Datensammlungen in Python

Gesamtpunktzahl 145/155

Dieses Quiz konzentriert sich auf die verschiedenen Arten von Datensammlungen in Python und deren Verwendung. Du wirst zu Listen und ihren Methoden geprüft, die eine vielseitige und veränderbare Möglichkeit bieten, Daten zu speichern. Auch Tupel und ihre Unveränderlichkeit werden behandelt, sowie Dictionaries mit ihren Schlüssel-Wert-Paaren für strukturierte Daten. Das Quiz umfasst außerdem Sets und ihre speziellen Operationen wie Vereinigung und Schnittmenge. Zusätzlich werden verschiedene Techniken zur Iteration über diese unterschiedlichen Sammlungstypen abgefragt. Die Beherrschung dieser Datenstrukturen ist entscheidend für effiziente und elegante Python-Programme. Fülle das Quiz aus und gebe bitte deine Techstarter-Email an. Falls du eine Antwort nicht weißt, recherchiere die Begriffe oder verwende die Unterlagen, um dein Wissen zu vertiefen. Viel Erfolg! :)

E-Mail-Adresse *

alexander.manhardt@tn.techstarter.de

✓ Welche der folgenden Datensammlungen in Python ist geordnet und veränderbar (mutable)? *5/5

- ○ Tupel (tuple)
- ○ Dictionary (dict)
- ○ Set (set)
- ◉ Liste (list) ✓

Feedback

Korrekt! Listen in Python sind geordnete und veränderbare (mutable) Sammlungen. Das bedeutet, dass die Reihenfolge der Elemente beibehalten wird und Elemente nach der Erstellung hinzugefügt, geändert oder entfernt werden können. Tupel sind geordnet, aber unveränderlich (immutable); Dictionaries sind veränderbar, aber ungeordnet (in Python 3.7+ wird die Einfügereihenfolge beibehalten); Sets sind veränderbar, aber ungeordnet und erlauben keine Duplikate.

✓ Was ist die Ausgabe des folgenden Codes? my_list = [1, 2, 3]; my_list.append(4); print(my_list) *5/5

- (●) [1, 2, 3, 4] ✓
- ( ) [1, 2, 3]
- ( ) [4, 1, 2, 3]
- ( ) Error

Feedback

Korrekt! Die append()-Methode fügt ein Element am Ende einer Liste hinzu. In diesem Fall wird 4 am Ende der Liste [1, 2, 3] hinzugefügt, was zu [1, 2, 3, 4] führt.

✓ Welche der folgenden Aussagen über Tupel ist FALSCH? * 5/5

- ( ) Tupel werden mit runden Klammern () definiert
- ( ) Tupel sind unveränderlich (immutable)
- ( ) Tupel können nicht sortiert werden
- (●) Tupel-Elemente können nach der Erstellung geändert werden ✓

Feedback

Korrekt! Tupel sind unveränderlich (immutable), was bedeutet, dass ihre Elemente nach der Erstellung nicht geändert werden können. Dies ist der Hauptunterschied zu Listen. Tupel werden mit runden Klammern definiert, sind unveränderlich und können nicht direkt sortiert werden (obwohl man ein neues, sortiertes Tupel erstellen kann).

✓ Was ist die Ausgabe des folgenden Codes? my_dict = {'a': 1, 'b': 2}; print(my_dict['a']) *5/5

- ○ a
- ◉ 1 ✓
- ○ {'a': 1}
- ○ Error

Feedback

Korrekt! Der Code erstellt ein Dictionary mit den Schlüssel-Wert-Paaren 'a': 1 und 'b': 2. Der Ausdruck my_dict['a'] greift auf den Wert zu, der dem Schlüssel 'a' zugeordnet ist, nämlich 1.

✓ Welche Methode wird verwendet, um ein Element aus einer Liste zu entfernen, wenn der Index bekannt ist? *5/5

- ○ remove()
- ○ delete()
- ◉ pop() ✓
- ○ discard()

Feedback

Korrekt! Die pop()-Methode entfernt ein Element an einem bestimmten Index aus einer Liste und gibt es zurück. Wenn kein Index angegeben wird, wird das letzte Element entfernt. Die remove()-Methode entfernt das erste Vorkommen eines bestimmten Werts. delete() ist keine Listenmethode, und discard() ist eine Methode für Sets, nicht für Listen.

✓ Was ist die Ausgabe des folgenden Codes? my_set = {1, 2, 3, 2, 1}; print(my_set) *5/5

- ○ {1, 2, 3, 2, 1}
- ◉ {1, 2, 3} ✓
- ○ [1, 2, 3]
- ○ Error

Feedback

Korrekt! Sets in Python sind Sammlungen von eindeutigen Elementen. Beim Erstellen eines Sets werden Duplikate automatisch entfernt. Daher enthält my_set nur die eindeutigen Elemente {1, 2, 3}.

✓ Welche der folgenden Optionen ist ein gültiger Weg, um über ein Dictionary zu iterieren und sowohl die Schlüssel als auch die Werte zu erhalten? *5/5

- ○ for key in my_dict: print(key, my_dict[key])
- ○ for key, value in my_dict: print(key, value)
- ◉ for key, value in my_dict.items(): print(key, value) ✓
- ○ for item in my_dict: print(item[0], item[1])

Feedback

Korrekt! Die items()-Methode eines Dictionaries gibt eine Ansicht der Schlüssel-Wert-Paare als Tupel zurück. Mit der for key, value in my_dict.items()-Syntax können wir direkt über diese Paare iterieren und auf die Schlüssel und Werte zugreifen.

✓ Was ist die Ausgabe des folgenden Codes? my_tuple = (1, 2, 3); my_tuple[0] = 0; print(my_tuple) *5/5

- ○ (0, 2, 3)
- ○ (1, 2, 3)
- ○ (1, 0, 3)
- ◉ TypeError: 'tuple' object does not support item assignment ✓

Feedback

Korrekt! Tupel sind unveränderlich (immutable), was bedeutet, dass ihre Elemente nach der Erstellung nicht geändert werden können. Der Versuch, ein Element eines Tupels zu ändern (my_tuple[0] = 0), führt zu einem TypeError mit der Meldung 'tuple' object does not support item assignment.

✓ Welche Methode wird verwendet, um alle Elemente aus einer Liste zu entfernen? *5/5

- ◉ clear() ✓
- ○ remove_all()
- ○ delete()
- ○ empty()

Feedback

Korrekt! Die clear()-Methode entfernt alle Elemente aus einer Liste. Nach dem Aufruf von my_list.clear() wird die Liste leer sein: [].

✓ Was ist die Ausgabe des folgenden Codes? a = {1, 2, 3}; b = {3, 4, 5}; print(a | b) *5/5

- [ ] [1, 2, 3, 4, 5]
- [x] {1, 2, 3, 4, 5} ✓
- [ ] {1, 2, 4, 5}
- [ ] {3}

Feedback

Korrekt! Der Operator | führt eine Vereinigung (union) von zwei Sets durch. Das Ergebnis ist ein neues Set, das alle eindeutigen Elemente aus beiden Sets enthält. In diesem Fall sind die eindeutigen Elemente aus a und b: 1, 2, 3, 4 und 5.

✓ Welche der folgenden Methoden kann NICHT auf ein Tupel angewendet werden? *5/5

- [ ] count()
- [ ] index()
- [x] append() ✓
- [ ] len()

Feedback

Korrekt! Die append()-Methode ist eine Listenmethode und kann nicht auf Tupel angewendet werden, da Tupel unveränderlich sind. Die count()-Methode zählt, wie oft ein bestimmter Wert im Tupel vorkommt, index() gibt den Index des ersten Vorkommens eines Werts zurück, und len() ist eine eingebaute Funktion, die die Länge des Tupels zurückgibt.

✓ Was ist die Ausgabe des folgenden Codes? my_list = [1, 2, 3]; print(my_list * 2) *5/5

- (●) [1, 2, 3, 1, 2, 3] ✓
- ( ) [2, 4, 6]
- ( ) [1, 1, 2, 2, 3, 3]
- ( ) Error

Feedback

*Korrekt! Der Operator * mit einer Liste und einer Ganzzahl wiederholt die Liste. my_list * 2 erzeugt eine neue Liste, in der die Elemente von my_list zweimal wiederholt werden: [1, 2, 3, 1, 2, 3].*

✓ Welche der folgenden Optionen ist KEINE gültige Methode, um ein neues Element zu einem Dictionary hinzuzufügen? *5/5

- ( ) my_dict['new_key'] = 'new_value'
- (●) my_dict.add('new_key', 'new_value') ✓
- ( ) my_dict.update({'new_key': 'new_value'})
- ( ) my_dict.setdefault('new_key', 'new_value')

Feedback

Korrekt! Dictionaries haben keine add()-Methode. Um ein neues Element zu einem Dictionary hinzuzufügen, kann man direkt zuweisen (my_dict['new_key'] = 'new_value'), die update()-Methode verwenden oder setdefault(), das einen Wert nur setzt, wenn der Schlüssel noch nicht existiert.

✓ Was ist die Ausgabe des folgenden Codes? my_set = {1, 2, 3}; my_set.add(2); print(my_set) *5/5

- ○ [1, 2, 3, 2]
- ○ {1, 2, 3, 2}
- ◉ {1, 2, 3} ✓
- ○ Error

Feedback

Korrekt! Sets enthalten nur eindeutige Elemente. Wenn man versucht, ein Element hinzuzufügen, das bereits im Set vorhanden ist (wie 2 in diesem Fall), bleibt das Set unverändert. Daher bleibt my_set {1, 2, 3}.

✓ Welches der folgenden Elemente kann NICHT als Schlüssel in einem Dictionary verwendet werden? *5/5

- ○ Integer
- ○ String
- ◉ Liste ✓
- ○ Tupel

Feedback

Korrekt! In Python können nur unveränderliche (immutable) Typen als Dictionary-Schlüssel verwendet werden. Listen sind veränderlich (mutable) und können daher nicht als Schlüssel verwendet werden. Integer, Strings und Tupel (solange sie nur unveränderliche Elemente enthalten) können als Schlüssel verwendet werden.

✓ Was ist die Ausgabe des folgenden Codes? my_dict = {'a': 1, 'b': 2}; print('c' in my_dict) *5/5

- ○ True
- ◉ False ✓
- ○ None
- ○ KeyError

Feedback

Korrekt! Der Ausdruck 'c' in my_dict prüft, ob der Schlüssel 'c' im Dictionary my_dict vorhanden ist. Da 'c' nicht in my_dict enthalten ist, ist das Ergebnis False.

✗ Welche der folgenden Methoden wird verwendet, um die Elemente einer Liste in umgekehrter Reihenfolge zu durchlaufen? *0/5

- ○ list.invert()
- ○ reversed(list)
- ○ list.reverse()
- ◉ list[::-1] ✗

Richtige Antwort

- ◉ reversed(list)

Feedback

Falsch. Die eingebaute Funktion reversed() gibt ein Iterator-Objekt zurück, das die Elemente der Liste in umgekehrter Reihenfolge durchläuft. Alternativ kann man auch list[::-1] verwenden, um eine umgekehrte Kopie der Liste zu erstellen. list.reverse() kehrt die Liste an Ort und Stelle um, gibt aber nichts zurück. list.invert() existiert nicht.

✓ Was ist die Ausgabe des folgenden Codes? a = {1, 2, 3}; b = {3, 4, 5}; print(a & b) *5/5

- ○ []
- ○ {1, 2, 3, 4, 5}
- ◉ {3} ✓
- ○ {}

Feedback

Korrekt! Der Operator & führt einen Schnitt (intersection) von zwei Sets durch. Das Ergebnis ist ein neues Set, das nur die Elemente enthält, die in beiden Sets vorkommen. In diesem Fall ist das einzige gemeinsame Element 3, daher ist a & b = {3}.

✗ Welche der folgenden Optionen erzeugt ein leeres Dictionary? * 0/5

- ○ {}
- ◉ dict() ✗
- ○ []
- ○ ()

Richtige Antwort

- ◉ {}

Feedback

Falsch. Ein leeres Dictionary kann mit geschweiften Klammern {} oder mit der dict()-Konstruktorfunktion erstellt werden. [] erzeugt eine leere Liste und () ein leeres Tupel.

✓ Was ist die Ausgabe des folgenden Codes? my_tuple = (1, 2, 3); print(my_tuple.count(2)) *5/5

- (●) 1 ✓
- ( ) 2
- ( ) 3
- ( ) Error

Feedback

Korrekt! Die count()-Methode eines Tupels zählt, wie oft ein bestimmter Wert im Tupel vorkommt. In diesem Fall kommt der Wert 2 genau einmal im Tupel (1, 2, 3) vor, daher ist das Ergebnis 1.

✓ Welche der folgenden Optionen ist ein gültiger Weg, um eine Kopie einer Liste zu erstellen? *5/5

- ( ) new_list = my_list.copy()
- ( ) new_list = list(my_list)
- ( ) new_list = my_list[:]
- (●) Alle der oben genannten ✓

Feedback

Korrekt! Alle drei Optionen sind gültige Wege, um eine flache Kopie (shallow copy) einer Liste zu erstellen: die copy()-Methode, die list()-Konstruktorfunktion und das Slicing mit [:]. Jede dieser Methoden erzeugt eine neue Liste mit denselben Elementen wie die ursprüngliche Liste.

✓ Was ist die Ausgabe des folgenden Codes? my_list = [1, 2, 3]; my_list.insert(1, 4); print(my_list) *5/5

- (●) [1, 4, 2, 3] ✓
- ( ) [1, 2, 4, 3]
- ( ) [1, 2, 3, 4]
- ( ) [4, 1, 2, 3]

Feedback

Korrekt! Die insert()-Methode fügt ein Element an einer bestimmten Position in die Liste ein. Der erste Parameter gibt den Index an, an dem das Element eingefügt werden soll, und der zweite Parameter ist das einzufügende Element. In diesem Fall wird 4 an Index 1 (zwischen dem ersten und zweiten Element) eingefügt, was zu [1, 4, 2, 3] führt.

✓ Welche der folgenden Aussagen über Sets ist FALSCH? * 5/5

- ( ) Sets können nur eindeutige Elemente enthalten
- ( ) Sets sind ungeordnet
- ( ) Sets können mit geschweiften Klammern {} erstellt werden
- (●) Sets können verschachtelte Sets enthalten ✓

Feedback

Korrekt! Sets können keine veränderlichen (mutable) Objekte enthalten, daher können sie keine verschachtelten Sets enthalten, da Sets selbst veränderlich sind. Die anderen Aussagen sind korrekt: Sets enthalten nur eindeutige Elemente, sind ungeordnet und werden mit geschweiften Klammern erstellt.

✓ Was ist die Ausgabe des folgenden Codes? my_dict = {'a': 1, 'b': 2}; print(list(my_dict.keys())) *5/5

- (●) ['a', 'b'] ✓
- ( ) [1, 2]
- ( ) [('a', 1), ('b', 2)]
- ( ) {'a': 1, 'b': 2}

Feedback

Korrekt! Die keys()-Methode eines Dictionaries gibt eine Ansicht der Schlüssel zurück. Durch Umwandlung in eine Liste mit list() erhalten wir eine Liste der Schlüssel: ['a', 'b'].

✓ Welcher der folgenden Datentypen unterstützt Indizierung mit ganzen Zahlen? *5/5

- ( ) Dictionary
- ( ) Set
- (●) Liste ✓
- ( ) Alle der oben genannten

Feedback

Korrekt! Listen und Tupel unterstützen Indizierung mit ganzen Zahlen (z.B. my_list[0]), da sie geordnete Sammlungen sind. Dictionaries werden mit Schlüsseln indiziert (nicht notwendigerweise ganze Zahlen), und Sets unterstützen überhaupt keine Indizierung, da sie ungeordnet sind.

✓ Was ist die Ausgabe des folgenden Codes? a = {1, 2, 3}; b = {3, 4, 5}; print(a - b) *5/5

- [1, 2]
- {1, 2} ✓
- {1, 2, 4, 5}
- {1, 2, 3, 4, 5}

Feedback

Korrekt! Der Operator - führt eine Differenz von zwei Sets durch. Das Ergebnis ist ein neues Set, das alle Elemente enthält, die in a, aber nicht in b vorkommen. In diesem Fall sind das die Elemente 1 und 2, daher ist a - b = {1, 2}.

✓ Welche der folgenden Methoden wird verwendet, um ein bestimmtes Element aus einem Set zu entfernen und einen Fehler zu werfen, wenn das Element nicht vorhanden ist? *5/5

- remove() ✓
- discard()
- pop()
- delete()

Feedback

Korrekt! Die remove()-Methode entfernt ein bestimmtes Element aus einem Set und wirft einen KeyError, wenn das Element nicht im Set vorhanden ist. Die discard()-Methode entfernt ein Element, wenn es vorhanden ist, und tut nichts, wenn es nicht vorhanden ist. Die pop()-Methode entfernt und gibt ein beliebiges Element zurück. delete() ist keine Set-Methode.

✓ Was ist die Ausgabe des folgenden Codes? t1 = (1, 2); t2 = (3, 4); print(t1 + t2) *5/5

- [ ] [1, 2, 3, 4]
- [x] (1, 2, 3, 4) ✓
- [ ] ((1, 2), (3, 4))
- [ ] Error

Feedback

Korrekt! Der Operator + mit zwei Tupeln führt eine Verkettung durch. Das Ergebnis ist ein neues Tupel, das alle Elemente der beiden Tupel in der angegebenen Reihenfolge enthält: (1, 2, 3, 4).

✓ Welche der folgenden Methoden wird verwendet, um alle Schlüssel-Wert-Paare aus einem Dictionary zu entfernen? *5/5

- [x] clear() ✓
- [ ] empty()
- [ ] delete()
- [ ] remove_all()

Feedback

Korrekt! Die clear()-Methode entfernt alle Schlüssel-Wert-Paare aus einem Dictionary. Nach dem Aufruf von my_dict.clear() wird das Dictionary leer sein: {}.

✓ Was ist die Ausgabe des folgenden Codes? a = [1, 2, 3]; b = a; b.append(4); print(a) *5/5

- ○ [1, 2, 3]
- ◉ [1, 2, 3, 4] ✓
- ○ [1, 2, 3, [4]]
- ○ Error

Feedback

Korrekt! In Python werden Listen als Referenzen übergeben. Wenn wir b = a zuweisen, verweisen sowohl a als auch b auf dieselbe Liste. Wenn wir b.append(4) aufrufen, wird 4 zur Liste hinzugefügt, auf die beide Variablen verweisen. Daher enthält a auch den neuen Wert und die Ausgabe ist [1, 2, 3, 4].

✓ Welcher der folgenden Ausdrücke prüft korrekt, ob ein Schlüssel in einem Dictionary vorhanden ist? *5/5

- ◉ key in my_dict ✓
- ○ key in my_dict.keys()
- ○ my_dict.has_key(key)
- ○ my_dict.contains(key)

Feedback

Korrekt! Der Ausdruck key in my_dict prüft, ob key ein Schlüssel im Dictionary my_dict ist. Die zweite Option funktioniert auch, ist aber weniger effizient. Die has_key()-Methode wurde in Python 3 entfernt, und contains() ist keine Dictionary-Methode.

Google Formulare