

Python Programming Associate

Prüfungscode IFW-PYTHON-200-01 · Fassung 1.0 · Stand 18.09.2026

Python 3 im Alltag: Datentypen und Comprehensions, Funktionen und Generatoren, Objektorientierung, Fehlerbehandlung und Module.

Die Prüfung im Überblick

Bearbeitungszeit 1 Std. 30 Min.	Fragen 70
Bestehensgrenze 70 %	Fragetypen Einfach- und Mehrfachauswahl
Prüfungssprache Deutsch	Zertifikat gültig Unbegrenzt
Durchführung Online, zeitlich überwacht	Preis 49,00 €

Themenblöcke

Block	Thema	Gewichtung	Fragen
1	Datentypen, Sequenzen und Comprehensions	33 %	23
2	Funktionen, Generatoren und Namensräume	27 %	19
3	Objektorientierung	20 %	14
4	Fehlerbehandlung	13 %	9
5	Module, Dateien und Ausführung	7 %	5
Gesamt		100 %	70

Die Gewichtung gibt an, welchen Anteil ein Block an den Prüfungsfragen hat. Die Fragen werden beim Start zufällig und anteilig je Block aus dem Fragenpool gezogen.

Datentypen, Sequenzen und Comprehensions

Die eingebauten Typen, ihr Verhalten bei Zuweisung und Kopie sowie kompakte Schreibweisen.

- 1.1** Zahlen und Operatoren verwenden
//, % und die Wahrheitswerte von Zahlen
- 1.2** Strings bearbeiten
Slicing, negative Indizes, Sortierung nach Unicode-Codepunkten
- 1.3** Listen, Tupel, Sets und Dictionaries unterscheiden
veränderlich und unveränderlich, frozenset, hashbare Schlüssel
- 1.4** Referenzen und Kopien beurteilen
Zuweisung ohne Kopie, copy() als flache Kopie
- 1.5** Comprehensions schreiben
List- und Dict-Comprehension mit Bedingung
- 1.6** Wahrheitswerte von Objekten kennen
leere Container, 0, '0' und [] in einer Bedingung

Funktionen, Generatoren und Namensräume

Funktionen definieren und aufrufen, Gültigkeitsbereiche und verzögerte Auswertung.

- 2.1** Funktionen definieren und aufrufen
Positions- und Schlüsselwortargumente, *args und **kwargs
- 2.2** Standardwerte richtig setzen
veränderliche Standardwerte werden nur einmal ausgewertet
- 2.3** Argumente entpacken
Aufruf mit **werte aus einem Dictionary
- 2.4** Gültigkeitsbereiche verstehen
lokale und globale Namen, UnboundLocalError
- 2.5** Generatoren einsetzen
yield, gemerkter Zustand, next() und verzögerte Auswertung

Objektorientierung

Eigene Klassen schreiben, Zustand richtig ablegen und Sprachmittel anbinden.

- 3.1** Klassen und Instanzen anlegen
__init__, self, Instanzattribute

- 3.2** Klassen- und Instanzattribute unterscheiden
geteilte Werte und deren Überschreiben je Instanz
- 3.3** Vererbung nutzen
`super().__init__()`, Methoden überschreiben, `isinstance()`
- 3.4** Dunder-Methoden implementieren
`__repr__`, `__len__`, `__eq__`

BLOCK 4

13 % · 9 Fragen

Fehlerbehandlung

Ausnahmen gezielt abfangen, weitergeben und selbst definieren.

- 4.1** Ausnahmen behandeln
`try`, `except`, `else` und `finally` und ihre Aufgaben
- 4.2** Exception-Hierarchie beachten
Reihenfolge der `except`-Zweige, `KeyError` unter `LookupError`
- 4.3** Eigene Ausnahmen definieren und auslösen
Ableiten von `Exception`, `raise` ohne Argument

BLOCK 5

7 % · 5 Fragen

Module, Dateien und Ausführung

Code aufteilen, Dateien verarbeiten und den Einstiegspunkt festlegen.

- 5.1** Module importieren und verwenden
`import math` und der Zugriff über den Punktoperator
- 5.2** Dateien lesen und schreiben
`with open(...)` als Context Manager, Modi `r`, `w`, `a`, `x`
- 5.3** Skript und Modul unterscheiden
`if __name__ == '__main__':`

Institut für Webentwicklung · unabhängiges Lernangebot ohne Verbindung zu den genannten Technologieanbietern. Maßgeblich ist die Fassung auf der Zertifizierungsseite.