**SWP Python**

Tobias Seyrling 5AHWII

**Kompilieren/Interpretieren:**

**Plattformunabhängig**: Der Quellcode wird in Bytecode kompiliert, der von einer virtuellen Maschine (z.B. Java Virtual Machine) auf verschiedenen Betriebssystemen interpretiert oder just-in-time kompiliert werden kann (z.B. Java). In Python wird der Quellcode interpretiert oder in Bytecode für die Python Virtual Machine (PVM) übersetzt.

**Plattformspezifisch**: Der Quellcode wird direkt in Maschinencode kompiliert, der nur auf einer bestimmten Plattform ausgeführt werden kann (z.B. C++).

**Instanz/Referenz**

* Instanz: Speicherplatz für Daten
* Referenz: Verweisung auf einen Speicher

**Garbage Collector**: Gibt es nur in Verbindung mit Plattformunabhängigen-Sprachen. Gibt ungenutzten Speicher frei und dieser wird mit Nullen überschrieben. Mit dem Del Befehl kann man Speicher im Code freigeben.

**Rekursive/Iterativ:**

* Rekursive: Methode ruft sich selbst auf. Wir für die Lesbarkeit gemacht, Compiler wandelt Rekursive in Iterativ um.
* Iterativ: Schleife

**Arrays/Listen**

* Array: Organisation von gleichen Objekten ohne Referenz, können wir nicht erweitern
* List: Einzelne Objekte sind verknüpft mit einfachen Referenzen auf die Liste, hochgradig flexible (Echte Listen = LinkedListen) (Referenzen werden mit der Liste mitgespeichert) (Liste geht Punkt für Punkt durch)
* ArrayList (Java): Ist ein Array, wird etwas hinzugefügt, wird ein komplett neues Array erstellt (doppelt so groß). Daten werden von altem Array in neues kopiert (Deep Copie (Einzelkopien)). Altes Array muss vom Garbage Collector eingesammelt werden.

Schneller als eine Liste

Array ist zusammenhängend, Liste einzelne Objekte werden in freien Speicher gespeichert muss nicht zusammen liegenVorteil

Es gibt keine Arrays in Python, man kann sie importieren muss jedoch den Datentyp angeben.

Wenn man ein Array mit [] anlegt, wird im Hintergrund eine Liste erstelle, hat jedoch das gleiche Verhalten wie eine ArrayList in Java.

**HashMap**

In Python Dictionary (dict), Key Value System. Erkennt man in Python an {}

woerterbuch = {

"Germany" : "Deutschland",

"Spain" : "Spanien"

}

Prozesse/Threads:

Prozess: ein Programm ist ein Prozess, bekommt gewissen Speicher zugeteilt, verbraucht das Programm = Prozess zu viel Speicher => stürzt ab. Prozesse können nicht miteinander kommunizieren.

Threads laufen innerhalb eines Prozesses. Können miteinander reden aber nur innerhalb eines Prozesses.

**Unterschied Python und Java**

Python und Auto Boxing dynamisch "Typisierung": es werden keine Datentypen angelegt sondern automatisch angepasst = dynamisch

Java zwingt zur Arbeit mit Objekten -> statisch Typisiert = statisch

**Floor Division**: Kommazahlen werden weggelassen

9/10 = 0.9

9//10 = 0

**Interaktiver Modus**: CMD Befehl "python3" oder "py"

-> um Datei im Interaktiven Modus zu öffnen "cd Verzeichnis", "python3 Datei.py"

**Python Keywords**: raise, return, True, try, while, with, yield. Dürfen nicht als variable Namen verwendet werden.

Keywords darf man im Programmcode nicht verwenden, sind vordefiniert da sonst Exceptions auftreten könnten

Python arbeitet auch mit floor-division-in-python gibt dann Ganzzahlige Zahlen aus .

Bsp: 10/20 = 0.5 in Python

Bsp floor-division: 10//20 = 0

Programmiersprachen sind Textdateien

Durch pass kann man eine leere (abstrakte) Methode in Python erstellen

Maschinencode kann von betriebssystem einfach ausgeführt werden

Textdateien müssen durch SheBeng so umgeändert werden dass es vom betriebssystem gelesen werden kann

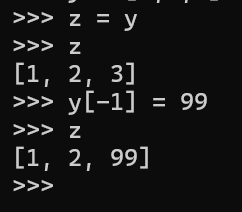
Ein **Shebang** (oft #! gefolgt von einem Interpreterpfad, z.B. #!/bin/bash) am Anfang eines Skripts gibt dem Betriebssystem an, **welcher Interpreter** verwendet werden soll, um die Datei auszuführen.

Mutabler Datentyp: Die Variable wird im Speicher angelegt, und das Objekt kann direkt an Ort und Stelle verändert werden. Zu den mutablen Datentypen gehören in vielen Programmiersprachen z. B. Listen oder Dictionaries (in Python), Arrays (in Java), etc.

Immutable Datentyp (z. B. primitive Datentypen): Das Objekt wird einmal im Speicher gespeichert, und alle Referenzen mit denmselben Wert zeigen auf denselben Speicherort. Wenn der Wert einer immutablen Variable geändert wird, wird ein neues Objekt an einer neuen Speicheradresse angelegt, und die Referenz wird aktualisiert. Z.B. tup

Ein Bild, das Text, Schrift, Screenshot, Typografie enthält.

Automatisch generierte Beschreibung



Seiteneffekt: mutable objekt => alle Referenzen die dorthin zeigen verändern sich mit dem Verändern des mutable objekts (siehe oben wenn man y ändert ändert sich z auch)

Liste speichert nur Referenzen

Primitive Datentypen sind immer Immutable. Werden einfach neu angelegt wenn man z.B. einen String verändert. GarbageCollector gibt dann den speicher der alten referenz frei

Ein Bild, das Text, Schrift, Typografie, Design enthält.

Automatisch generierte Beschreibung

Poker HÜ: 2 Methoden

1 gib mir Farbe

2 gib mir symbol

Als nächstes Methoden, z.B isthisFlash. Man fangt mit den Wenigst Wahrscheinlichen Methoden an, falls etwas getroffen wird wird abgetroffen und wenn nicht geht es weiter hinunter

In einem Dictionary dann abspeichern wie oft welche methode zutrifft und am ende prozentuell ausrechnen

Vorteil Array in Python: Man kann nur einen Datentyp in der Array speichern, weniger speicherplatz

Wenn man verschachtelte Liste kopieren will muss man Deep Kopie machen da ansonst werte in der Verschachtelung verändert werden können

Es gibt 2 Arten von Fehlern:

1. Exceptions: Coding Fehler, von uns verursacht

* 2 Arten
* Die wo es sich lohnt, abzufangen
* Die was sich nicht lohnen abzufangen

1. Errors

Raise Exception schmeißt exception in die nächste exception, man möchte nicht das an dieser Stelle eine Exception darstellt

Pdb : Python Debugger

fStrings, da strings mit srting + „ „ + string mehr Speicher frisst. Gibt es seit python 3.6

SQLInjection, schädigende Befehle die man in Datenbank bringen möchte, verhinderbar durch validierte Eingabe

Prepared Statements sind SQL-Abfragen, die einmal vorbereitet und dann sicher und effizient mit variablen Werten ausgeführt werden können. Sie bieten zwei wesentliche Vorteile:

1. **Schutz vor SQL-Injection**: Da Nutzereingaben separat von der Abfrage verarbeitet werden, sind sie gegen schädliche Manipulationen geschützt.
2. **Leistungsverbesserung**: Da die Abfrage nur einmal kompiliert und dann mehrfach ausgeführt wird, spart dies Ressourcen.

**Beispiel**:

sql = "INSERT INTO users (username, password) VALUES (?, ?)"

cursor.execute(sql, ("user1", "password1"))

Hier wird die Abfrage vorbereitet und kann sicher und schnell mit neuen Werten wiederholt werden.

Lambda: minimal schneller da weniger Speicher belegt wird

Function = lambda x : x +1

Function(5)

Ausgabe 6

Unterschied Methode, Lambda: Man kann bei Lambda nur einen Return wert zurückgeben.

Lambda hat späte Bindung.

Lambdas und Schleifen kann eine Fehlerquelle sein da späte Bindung

Einkapselung durch Lambda

Map: für jedes einzelne iterable wird die function angewendet

Schreibweise: map(function, iterable1, iterable2, iterable3, …)

Iterable: Datenstruktur mit der man durch eine Schleife geben kann

Map (viele Fragen bei nächstem Test)

3 Methoden/functions können in Map eingebaut werden: Normale Methode, Build-In-Methode, Lambda

Wenn die Methode viele Parameter hat braucht man auch viele Iterables

Eine Map gibt den Typ Map zurück

Filter, weniger komplex wie List-Comprehension. None, filtern leere Strings, NULL heraus

Rückgabewert gibt true oder false aus

Filter gibt den Typ filter zurück

Wenn man zwei Listen mit zip kombiniert werden sie zu einem tuple. Geht nur bis zur kürzeren Liste

Zip gibt den Typ zip zurück

Virtual Environments:

Python3.10 -m venv venv\_keller -> Linux

Virtual Environment, gibt uns eine nackte Python Version.

Kann man nicht verschieben.

Source venv-keller/bin/activate

Deactivate (man haut es aus dem speicher)

Man muss in einem Virtual Env. Mit pip install die Pakete herunterladen die man importieren will.

V. E. löschen: rm -rf venv\_keller/

Man ladet nichts herunter man verweist nur auf das Verzeichnis wo es bereits exestiert

Lambdas in Python

Definition: Lambdas sind anonyme Funktionen (d.h. ohne Namen), die direkt definiert und verwendet werden können. Sie sind kompakter als normale Funktionen und eignen sich besonders für einfache Operationen.

Syntax:

python

Code kopieren

function = lambda x: x + 1

print(function(5)) # Ausgabe: 6

Unterschied zwischen Methode und Lambda:

Lambda:

Kann nur einen einzigen Ausdruck enthalten und gibt direkt den Rückgabewert dieses Ausdrucks zurück.

Keine Möglichkeit, komplexe Logik oder mehrere Rückgabewerte einzubauen.

Beispiel:

python

Code kopieren

double = lambda x: x \* 2

print(double(4)) # Ausgabe: 8

Methode (normale Funktion):

Flexibler, kann mehrere Anweisungen und Rückgabewerte enthalten.

Beispiel:

python

Code kopieren

def double(x):

return x \* 2

print(double(4)) # Ausgabe: 8

Späte Bindung bei Lambdas:

Lambdas in Schleifen können fehleranfällig sein, da sie Variablen erst zur Laufzeit binden. Das führt dazu, dass der Wert der Variable sich ändern kann, bevor die Lambda-Funktion ausgeführt wird.

Beispiel:

python

Code kopieren

funcs = [lambda x: x + i for i in range(3)]

print([f(0) for f in funcs]) # Ausgabe: [2, 2, 2] (nicht [0, 1, 2])

Grund: i wird erst ausgewertet, wenn die Lambda-Funktion aufgerufen wird, und hat am Ende den Wert 2.

Einkapselung durch Lambda:

Lambdas können genutzt werden, um Funktionen oder Werte zu kapseln, was bei funktionalem Programmieren hilfreich ist. Dennoch bleiben sie limitiert auf einfache Aufgaben.

Map in Python

Funktion: Wendet eine gegebene Funktion auf jedes Element eines oder mehrerer iterierbarer Objekte (z. B. Listen) an.

Schreibweise:

python

Code kopieren

map(function, iterable1, iterable2, ...)

Beispiel:

python

Code kopieren

nums = [1, 2, 3]

doubled = map(lambda x: x \* 2, nums)

print(list(doubled)) # Ausgabe: [2, 4, 6]

Hinweis: Map kann mit mehreren Iterables arbeiten:

python

Code kopieren

nums1 = [1, 2, 3]

nums2 = [4, 5, 6]

summed = map(lambda x, y: x + y, nums1, nums2)

print(list(summed)) # Ausgabe: [5, 7, 9]

Filter in Python

Funktion: Filtert Elemente eines Iterables basierend auf einer Bedingung (Boolean-Ausdruck).

Schreibweise:

python

Code kopieren

filter(function, iterable)

Beispiel:

python

Code kopieren

nums = [0, 1, 2, 3, 4]

even = filter(lambda x: x % 2 == 0, nums)

print(list(even)) # Ausgabe: [0, 2, 4]

Vergleich mit List Comprehension:

Filter:

python

Code kopieren

even = filter(lambda x: x % 2 == 0, nums)

List Comprehension:

python

Code kopieren

even = [x for x in nums if x % 2 == 0]

Virtuelle Umgebungen in Python

Was ist eine virtuelle Umgebung?

Eine isolierte Python-Umgebung mit eigener Installation von Python und Libraries.

Hilfreich, um Versionskonflikte zwischen Projekten zu vermeiden.

Wichtig: In einer virtuellen Umgebung müssen alle benötigten Libraries (z. B. über pip install) separat installiert werden.

Erstellen einer virtuellen Umgebung:

bash

Code kopieren

python3.10 -m venv venv\_keller

Aktivieren der Umgebung:

Linux/Mac:

bash

Code kopieren

source venv\_keller/bin/activate

Windows:

cmd

Code kopieren

.\venv\_keller\Scripts\activate

Deaktivieren der Umgebung:

bash

Code kopieren

deactivate

Löschen der Umgebung:

bash

Code kopieren

rm -rf venv\_keller/

Wichtig: Virtuelle Umgebungen sind nicht verschiebbar, da sie auf den Pfaden basieren, die bei der Erstellung festgelegt wurden.

**Vererbung:**

\_\_init\_\_ … ist der Konstruktor in Java

self … this in Java

Klassenvariablen gibt es immer nur einmal, Instanzvariablen beliebig oft

Es gibt auch Klassenvariablen

z.B.

class Dog():

species = „Schäferhund“

Unterschied Klassenvariable und Instanzvariable:

z.B. in Java Person = new Person(); -> hat Attribute, diese Attribute sind Instanzvariablen, Person ist die Klassenvariable

class Person:

species = "Human" # Klassenvariable

def \_\_init\_\_(self, name, age):

self.name = name # Instanzvariable

self.age = age # Instanzvariable

person1 = Person("Alice", 30)

person2 = Person("Bob", 25)

# Zugriff auf Klassen- und Instanzvariablen

print(person1.name, person1.species) # Ausgabe: Alice Human

print(person2.name, person2.species) # Ausgabe: Bob Human

# Änderungen

Person.species = "Homo Sapiens" # Klassenvariable ändern

person1.age = 31 # Instanzvariable ändern

print(person1.age, person1.species) # Ausgabe: 31 Homo Sapiens

print(person2.age, person2.species) # Ausgabe: 25 Homo Sapiens

**Instanzvariablen**

* **Definition**: Variablen, die mit dem Präfix self innerhalb von Methoden definiert werden.
* **Gültigkeitsbereich**: Sie gehören **nur zu einer bestimmten Instanz** der Klasse. Jede Instanz hat ihre eigene Kopie der Instanzvariablen.
* **Lebensdauer**: Sie werden erstellt, wenn eine Instanz der Klasse erstellt wird, und gelöscht, wenn die Instanz gelöscht wird.

**Klassenvariablen**

* **Definition**: Variablen, die direkt auf Klassenebene definiert werden (außerhalb von Methoden).
* **Gültigkeitsbereich**: Sie werden von **allen Instanzen der Klasse geteilt**. Es gibt nur eine einzige Kopie der Klassenvariablen.
* **Lebensdauer**: Sie existieren, solange die Klasse im Speicher ist.

In Python ist alles Public

InstanzMethoden erkennt man daran, dass sie als Parameter self stehen haben.

Vererbte Klasse:

Class Child(Parent):

super() ist eine Referenz auf die Elternklasse

Unter Klassen wird das pass-Schlüsselwort häufig verwendet, um leere Klassen oder Methoden zu definieren, die später implementiert werden sollen. Es sorgt dafür, dass der Code ausführbar bleibt, auch wenn die Klasse oder Methode noch keinen Inhalt hat.

\_\_repr\_\_

Zweck:

Sie soll eine unmissverständliche und detaillierte Darstellung des Objekts bieten.

Häufig wird erwartet, dass der zurückgegebene String den Code darstellt, mit dem das Objekt erstellt werden könnte (dies ist jedoch nicht zwingend).

**Unterschied zu \_\_str\_\_**

* **\_\_repr\_\_**: Fokus auf eine **detaillierte und technische Darstellung** des Objekts, oft für Entwickler gedacht.
* **\_\_str\_\_**: Fokus auf eine **benutzerfreundliche Darstellung**, oft für Endbenutzer gedacht.

Wenn nur \_\_repr\_\_ definiert ist, verwendet Python diese Methode auch für str(), falls keine \_\_str\_\_-Methode vorhanden ist.

Wenn man den String kopiert, wird genau dieses Objekt mit den Instanzen erstellt

Dunder-Methoden (Double-Underscore-Methoden) verbinden Benutzeraktionen oder Operatoren mit der Implementierung in einer Klasse.

Proxy = Vermittler

Poylmorphismus: Vielseitigkeit

Die Funktion mit dem gleichen Namen wird vererbt aber jedoch mit anderem Code in der Klasse ausgeführt

Mehrfachvererbung -> auf einer Vererbungsebene gleichzeitig alle Klassen!!!!!