**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

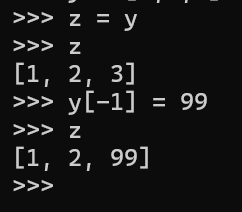
Ein **Shebang** (oft #! gefolgt von einem Interpreterpfad, z.B. #!/bin/bash) am Anfang eines Skripts gibt dem Betriebssystem an, **welcher Interpreter** verwendet werden soll, um die Datei auszuführen.

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.