` leer ist oder nur Sonderzeichen enthält, soll ein ValueError verursacht werden. Beachten Sie die Assertions als Beispiele für die Anwendung von `is_isogram`.

ENGLISH

Implement a function `is_isogram`, which takes a parameter `sentence` The function should determine if `sentence` is an isogram. An isogram is a sequence of characters, where every letter appears the same number of times. If `sentence` is an isogram, the function should return `True`, `False` otherwise. The function should be case-insensitive and ignore special characters. If `sentence` is not a String, a TypeError should be raised. If `sentence` is the empty string or contains only special characters, a ValueError should be raised. Consider the assertions given below as examples for using `is_isogram`.

```
def is_isogram(sentence):
    pass

# DO NOT SUBMIT THE LINES BELOW!
#assert is_isogram("uncopyrightable")
#assert is_isogram("The big dwarf only jumps.")
#assert is_isogram("Apple-ale")
#assert (not is_isogram("bass"))
#assert (not is_isogram("Tart"))
```

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

Lösung / Solution

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

def is_isogram(sentence):
    sentence = "".join(c for c in sentence if c.isalpha())
    if not isinstance(sentence, str):
        raise TypeError
    if len(sentence) == 0:
        raise ValueError
    sentence = sentence.lower()
```

```
    first = sentence[0]
    n = sentence.count(first)
    for c in sentence:
        if sentence.count(c) != n:
            return False
    return True
```

Warmup: words_by_len

Punkte / Points: 8

Aufgabe / Task

DEUTSCH

Implementieren Sie eine Funktion `words_by_len` die einen Parameter `sentence` annimmt. Wenn `sentence` kein String ist, soll ein AssertionError verursacht werden. Der String enthält Worte, die aus beliebigen Zeichen bestehen. Einzelne Worte sind durch einen oder mehrere Leerzeichen getrennt. Die Funktion soll ein Dictionary zurückgeben, dessen Keys Integer und dessen Werte Sets sind. Der Key gibt jeweils an, wie lang ein Wort ist, und der Wert enthält die Worte der entsprechenden Länge in einem Set. Beachten Sie die Assertions als Beispiele für die Anwendung von `words_by_len`.

ENGLISH

Implement a function `words_by_len`, which takes a string `sentence` as a parameter. If `sentence` is not a string, an AssertionError should be raised. The string contains words comprised of arbitrary characters. Individual words are separated by one or more spaces. The function should return a dictionary, where keys are integers and values are sets. Each key indicates the length of a word, and the corresponding value contains a set with all the words having this same length. Consider the assertions given below as examples for using `words_by_len`.

```
def words_by_len(sentence):
    pass

# DO NOT SUBMIT THE LINES BELOW!
#assert words_by_len("how are ya?") == {3: {"how", "are", "ya?"}}
#assert words_by_len(" I'm      so so ") == {2: {"so"}, 3: {"I'm"}}
#assert words_by_len("Get well soon !!") == {2: {"!!"}, 3: {"Get"}, 4: {"well", "soon"}}
#assert words_by_len("") == {}
```

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

Lösung / Solution

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

def words_by_len(sentence):
    assert isinstance(sentence, str)
    tokens = sentence.split()
    res = {}
    for t in tokens:
        l = len(t)
        if l == 0: continue
        if l not in res:
            res[l] = set()
        res[l].add(t)
    return res
```

Warmup: work

Punkte / Points: 8

Aufgabe / Task

DEUTSCH

Implementieren Sie eine Funktion `work`, die drei Parameter annimmt: eine Zahl `hours` und einen Integer `days`, sowie einen optionalen Boolean `teen`, welcher standardmässig `False` sein soll. Sie dürfen die Funktionssignatur entsprechend anpassen, sodass `teen` einen Defaultwert erhält. Die Funktion soll zwei Werte berechnen:

1. wieviele Stunden man durchschnittlich pro Tag arbeiten muss, gegeben der Anzahl Wochenstunden `hours` und der Anzahl Arbeitstage pro Woche `days`, wobei dieser Wert auf 2 Nachkommastellen gerundet werden soll.
2. wieviele Ferientage man pro Jahr erhält. Bei einer Vollzeistelle erhält man 25 Ferientage wenn man das 20. Lebensjahr noch nicht erreicht hat, ansonsten 20 Tage. Wenn der Parameter `teen` `True` ist, handelt es sich um einen jungen Menschen. Eine Vollzeitstelle entspricht 42 Wochenstunden. Die Anzahl Ferientage soll immer auf den nächsten Integer **auf**gerundet werden.

Die beiden Werte sollen in einem Dictionary in der Form `{"per_day": x, "vacation_days": y}` zurückgegeben werden. Beachten Sie die Assertions als Beispiele für die Anwendung von `work`.

ENGLISH

Implement a function `work`, which takes three parameters: a number `hours` and an integer `days` as well as an optional boolean `teen` which should be `False` by default. You may modify the function signature accordingly, so that `teen` receives a default value. The function should calculate two values:

1. how many hours one has to work per day, given the number of hours per week `hours` and the number of work days per week `days`. This value should be rounded to two decimal points.
2. how many vacation days one gets per year. With a full-time job 25 vacation days are permitted if the person has not reached the age of 20 years, while everyone else gets 20 days. If the parameter `teen` is `True` the person is young. A full-time job corresponds to 42 hours per week. The number of vacation days should always be rounded **up** to the next integer.

The two values should be returned as a dictionary like `{"per_day": x, "vacation_days": y}`. Consider the assertions given below as examples for using `work`.

```
def work(hours, days, teen):
    pass

# DO NOT SUBMIT THE LINES BELOW!
#assert work(42, 5) == {"per_day": 8.4, "vacation_days": 20}
#assert work(42, 5, True) == {"per_day": 8.4, "vacation_days": 25}
#assert work(34, 5) == {"per_day": 6.8, "vacation_days": 17}
#assert work(16.55, 4, True) == {"per_day": 4.14, "vacation_days": 10}
```

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

Lösung / Solution

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

import math

def work(hours, days, teen=False):
    max_vac = 25 if teen else 20
    percent = hours/42
    per_day = round(hours/days, 2)
    return {"per_day": per_day, "vacation_days": math.ceil(max_vac*percent)}
```

Functions

Gefragt / Asked: 1

Functions: draw

Punkte / Points: 18

Aufgabe / Task

DEUTSCH

Implementieren Sie eine Funktion `draw` die zwei Parameter annimmt: Einen Integer `size` und eine Liste von Tupeln `instructions`. Die Funktion soll mittels der Anweisungen in `instructions` einen String bestehend aus `size` Zeilen mit je `size` Buchstaben generieren und zurückgeben. Man stelle sich dies als quadratische, 2-dimensionale "Leinwand" vor auf der man mittels jeder Instruktion einen einzelnen Buchstaben auftragen kann. Jedes Tupel in `instructions` enthält dementsprechend 3 Werte:

- Die X-Koordinate
- Die Y-Koordinate
- Den zu verwendenden Buchstaben

Der erste Buchstabe auf der ersten Zeile hat die Koordinaten 0/0. Die Instruktionen können ungültige Werte enthalten:

- Wenn eine X- oder Y-Koordinate ausserhalb der durch `size` definierten Malfläche liegt, soll ein `IndexError` verursacht werden.
- Wenn der zu verwendende Buchstabe nicht genau die Länge 1 hat, soll ein `ValueError` verursacht werden.

Koordinaten, wo kein Buchstabe plaziert wurde, sollen ein Leerzeichen anzeigen. Sie können davon ausgehen, dass `size` nie kleiner als 1 ist. Beachten Sie die Assertions als Beispiele für die Anwendung von `draw`.

Tip:Benutzen Sie **nicht** ein statement wie `canvas = size * [ size * [" "] ]` um ihre Leinwand zu initialisieren! Dies würde nicht size*size Leerzeichen generieren, sondern nur **eine** innere Liste, die size mal in der äusseren referenziert wird.

ENGLISH

Implement a function `draw` that takes two parameters: An integer `size` and a list of tuples `instructions`. The function should use the instructions in `instructions` to generate and return a string consisting of `size` lines each having `size` characters. Imagine this as a square, 2-dimensional "canvas", onto which each instruction places a single character. As such, each tuple in `instructions` contains three values:

- The x coordinate
- The y coordinate
- The character to be drawn

The first character on the first line has the coordinate 0/0. The instructions may contain invalid values:

- If an x or y coordinate sits outside the canvas defined by `size`, an `IndexError` should be raised.
- If the character to be drawn does not have the exact length 1, a `ValueError` should be raised.

Coordinates which have not been drawn on should show a blank space. You may assume that `size` will never be smaller than 1. Consider the assertions given below as examples for using `draw`.

Hint: Do **not** use a statement like `canvas = size * [ size * [" "] ]` to initialize your canvas. This does not generate size*size blankspaces, but just **one** inner list, which is referenced size times in the outer list.

```
def draw(size, instructions):
    pass

# DO NOT SUBMIT THE LINES BELOW!
#assert draw(1, [(0, 0, "a")]) == "a"
#assert draw(2, [(0, 0, "a"), (1, 1, "b")]) == (
#     "a \n"
#     " b"
#)
#assert draw(3, [(1, 0, "a"), (0, 1, "b")]) == (
#     " a \n"
```

```
#     "b  \n"
#     "   "
#)
#
#assert draw(5, [
#  (0, 0, "a"),
#  (1, 2, "a"),
#  (4, 4, "b"),
#  (0, 4, "a"),
#  (1, 1, "a"),
#  (0, 3, "a")]) == (
#    "a    \n"
#    " a   \n"
#    " a   \n"
#    "a    \n"
#    "a   b"
#)
#
```

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

Lösung / Solution

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

def draw(size, instructions):
    canvas = []
    for x in range(size):
        canvas.append([])
        for y in range(size):
            canvas[x].append(" ")
    for i in instructions:
        x, y, c = i
        if x > size - 1: raise IndexError
        if y > size - 1: raise IndexError
        if len(c) != 1: raise ValueError
        canvas[y][x] = c
    lines = ["".join(chars) for chars in canvas]
    return "\n".join(lines)
```

Functions: ifc

Punkte / Points: 18

Aufgabe / Task

DEUTSCH

Der "International Fixed Calendar" (IFC) macht den Vorschlag, den gängigen Gregorianischen Kalender durch einen regelmässigeren Kalender zu ersetzen. Eine Vairante des IFC funktioniert so:

- Jeder Monat hat exakt 28 Tage (also genau 4 Wochen).
- Es gibt 13 Monate im Jahr (13*28 = 364).
- Das Jahr endet mit einem besonderen "Jahrestag", der keinem Monat und keiner Woche angehört.
- In einem Schaltjahr endet das Jahr mit zwei "Jahrestagen".

Implementieren Sie eine Funktion `gregorian_to_ifc` welche ein Gregorianisches Datum in der Form von zwei Integer `day` (zwischen 1 und 31) und `month` (zwischen 1 und 12) sowie einen optionalen Boolean `is_leap` als Parameter annimmt. `is_leap` gibt an, ob es sich um ein Schaltjahr mit 366 Tagen handelt, und ist standardmässig `False` (Sie dürfen die Signatur anpassen, um den Standardwert zu setzen). Die Funktion soll das übergebene Datum in ein IFC-Datum konvertieren. Sie soll den resultierenden Tag und Monat im IFC-Kalender als Tupel zurückgeben. Für das Ende des Jahres soll die Funktion den String `"Year Day"` zurückgeben. Beachten Sie die Assertions als Beispiele für die Anwendung von `gregorian_to_ifc`.

ENGLISH

The "International Fixed Calendar" (IFC) propose to replace the common Gregorian Calendar with a more regularly-paced alternative. One variant of the IFC works like this:

- Every month has exactly 28 days (exactly 4 weeks).
- There are 13 months in a year (13*28 = 364).
- The year ends with a special "Year Day", which does not belong to any month or week.
- In leap years, the year ends with two "Year Days".

Implement a function `gregorian_to_ifc` which takes a gregorian date as two integers `day` (between 1 and 31) and `month` (between 1 and 12) as well as an optional boolean `is_leap` as parameters. `is_leap` indicates, whether you're dealing with a leap year that has 366 days and is `False` by default (you may change the signature to set the default value). The function should convert the given date to an IFC-date. It should return the resulting day and month in the IFC calendar as as a tuple. For the end of the year, the function should return the String `"Year Day"`. Consider the assertions given below as examples for using `gregorian_to_ifc`.

```
def gregorian_to_ifc(day, month, leap):
    pass

# DO NOT SUBMIT THE LINES BELOW!
#assert gregorian_to_ifc(1, 1) == (1, 1)
#assert gregorian_to_ifc(28, 1) == (28, 1)
#assert gregorian_to_ifc(29, 1) == (1, 2)        # 29th Jan Gregorian is 1st Feb IFC
#assert gregorian_to_ifc(1, 3) == (4, 3)         # 1st  Mar Gregorian is 4rd Mar IFC
#assert gregorian_to_ifc(29, 2, True) == (4, 3)  # leap year
#assert gregorian_to_ifc(1, 8) == (17, 8)
#assert gregorian_to_ifc(15, 11) == (11, 12)
#assert gregorian_to_ifc(30, 12) == (28, 13)
#assert gregorian_to_ifc(30, 12, True) == "Year Day"
#assert gregorian_to_ifc(31, 12) == "Year Day"
```

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

Lösung / Solution

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

def gregorian_to_ifc(day, month, leap=False):
    month = month - 1
    day = day - 1
    days_per_month = [31, 28, 31, 30, 31, 30,
                      31, 31, 30, 31, 30, 31]
    if leap: days_per_month[1] = 29

    day_number = 0
    for m in range(month):
        day_number += days_per_month[m]
    day_number += day

    if day_number in [364, 365]:
        return "Year Day"

    ifc_month = day_number // 28
    ifc_day = day_number % 28

    return ifc_day+1, ifc_month+1
```

Functions: scale

Punkte / Points: 18

Aufgabe / Task

DEUTSCH

Implementieren Sie zwei Funktionen `scale_up` und `scale_down`. Beide Funktionen nehmen einen Integer `factor` und einen String `image` als Parameter an. `image` repräsentiert ein 2-dimensionales Bild bestehend aus ASCII-Symbolen. Die Funktion `scale_up` soll das Bild um den angegebenen Faktor vergrössern, und `scale_down` soll das Bild um den angegebenen Faktor verkleinern. Um ein Bild um einen Faktor *n* zu vergrössern wiederholt man einfach die einzelnen Zeichen und Zeilen *n* mal. Um ein Bild um einen Faktor *n* zu verkleiner, behält man nur diejenigen Zeichen und Zeilen, deren Indices durch *n* teilbar sind. Beachten Sie die Assertions als Beispiele für die Anwendung von `scale_up` und `scale_down`. Korrekt implementiert, gibt jede der zwei Funktionen jeweils die Hälfte der vollen Punktzahl.

ENGLISH

Implement two functions `scale_up` and `scale_down`. Both functions take an integer `factor` and a string `image` as parameters. `image` represents a 2-dimensional image made up from ASCII symbols. The function `scale_up` should scale up the image by the given factor, and `scale_down` should scale down the image by the given factor. To increase the size of an image by a factor *n*, each character and line is simply repeated *n* times. To decrease the size of an image by a factor *n*, only those characters and lines, whose indices are divisible by *n* are retained. Consider the assertions given below as examples for using `scale_up` and `scale_down`.

```
def scale_up(factor, image):
    pass

def scale_down(factor, image):
    pass

# DO NOT SUBMIT THE LINES BELOW!
#img1 = ("xxx\n"
#        "x x\n"
#        "xxx")
#
#img2 = ("xxxxxx\n"
#        "xxxxxx\n"
#        "xx  xx\n"
#        "xx  xx\n"
#        "xxxxxx\n"
#        "xxxxxx")
#
#assert scale_up(2, img1) == img2
#assert scale_down(2, img2) == img1
#
#img3 = ("123\n"
#        "345")
#
#img4 = ("111222333\n"
#        "111222333\n"
#        "111222333\n"
#        "333444555\n"
#        "333444555\n"
#        "333444555")
#
#assert scale_up(3, img3) == img4
#assert scale_down(3, img4) == img3
#
#img5 = ("12345\n"
#        "abcde\n"
#        "ABCDE")
#
#img6 = ("14")
#
#assert scale_down(3, img5) == img6
```

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

Lösung / Solution

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

def scale_up(factor, image):
    canvas = []
    image = image.split("\n")
    for line in image:
        new_line = []
        for character in line:
            new_line.append(character * factor)
        new_line = "".join(new_line)
        for i in range(factor):
            canvas.append(new_line)
    return "\n".join(canvas)

def scale_down(factor, image):
    canvas = []
    image = image.split("\n")
    for y, line in enumerate(image):
        new_line = []
        for x, character in enumerate(line):
            if x % factor == 0:
                new_line.append(character)
        new_line = "".join(new_line)
        if y % factor == 0:
            canvas.append(new_line)
    return "\n".join(canvas)
```

Classes

Gefragt / Asked: 1

Classes: brewery

Punkte / Points: 17

Aufgabe / Task

DEUTSCH

Implementieren Sie eine Klasse `Brewery` die eine Brauerei repräsentiert. Eine Brauerei hat einen Namen, der im Konstruktor übergeben wird. Ausserdem hat eine Brauerei einen Lagerbestand an verschiedenen Rohstoffen. Die Brauerei startet ohne Lagerbestand. Die Klasse soll folgende Methoden implementieren:

- `to_gram` soll eine statische Methode sein, die Werte in Gramm, Kilogramm oder Tonnen in Grammwerte umwandelt. Sie soll eine Menge als Integer und eine Einheit (entweder Tonnen "t", Kilogramm "kg" oder Gramm "g") als String annehmen, und den entsprechenden Grammwert als Integer zurückgeben.
- `add_stock` soll es ermöglichen, beliebig benannte Rohstoffe einzulagern. Die Methode soll einen Rohstoffnamen als String, die Menge als Integer und die Einheit als String (wie oben) annehmen. Alle Rohstoffe sollen intern in Gramm verarbeitet und abgespeichert werden. Sie können die statische Methode `to_gram` verwenden um für die einzulagernden Rohstoffe den Grammwert zu berechnen.
- `show_stock` soll für den als String übergebenen Rohstoffnamen den Lagerbestand als Integer zurückgeben. Wenn nichts von diesem Rohstoff im Lager ist, soll die Zahl `0` zurückgegeben werden.
- `brew` nimmt ein Rezept in Form eines Dictionary als Parameter an. Das Dictionary enthält als Keys die Namen von Rohstoffen und als dazugehörige Values die jeweils im Rezept benötigte Menge in Gramm. Wenn `brew` aufgerufen wird und genügend Rohstoffe eingelagert sind, um das Getränk herzustellen, sollen die benötigten Mengen aus dem Lagerbestand entfernt werden. Sind nicht alle Rohstoffe ausreichend an Lager, so soll der Lagerbestand unverändert bleiben und ein LookupError verursacht werden werden.

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

ENGLISH

Implement a class `Brewery` which represents a brewery. A brewery has a name which can be passed to its constructor. Furthermore, the brewery has a stock of various resources. The brewery starts without any resources. The class should implement the following methods:

- `to_gram` should be a static method, which converts a value in grams, kilograms or metric tons into a gram value. It should take an amount as an integer and the unit (either metric tons "t", kilograms "kg", or grams "g") as a string, and return the gram value as an integer.
- `add_stock` should make it possible to add arbitrarily named resources to the storehouse. The method should take the name of a resource as a string, the amount as an integer, and the unit as a string (like above). All resources should be internally measured and stored in grams. You can use the `to_gram` method to calculate the gram values for the resources to be stocked.
- `show_stock` should take a resource name as a string and return the stock as an integer. If there isn't any stock in the storehouse for the requested resource, it should return the number `0`
- `brew` takes a recipe in form of a dictionary as a parameter. The dictionary contains the names of resources as keys and the gram amounts required in the recipe as the corresponding values. If `brew` is called and enough resources are in stock to produce the beverage, the required amounts should be removed from the stock. If not all resources are sufficiently in stock, the stock should not be modified and a LookupError should be raised.

Consider the assertions given below as examples for using `Event`.

```
class Brewery:
    pass

# DO NOT SUBMIT THE LINES BELOW!
#assert Brewery.to_gram(1, "t") == 1000000
#assert Brewery.to_gram(1, "kg") == 1000
#assert Brewery.to_gram(1, "g") == 1
#b = Brewery("KegOverflow")
#assert b.show_stock("Syrup") == 0
#b.add_stock("Malt", 5, "kg")
#b.add_stock("Malt", 5, "kg")
#b.add_stock("Water", 50, "kg")
#b.add_stock("Hops", 30, "g")
#assert b.show_stock("Malt") == 10000
#b.brew({"Malt": 8000, "Water": 40000, "Hops": 20})
#assert b.show_stock("Malt") == 2000
#b.brew({"Water": 10000})
#assert b.show_stock("Water") == 0
```

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

Lösung / Solution

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

class Brewery:
    def __init__(self, name):
        self.name = name
        self.__stock = {}

    @staticmethod
    def to_gram(amount, src_unit):
        factor = 0
        if src_unit == "kg":
            return amount * 1000
        if src_unit == "t":
            return amount * 1000 * 1000
        return amount

    def add_stock(self, resource, amount, unit):
        if resource not in self.__stock:
            self.__stock[resource] = 0
        grams = Brewery.to_gram(amount, unit)
        self.__stock[resource] += grams

    def show_stock(self, resource):
        if resource not in self.__stock:
            return 0
        return self.__stock[resource]

    def brew(self, recipe):
        for resource, amount in recipe.items():
            if (resource not in self.__stock or
                self.__stock[resource] < amount):
                raise LookupError
        for resource, amount in recipe.items():
            self.__stock[resource] -= amount
            if self.__stock[resource] == 0:
                del(self.__stock[resource])
```

Classes: event

Punkte / Points: 17

Aufgabe / Task

DEUTSCH

Implementieren Sie eine Klasse `Event`, die einen Anlass wie zum Beispiel ein Fussballspiel oder eine Theateraufführung repräsentiert. Im Konstruktor soll `Event` einen Namen als String annehmen und einen Integer, der angibt, wieviele Plätze es an diesem Anlass gibt. Die Sitzplätze sind dann entsprechend von 1 bis zum Maximalwert durchnummeriert. Die Klasse soll folgende Methoden implementieren:

- `enter` soll zwei Parameter annehmen: eine ganzzahlige Sitzplatznummer und einen String. Ein Aufruf dieser Methode bewirkt, dass der entsprechende Sitzplatz von einer Person besetzt wird. Falls ein Platz bereits besetzt ist, soll ein NameError verursacht werden. Falls die Platznummer kleiner als 1 oder grösser als die letzte Platznummer ist, soll ein IndexError verursacht werden.
- `get` soll eine Sitzplatznummer als Parameter annehmen, und zurückgeben, wer den entsprechenden Sitzplatz besetzt. Sitzt dort niemand, soll `None` zurückgegeben werden. Falls die Platznummer kleiner als 1 oder grösser als die letzte Platznummer ist, soll ein IndexError verursacht werden.
- `occupied` soll keine Parameter annehmen und die Anzahl besetzter Sitzplätze zurückgeben.
- `empty` soll keine Parameter annehmen und die Anzahl freier Sitzplätze zurückgeben.

Des weiteren soll es möglich sein, zwei Events mittels den Vergleichsoperatoren <, >, und == zu vergleichen. Hierfür müssen Sie die Methoden `__lt__(self, other)`, `__gt__(self, other)`, und `__eq__(self, other)` implementieren. Ein Event `a` ist grösser als ein Event `b`, wenn die Anzahl *besetzter* Plätze in `a` grösser ist als in `b`. Instanzattribute sollen privat sein, aber es soll möglich sein, den Namen des Anlasses via einer Methode `get_name` abzurufen. Beachten Sie die Assertions als Beispiele für die Anwendung von `Event`.

ENGLISH

Implement a class `Event` which represents an event like a soccer game or a theater show. The constructor should take an integer which indicates, how many seats are available at the event. The seats are numbered starting at 1 up to the maximum value. The class should implement the following methods:

- `enter` should take two parameters: an integer seat number and a string. Calling this method occupies the given seat by a person. If the seat is already occupied, a NameError should be raised. If the seat number is smaller than 1 or larger than the last seat number, and IndexError should be raised.
- `get` should take a seat number and return who's sitting on the corresponding seat. If there's nobody sitting there, `None` should be returned. If the seat number is smaller than 1 or larger than the last seat number, and IndexError should be raised.
- `occupied` should take no parameters and return the number of occupied seats.
- `empty` should take no parameters and return the number of empty seats.

Furthermore, it should be possible to compare events using the comparison operators <, >, and ==. For this, you need to implement the methods `__lt__(self, other)`, `__gt__(self, other)`, und `__eq__(self, other)`. An event `a` is larger than an event `b`, if the number of *occupied* seats in `a` is larger. Instance attributes should be private, but it should be possible to retrieve the name of an event via a method `get_name`. Consider the assertions given below as examples for using `Event`.

```
class Event:
    pass

# DO NOT SUBMIT THE LINES BELOW!
#e1 = Event(150)
#e1.enter(45, "Alice")
#assert e1.get(45) == "Alice"
#e1.enter(42, "Bob")
#assert e1.occupied() == 2
#assert e1.empty() == 148
#e2 = Event(40)
#assert e2.get(40) == None
#e2.enter(1, "Andrea")
#e2.enter(2, "Beatrice")
#assert e2 == e1
#e2.enter(20, "Charly")
#assert e2 > e1
```

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

Lösung / Solution

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

# This solution adheres to the task description including a name parameter.
# Because the assertions for the Event constructor are faulty in the task,
# omitting the name parameter and get_name method is also an accepted solution.
class Event:
    def __init__(self, name, seats):
        self.__name = name
        self.__seats = [None] * seats

    def get_name(self):
        return self.__name
```

```
    def enter(self, seat_no, name):
        if seat_no < 1 or seat_no > len(self.__seats):
            raise IndexError
        if self.__seats[seat_no-1]:
            raise NameError
        self.__seats[seat_no-1] = name

    def get(self, seat_no):
        if seat_no < 1 or seat_no > len(self.__seats):
            raise IndexError
        return self.__seats[seat_no-1]

    def occupied(self):
        return sum(1 for s in self.__seats if s)

    def empty(self):
        return len(self.__seats) - self.occupied()

    def __lt__(self, other):
        return self.occupied() < other.occupied()

    def __gt__(self, other):
        return self.occupied() > other.occupied()

    def __eq__(self, other):
        return self.occupied() == other.occupied()
```

Classes: school

Punkte / Points: 17

Aufgabe / Task

DEUTSCH

Implementieren Sie zwei Klassen `Student` und `School`. Die Klasse `Student` repräsentiert eine/n Schüler/in, die/der einen Namen und ein Schuljahr hat. Der Name soll im Konstruktor als String übergeben werden; ein/e neue/r Schüler/in beginnt immer im Schuljahr Nummer 1. `Student` soll folgende Methode implementieren:

- `learn` soll keine Parameter annehmen und das Schuljahr des Schülers / der Schülerin um 1 erhöhen.
- Via zwei Methoden `get_name` und `get_year` soll man den Namen und das Schuljahr abrufen können.

Die Klasse `School`. repräsentiert eine Schule. Eine Schule hat einen Namen und kann Schüler/innen bestimmter Schuljahre unterrichten. `School` soll dementsprechend im Konstruktor einen String als Namen annehmen, und eine Liste von Integer, die angibt, welche Jahrgänge unterrichtet werden können. Des weiteren speichert die Schule ab, wie oft ein Schüler erfolgreich unterrichtet wurde (bei der Instanzierung 0). Zusätzlich soll in einer Klassenvariabel `national_taught` erfasst werden, wie oft insgesamt erfolgreich in allen Schulen unterrichtet wurde (zum Anfang 0). `School` soll folgende Methode implementieren:

- `educate` soll ein Objekt vom Typ `Student` als Parameter annehmen. Diese Methode soll folgende Funktionalität haben:
  - Wenn der/die Schüler/in einem Jahrgang angehört, den die Schule nicht unterrichten kann, soll ein ValueError verursacht werden.
  - Ansonsten soll `educate` mit einer Wahrscheinlichkeit von 9/10 die `learn`-Methode des übergebenen Schülers aufrufen. Falls dies passiert, soll die Anzahl erfolgreicher Lehrversuche in der Schule und auch in der Klassenvariabel um 1 erhöht werden.
- `get_taught` soll die Anzahl erfolgreicher Lehrversuche der Schule zurückgeben.

Beachten Sie die Assertions als Beispiele für die Anwendung von `Student` und `School`.

ENGLISH

Implement two classes `Student` and `School`. The class `Student` represents a student which has a name and a school year. The name should be passed to the constructor as a string; a new student always starts in school year number 1. `Student` should implement the following methods:

- `learn` should take no parameters and increase the school year of the student by 1.
- Two methods `get_name` and `get_year` should return the name and school year.

The class `School` represents a school. A school has a name and can educate students of specific school years. As such, the constructor of `School` should take the name as a string, and a list of integers which indicates which school years the school can teach. Furthermore, the school keeps track of how often any student has successfully been educated (0 at instantiation). Additionally, a class variable `national_taught` should store how many times in total a student has been successfully educated across all schools (0 at the beginning). `School` should implement the following methods:

- `educate` should take an object of type `Student` as a parameter. The method should implement the following functionality:
  - If the student belongs to a school year that cannot be taught at this school, a ValueError should be raised.
  - Otherwise, `educate` should call the `learn` method of the student with a probability of 9/10. If this happens, the number of successfully educated students at the school should be increased by 1 and the class variable should also be increased by 1.
- `get_taught` should return the number of successful education attempts at this school.

Consider the assertions given below as examples for using `Student` and `School`.

```
class Student:
    pass

class School:
    pass

# DO NOT SUBMIT THE LINES BELOW!
#a = Student("Ueli")
#assert a.get_name() == "Ueli"
#assert a.get_year() == 1
#s1 = School("Mätteliwise", [1,2,3,4,5,6])
#s2 = School("Blüemlihof", [1,2,3,4,5,6])
#assert s1.get_taught() == 0
## the following calls have random outcomes
#s1.educate(a)
#assert a.get_year() in [1, 2]
#assert s1.get_taught() in [0, 1]
#s2.educate(a)
#assert a.get_year() in [1, 2, 3]
#assert s2.get_taught() in [0, 1]
#assert s2.national_taught in [0, 1, 2]
```

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

Lösung / Solution

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

import random

class Student:
    def __init__(self, name):
        self.__name = name
        self.__year = 1

    def learn(self):
        self.__year += 1

    def get_year(self):
        return self.__year

    def get_name(self):
        return self.__name

class School:
    national_taught = 0
    def __init__(self, name, years):
        self.__name = name
        self.__level = years
        self.__taught = 0

    def educate(self, student):
        if student.get_year() not in self.__level:
            raise ValueError
        if random.randint(0, 9) > 0:
            student.learn()
            self.__taught += 1
            School.national_taught += 1
```

```
    def get_taught(self):
        return self.__taught
```

Testing

Gefragt / Asked: 1

Testing: are_anagrams

Punkte / Points: 7

Aufgabe / Task

DEUTSCH

Implementieren Sie eine Testklasse namens `MyTestSuite`, die eine unbekannte Implementation einer Funktion `are_anagrams(x, y)` nach dem Blackbox-Testing-Prinzip testet. Die Funktion ist wie folgt spezifiziert:

`are_anagrams` nimmt zwei Strings als Parameter `a` und `b` and. Die Funktion bestimmt, ob `a` und `b` Anagramme sind und wenn ja, wird `True` zurückgegeben, ansonsten `False`. Wenn die Buchstaben in `a` verschoben werden können um `b` zu erhalten, ohne Buchstaben hinzuzufügen oder zu entfernen, handelt es sich um Anagramme. Dabei werden nicht-alphanumerische Zeichen so wie Gross- und Kleinschreibung ignoriert. Wenn `a` oder `b` kein String ist, wird ein TypeError verursacht. Die folgenden Assertions illustrieren die Anwendung von `are_anagrams`

```
assert are_anagrams('Dog', 'God')
assert are_anagrams("The Meaning of Life.", "The fine game of nil!")
assert (not are_anagrams("The Meaning of Life", "Work"))
```

Sie müssen ungefähr 7 Testmethoden implementieren, um einzelne Abweichung von dieser Spezifikation isoliert zu testet. Bitte rufen Sie die Funktion in ihren Test-Methoden direkt und ohne Modulangabe auf, also zum Beispiel so:

```
self.assertTrue(are_anagrams("Dog", "God"))
```

Fügen Sie keine top-level statements, wie zum Beispiel Imports hinzu. Ihre Lösung darf nur `import unittest` und Ihre Implementation der Testklasse `MyTestSuite`, beinhalten.

ENGLISH

Implement a test class named `MyTestSuite`, which tests and unknown implementation of a function `are_anagrams(x, y)` via the blackbox testing principle. The function is specified as follows:

`are_anagrams` takes two strings as parameters `a` and `b`. The function determines if `a` and `b` are anagrams and returns `True` if yes, otherwise `False`. If the characters in `a` can be moved to form `b` without adding or removing characters, the two strings are Anagrams. Both non-alphanumerical symbols as well as casing are ignored. If `a` or `b` is not a string, a TypeError is raised. The following assertions illustrate the usage of `are_anagrams`

```
assert are_anagrams('Dog', 'God')
assert are_anagrams("The Meaning of Life.", "The fine game of nil!")
assert (not are_anagrams("The Meaning of Life", "Work"))
```

You should implement roughly 7 test methods to test individual deviations from this specification in an isolated manner. Please call the function in your test methods directly and without specifying a module, for example like this:

```
self.assertTrue(are_anagrams("Dog", "God"))
```

Do not add top-level statements, like imports. Your solution may only contain `import unittest` and your implementation of the test class `MyTestSuite`.

```
import unittest

class MyTestSuite(unittest.TestCase):
    pass
```

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

Lösung / Solution

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

import unittest

class MyTestSuite(unittest.TestCase):
    def test_empty(self):
        self.assertTrue(are_anagrams("", ""))

    def test_basic(self):
        self.assertTrue(are_anagrams("xxxyyz", "zyyxxx"))

    def test_with_nonalpha(self):
        self.assertTrue(are_anagrams("xxx, yy, z", "zyyxxx"))

    def test_with_mixed_casing(self):
        self.assertTrue(are_anagrams("xxXyya", "XXXYYA"))

    def test_false_different_letters(self):
        res = are_anagrams("xxxyya", "zyyxxx")
        self.assertFalse(res)

    def test_false_different_counts(self):
        res = are_anagrams("xxxyy", "yyyxx")
        self.assertFalse(res)

    def test_rejects_non_strings(self):
        with self.assertRaises(TypeError):
            are_anagrams(True, "hoot")
        with self.assertRaises(TypeError):
            are_anagrams("hoot", True)
```

Testing: dict_to_lists

Punkte / Points: 7

Aufgabe / Task

DEUTSCH

Implementieren Sie eine Testklasse namens `MyTestSuite`, die eine unbekannte Implementation einer Funktion `dict_to_lists(d)` nach dem Blackbox-Testing-Prinzip testet. Die Funktion ist wie folgt spezifiziert:

`dict_to_lists` nimmt ein dictionary als Parameter `d` an. Die Funktion gibt zwei Werte zurück: eine Liste der Keys in `d`, und eine Liste der dazugehörigen Werte in `d`. Die zwei Listen sind so sortiert, dass die Werte der ersten Liste aufsteigend sortiert sind, und die entsprechenden Werte in der zweiten Liste in derselben Reihenfolge vorliegen, sodass für jeden ehemaligen Key in der ersten Liste der entsprechende ehemalige Wert in der zweiten Liste am gleichen Index gefunden werden kann. Wenn `d` kein Dictionary ist, wird ein TypeError verursacht. Die folgenden Assertions illustrieren die Anwendung von `dict_to_lists`

```
assert dict_to_lists({"b": 1, "a": 2}) == (["a", "b"], [2, 1])
assert dict_to_lists({"a": 20, "z": 1}) == (["a", "z"], [20, 1])
```

Sie müssen ungefähr 7 Testmethoden implementieren, um einzelne Abweichung von dieser Spezifikation isoliert zu testet. Bitte rufen Sie die Funktion in ihren Test-Methoden direkt und ohne Modulangabe auf, also zum Beispiel so:

```
l1, l2 = dict_to_lists({"a": 20, "z": 1})
self.assertTrue(len(l1) == len(l2))
```

Fügen Sie keine top-level statements, wie zum Beispiel Imports hinzu. Ihre Lösung darf nur

`import unittest` und Ihre Implementation der Testklasse `MyTestSuite`, beinhalten.

ENGLISH

Implement a test class named `MyTestSuite`, which tests and unknown implementation of a function `dict_to_lists(d)` via the blackbox testing principle. The function is specified as follows:

`dict_to_lists` takes a dictionary as parameter `d`. The function returns two values: a list of keys in `d` and a list of corresponding values in `d`. The two lists are sorted such that the values in the first list are sorted in ascending order, and the values in the second list follow the same sequence, such that for every former key in the first list, its former value can be found in the second list at the same index. If `b` is not a dictionary, a TypeError is raised. The following assertions illustrate the usage of `dict_to_lists`

```
assert dict_to_lists({"b": 1, "a": 2}) == (["a", "b"], [2, 1])
assert dict_to_lists({"a": 20, "z": 1}) == (["a", "z"], [20, 1])
```

You should implement roughly 7 test methods to test individual deviations from this specification in an isolated manner. Please call the function in your test methods directly and without specifying a module, for example like this:

```
l1, l2 = dict_to_lists({"a": 20, "z": 1})
self.assertTrue(len(l1) == len(l2))
```

Do not add top-level statements, like imports. Your solution may only contain `import unittest` and your implementation of the test class `MyTestSuite`.

```
import unittest

class MyTestSuite(unittest.TestCase):
    pass
```

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

Lösung / Solution

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

import unittest

from correct.correct import dict_to_lists

class MyTestSuite(unittest.TestCase):
    def test_empty(self):
        self.assertEqual(dict_to_lists({}), ([], []))

    def test_keys(self):
        l1, l2 = dict_to_lists({2: "b", 1: "a"})
        self.assertEquals(l1, [1,2])

    def test_values(self):
        l1, l2 = dict_to_lists({2: "b", 1: "a"})
        self.assertEquals(l2, ["a", "b"])

    def test_return_types(self):
        l1, l2 = dict_to_lists({2: "b", 1: "a"})
        self.assertTrue(isinstance(l1, list))
        self.assertTrue(isinstance(l2, list))

    def test_keys_sorted(self):
        d = {}
        for i in [5,1,2,3,7,6,9]:
            d[i] = -i
        l1, l2 = dict_to_lists(d)
        self.assertEqual(l1, sorted(l1))

    def test_values_correspond_to_keys(self):
        d = {3: "b", 2: "a", 1: "c"}
        l1, l2 = dict_to_lists(d)
        for i, e in enumerate(l1):
            self.assertEqual(d[e], l2[i])

    def test_rejects_non_dicts(self):
        with self.assertRaises(TypeError):
            dict_to_lists(True)
```

Testing: is_palindrome

Punkte / Points: 7

Aufgabe / Task

DEUTSCH

Implementieren Sie eine Testklasse namens `MyTestSuite`, die eine unbekannte Implementation einer Funktion `is_palindrome(x)` nach dem Blackbox-Testing-Prinzip testet. Die Funktion ist wie folgt spezifiziert:

`is_palindrome` nimmt einen Strings als Parameter `x` und bestimmt, ob es sich dabei um ein Palindrom handelt. Wenn ja, wird `True` zurückgegeben, ansonsten `False`. Ein Palindrom ist eine Zeichensequenz, die vorwärts wie rückwärts gleich gelesen werden kann. Nicht-alphanumerische Zeichen so wie Gross- und Kleinschreibung werden ignoriert. Wenn `d` kein String ist, wird ein TypeError verursacht. Wenn `d` weniger als 3 Zeichen lang ist, wird ein ValueError verursacht. Die folgenden Assertions illustrieren die Anwendung von `is_palindrome`

```
assert is_palindrome("maoam")
assert is_palindrome("Was it a car or a cat I saw?")
assert (not is_palindrome("I don't know"))
```

Sie müssen ungefähr 7 Testmethoden implementieren, um einzelne Abweichung von dieser Spezifikation isoliert zu testet. Bitte rufen Sie die Funktion in ihren Test-Methoden direkt und ohne Modulangabe auf, also zum Beispiel so:

```
self.assertTrue(is_palindrome("sugus"))
```

Fügen Sie keine top-level statements, wie zum Beispiel Imports hinzu. Ihre Lösung darf nur `import unittest` und Ihre Implementation der Testklasse `MyTestSuite`, beinhalten.

ENGLISH

Implement a test class named `MyTestSuite`, which tests an unknown implementation of a function `is_palindrome(x)` via the blackbox testing principle. The function is specified as follows:

`is_palindrome` takes a string as parameter `x` and determines if it is a palindrome. It returns `True` if yes, otherwise `False`. A palindrome is a word that can be read the same forwards and backwards. If `x` is not a string, a TypeError is raised. If `x` is less than 3 characters long, a ValueError is raised. The following assertions illustrate the usage of `is_palindrome`

```
assert is_palindrome("maoam")
assert is_palindrome("Was it a car or a cat I saw?")
assert (not is_palindrome("I don't know"))
```

You should implement roughly 7 test methods to test individual deviations from this specification in an isolated manner. Please call the function in your test methods directly and without specifying a module, for example like this:

```
self.assertTrue(is_palindrome("sugus"))
```

Do not add top-level statements, like imports. Your solution may only contain `import unittest` and your implementation of the test class `MyTestSuite`.

```
import unittest

class MyTestSuite(unittest.TestCase):
    pass
```

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

Lösung / Solution

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

import unittest

class MyTestSuite(unittest.TestCase):
    def test_len_3(self):
        self.assertTrue(is_palindrome("bob"))
```

```
    def test_len_less_than_3_raises_ValueError(self):
        with self.assertRaises(ValueError):
            self.assertTrue(is_palindrome("aa"))

    def test_basic(self):
        self.assertTrue(is_palindrome("atoyota"))

    def test_with_nonalpha(self):
        self.assertTrue(is_palindrome("a toyota!"))

    def test_with_mixed_casing(self):
        self.assertTrue(is_palindrome("AToyoTa"))

    def test_false(self):
        self.assertFalse(is_palindrome("Honda"))

    def test_rejects_non_string(self):
        with self.assertRaises(TypeError):
            is_palindrome(True)
```

Inheritance

Gefragt / Asked: 1

Inheritance: 000

Punkte / Points: 24

Aufgabe / Task

Points: 24

English version below the code template.

DEUTSCH

Sie wurden beauftragt, das Personalsystem für ein Filmstudio zu entwickeln. Die Aufgabenstellung hat drei Teile:

- Eine Spezifikation der relevanten Konzepte und Prozesse.
- Komplementäre Implementationshinweise.
- Ein Code-Template mit vordefinierten Klassen, die implementiert werden müssen.

Bitte lesen Sie alle drei Teile sorgfältig für ein umfassendes Verständnis der Spezifikation. Wenn Sie ein klares Bild davon haben, was zu tun ist, implementieren Sie die nötigen Klassen und Tests.

Schauen Sie auf die Uhr oder stellen Sie einen Wecker! Rechnen Sie unbedingt genug Zeit ein, um Ihre Lösung in das Antwortfeld zu kopieren und einzureichen, bevor die Prüfung endet! Nach dieser Aufgabe folgen noch 2 Multiple-Choice-Fragen. Sie können die Aufgabe auch abgeben, wenn Sie nicht vollständig gelöst ist. Stellen Sie aber unbedingt sicher, dass ihre Abgabe keine Syntax- oder anderen grundlegenden Fehler enthält.

Spezifikation

Ein Filmstudio (`Studio`) kann zwei Arten von Schauspielern (`Actor`) anstellen. Jede/r Schauspieler/in hat einen Namen. Darsteller (`Lead`) sind Schauspieler mit einer Festanstellung. Darsteller haben einen fixen Jahreslohn. Ausserdem soll jede/r Darsteller/in mit einer einzigartigen ID, einem aufsteigenden Integer, beginnend bei 0 für die/den erste/n Darsteller/in, erfasst werden. Extras (`Extra`) sind Schauspieler, die auf Stundenbasis angestellt sind. Sie haben einen Studenlohn und arbeiten eine fixe Anzahl Stunden pro Monat. Extras erhalten einen Mindestlohn. Extras haben keine einzigartige ID.

Für alle Arten von Schauspielern soll es möglich sein, das Monatsgehalt herauszufinden. Für Darsteller berechnet sich der Monatslohn aus dem Jahresgehalt geteilt durch die Anzahl

Monate. Für Lohnzahlungen hat ein Jahr 12 Monate. Für Extras berechnet sich der Monatslohn aus dem Stundenlohn multipliziert mit der Anzahl Arbeitsstunden pro Monat.

Wenn man ein/e Darsteller/in mittels Python's `print`-Funktion ausgibt, soll die Kommandozeile einen String in der Form "Salary: X (UNIQUE_ID)" ausgeben, wo X der Monatslohn und UNIQUE_ID die einzigartige Identifikationsnummer ist. Wenn ein Extra geprintet wird, soll ein String in der Form "Salary: X (temp)" ausgegeben werden, wo X für den Monatslohn steht.

Ein Filmstudio hat einen Namen und eine belibige Anzahl Schauspieler. Es soll möglich sein, dem Filmstudio Schauspieler hinzuzufügen. Ausserdem soll man die gesamten monatlichen Anstellungskosten für das Filmstudio berechnen können, welche sich einfach aus der Summe der Monatsgehälter aller Schauspieler berechnet.

Implementationshinweise:

Vervollständigen Sie die fehlenden Klassenimplementationen in der vorgegebenen Vorlage. Fügen Sie keine top-level-Definitionen hinzu! Beachten Sie folgendes:

- Die vorgebenen Assertions enthalten essenzielle Hinweise darauf, wie die nötigen Objekte sich verhalten sollen (insbesondere bezüglich Konstruktor- und Methodenparameter).
- Spezifizieren Sie wo nötig Superklassen.
- Fügen Sie wo nötig Konstruktoren, Klassenvariabeln und Instanzattribute hinzu.
- Wenn man versucht, ein Extra mit einem Stundenlohn von weniger als 9.4 zu erstellen, soll ein `ValueError` verursacht werden.
- Wenn man versucht ein/e Schauspieler/in dem Filmstudio hinzuzufügen, obwohl diese/r schon hinzugefügt wurde, soll ein `ValueError` verursacht werden.
- Alle Datenattribute von Schauspielern sollten privat sein, aber es soll möglich sein, den Namen von Schauspielern und die einzigartige ID von Darstellern mittels Methoden herauszufinden. Der Name eines Filmstudios soll public sein, aber jeglicher interner Zustand für die Speicherung von Schauspielern soll privat sein.
- Fügen Sie die nötigen Methoden hinzu und benennen Sie diese gemäss den Assertions.
- Wo eine Klasse von einer anderen erbt, soll gemeinsame Funktionalität in der Basisklasse platziert werden, anstatt sie doppelt zu schreiben.

Template

- Fügen Sie keine zusätzlichen Klassen hinzu! Der top-level Scope Ihrer Abgabe darf NUR Klassendefinitionen und Importstatements enthalten, sonst nichts!
- Ändern Sie nicht die vorgegebenen Klassen- und Methodennamen!
- Ihre Lösung muss mit den vorgegebenen Assertions kompatibel sein; achten Sie also darauf, wie Sie ihre Methoden benennen!

```
from abc import ABC, abstractmethod

class Actor:
    pass

class Lead:
    pass

class Extra:
    pass

class Studio:
    pass

# DO NOT SUBMIT THE LINES BELOW!
#e = Studio("Warmer Sisters")
#i1 = Lead("Bob", 60000) # yearly salary
#i2 = Lead("Alice", 75000) # yearly salary
#i3 = Extra("Taylor", 21.50, 15) # hourly salary, hours per month
#assert i1.get_name() == "Bob"
#assert i3.get_name() == "Taylor"
#assert i1.get_id() == 0
#assert i2.get_id() == 1
#assert i1.get_monthly_salary() > 4000
#assert i3.get_monthly_salary() == 322.50
#e.add_actor(i1)
#e.add_actor(i2)
#e.add_actor(i3)
#assert e.get_monthly_staff_cost() > 9000
```

ENGLISH

You've been tasked with programming a human resources system for a movie studio. This task description consists of three parts:

- A specification of the relevant concepts and processes.
- Complementary implementation instructions.

- A code template (above) containing pre-defined classes which have to be implemented.

Please read all three parts carefully for a comprehensive specification. Once you have a clear picture of what to do, implement the necessary classes. Finally, submit your solution.

Watch the clock or set an alarm! You must ensure that you have enough time to copy and submit your solution, before the exam ends! After this task, there are two more multiple choice questions. You can hand in your solution even if it is incomplete. Just make sure that your submission does not contain syntax- or other basic errors.

Conceptual description

The movie studio (`Studio`) can employee two different kinds of actors (`Actor`). Every actor has a name. Leads (`Lead`) are actors with a permanent contract. Leads have a fixed annual salary. Furthermore, each new lead should be created with a unique ID, which is an integer, counting up, starting at zero for the first lead. Extras (`Extra`) are actors which are employed on an hourly basis. They have an hourly salary and work a fixed number of hours per month. Extras must receive a minimum wage. Extras do not have a unique ID.

For all kinds of actors it should be possible to get their monthly salary. For leads, the monthly salary is calculated by dividing the annual salary by the number of months. For salary payments, there are 12 months in a year. For extras, the monthly salary is calcualted by multiplying the hourly salary by the number of hours worked per month.

When printing a lead using Python's `print` function, the command line should display a string like "Salary: X (UNIQUE_ID)", where X denotes the monthly salary and UNIQUE_ID denotes the actor's unique identifier. When printing a extra, the command line should display "Salary: X (temp)", where X denotes the monthly salary.

A movie studio has a name and any number of actors. It should be possible to add actors to a movie studio. It should be possible to calculate the total monthly staff cost of a movie studio, which is simply the sum of all salaries of all actors for a month.

Implementation instructions:

Fill in the missing class implementations in the provided template. Do not add any other top-level definitions! Observing the following:

- The provided assertions contain essential information on how the required objects should behave (especially regarding constructor- and method parameters).
- Specify super classes where necessary.
- Add constructors, class variables and instance attributes where necessary.
- When trying to create a extra with an hourly salary less than 9.4, a `ValueError` should be raised.
- When trying to add a actor to a movie studio although it has already been added, a `ValueError` should be raised.
- All instance attributes of actors should be private, but it should be possible to get the name of a actor and the unique ID of a lead via methods. The name of a movie studio should be public, but any internal state to store the actors should be private.
- Add the necessary methods and make sure to name them correctly according to the assertions.
- Where one class inherits from another, ensure to put common functionality into the base class instead of duplicating it in sub classes.

Template

- Do not add additional classes! The top-level scope of your submission may ONLY contain class definitions and import statements, nothing else!
- Do not change the pre-defined class names!
- Your solution must be compatible with the pre-defined assertions; so take care when naming your methods!

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

Lösung / Solution

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

from abc import ABC, abstractmethod

class Actor(ABC):

    def __init__(self, name):
        self.__name = name

    def get_name(self):
```

```
        return self.__name

    @abstractmethod
    def get_monthly_salary(self):
        pass

class Lead(Actor):
    seq = 0
    def __init__(self, name, annual_salary):
        super().__init__(name)
        self.__annual_salary = annual_salary
        self.__id = Lead.seq
        Lead.seq += 1

    def get_id(self):
        return self.__id

    def get_monthly_salary(self):
        return self.__annual_salary / 12

    def __str__(self):
        return f"{self.get_name()}: {self.get_monthly_salary()} ({self.__id})"

class Extra(Actor):
    def __init__(self, name, hourly_salary, hours_per_month):
        super().__init__(name)
        if hourly_salary < 9.4:
            raise ValueError
        self.__hourly_salary = hourly_salary
        self.__hours_per_month = hours_per_month

    def get_monthly_salary(self):
        return self.__hourly_salary * self.__hours_per_month

    def __str__(self):
        return f"{self.get_name()}: {self.get_monthly_salary()} (temp)"

class Studio:
    def __init__(self, name):
        self.name = name
        self.__actors = []

    def add_actor(self, actor):
        if actor in self.__actors:
            raise ValueError
        self.__actors.append(actor)

    def get_monthly_staff_cost(self):
        res = 0
        for e in self.__actors:
            res += e.get_monthly_salary()
        return res
```

Final Questions

Gefragt / Asked: 2

Booleans

Punkte / Points: 2 Kprim

Frage / Question

EN: Entscheiden Sie, welche der folgenden Aussagen wahr sind und kreuzen Sie diese an.

DE: For each of the following statements, check the box if the statement is true.

Antworten / Answers

- EN: Diese expression evaluiert als True: `not False and True`

  DE: This expression evaluates as True: `not False and True`

  Correct

- EN: Diese expression evaluiert als True: `"1" == str(int(str(1)))`

  DE: This expression evaluates as True: `"1" == str(int(str(1)))`

  Correct

- EN: Diese expression evaluiert als True: `[] == {}`

  DE: This expression evaluates as True: `[] == {}`

  False

- EN: Diese expression evaluiert als True: `bool("False")`

  DE: This expression evaluates as True: `bool("False")`

  Correct

Dictionaries

Punkte / Points: 2 Kprim

Frage / Question

EN: Entscheiden Sie, welche der folgenden Aussagen wahr sind und kreuzen Sie diese an.

DE: For each of the following statements, check the box if the statement is true.

Antworten / Answers

- EN: Werte beliebigen Typs können als Keys in Dictionaries verwendet werden.

  DE: Values of any type can be used as keys in dictionaries.

  False

- EN: Werte beliebigen Typs können als Values in Dictionaries verwendet werden.

  DE: Values of any type can be used as values in dictionaries.

  Correct

- EN: In Bezug auf Dictionaries: Wenn ein Wert einem Key zugewiesen wird, der bereits einen Wert hat, passiert nichts.

  DE: Regarding dictionaries: when assigning a value to a key that already has a value, nothing happens.

  False

- EN: Daten in einem Dictionary haben eine undefinierte Reihenfolge.

  DE: The order of items in a dictionary is undefined.

  Correct

Conditionals / Loops

Punkte / Points: 2 Kprim

Frage / Question

EN: Entscheiden Sie, welche der folgenden Aussagen wahr sind und kreuzen Sie diese an.

DE: For each of the following statements, check the box if the statement is true.

Antworten / Answers

- EN: Auf jeden `if`-Block muss ein `else`-Block folgen.

  DE: An `if` block must always be followed with an `else` block.

  False

- EN: Die folgenden Werte werden als False gewertet, wenn man sie in einer Bedingung verwendet: `[]` `"0"` `{}` `"False"`.

  DE: If used in a condition, the following values are considered False: `[]` `"0"` `{}` `"False"`.

  False

- EN: Loops können verschachtelt werden.

  DE: Loops can be nested.

  Correct

- EN: Ein while loop kann eine beschränke Anzahl Male durchlaufen bevor das Python-Programm zum Absturz gezwungen wird.

  DE: A while loop can only run a limited number of times before the Python program is forced to crash.

  False

Lists

Punkte / Points: 2 Kprim

Frage / Question

EN: Entscheiden Sie, welche der folgenden Aussagen wahr sind und kreuzen Sie diese an.

DE: For each of the following statements, check the box if the statement is true.

Antworten / Answers

- EN: Listen können Werte beliebigen Typs enthalten.

  DE: Lists can contain values of arbitrary types.

  Correct

- EN: Es ist nicht möglich, Werte aus einer Liste zu löschen.

  DE: It's impossible to delete values from a list.

  False

- EN: Wenn man eine Liste kopiert, werden auch alle Elemente darin kopiert.

  DE: Copying a list also makes copies of all values contained.

  False

- EN: Listen kann man abändern.

  DE: Lists can be modified.

  Correct

print vs. return

Punkte / Points: 2 Kprim

Frage / Question

EN: Entscheiden Sie, welche der folgenden Aussagen wahr sind und kreuzen Sie diese an.

DE: For each of the following statements, check the box if the statement is true.

Antworten / Answers

- EN: Die print Funktion gibt einen String zurück.

  DE: The print function returns a string.

  False

- EN: Wenn eine Funktion keine return statements enthält, dann gibt sie None zurück.

  DE: If a function does not contain any return statements, then it returns None.

  Correct

- EN: Eine Funktion kann entweder print oder return statement enthalten, aber nicht beides.

  DE: A function can either contain print or return statements, but not both.

  False

- EN: Der Rückgabewert eines Aufrufs von print kann in einer Variable gespeichert werden.

  DE: The return value of a call to print can be stored in a variable.

  Correct

Indices

Punkte / Points: 2 Kprim

Frage / Question

EN: Entscheiden Sie, welche der folgenden Aussagen wahr sind und kreuzen Sie diese an.

DE: For each of the following statements, check the box if the statement is true.

Antworten / Answers

- EN: The index -1 zeigt auf das letzte Element in einer Liste.

  DE: The index -1 refers to the last element in a list.

  Correct

- EN: Wenn eine Liste `x` 5 Elemente beinhaltet, dann verursacht die Expression `x[5:]` einen IndexError.

  DE: If a list `x` has 5 elements, the expression `x[5:]` raises and IndexError.

  False

- EN: The range-Funktion kann Zahlen nur in aufsteigender Reihenfolge generieren.

  DE: The range function can only produce numbers in ascending order.

False

- EN: `[1,2,3][::-1]` ist eine gültige Expression.

  DE: `[1,2,3][::-1]` is a valid expression.

  Correct

Stack / Exceptions

Punkte / Points: 2 Kprim

Frage / Question

EN: Entscheiden Sie, welche der folgenden Aussagen wahr sind und kreuzen Sie diese an.

DE: For each of the following statements, check the box if the statement is true.

Antworten / Answers

- EN: Für jeden Funktionsaufruf wird auf dem Stack ein neuer Call-Frame erstellt.

  DE: For every function call, a new call frame is created on the stack.

  Correct

- EN: Auf einen `try` Block muss zwingend ein `except` Block folgen.

  DE: A `try` block must always be followed by a `except` block.

  False

- EN: Wird eine Exception nichts mittels `except` abgefangen, stürzt das Programm ab.

  DE: If an exception is not caught by an `except` block, the program crashes.

  Correct

- EN: Eine Funktion die eine Exception verursacht gibt keinen Wert zurück.

  DE: A function that raises an exception does not return a value.

  Correct