2 Arten von Datenstrukturen:

Liste

langsamer als array, verkettete Listen, kann man neue elemente/objekte hinzufügen, in objekten stehen value (string value z.b.), es steht ein zweites elemente (next, symbol, zeiger,-die aufs nächste element zeigen)

wird durchgelooped

(einzelne objekte iwo im speicher verteilt und hängen über zeiger zusammen)

oder

Array

im speicher gibts eine Startstelle und die Elemente zum Array sind hintereinander, das betriebssystem definiert des mit anfangsaddresse, typen, (die zahlen hängen alle zusammen hintereinander), array ist schneller als eine Liste

java kann alles in einem rutsch durchlesen, zum array kann man keine neuen elemente hinzufügen

Namespaces

sind nur die Sichtbarkeiten, in Methode haben wir lokale variable -> namespace (leveling)

dictionary ist unsortiert

---------- neue Stunde

Lambda funktion ist kürzere Weise eine Methode zu schreiben und eine "anonyme Methode" da sie keinen Methodennamen braucht/hat

Instanz: Objekt (In einer Klasse macht man mit new eine Instanz (Kopie einer Klasse mit individueller Werte))

Referenz: Ist ein Verweis auf ein Objekt (Zeiger: Rubner zeigt auf Glatzl (link))

Identität: Fingerabdruck eines Objektes

|||||||||||||||||||||||||||||||||||||||| python code

a=6

b=5

a / b

zahl = 8 #instanz obwohl integer

id(zahl) #identität der Zahl

zahl1 = 8

id(zahl1) #id der speicherstelle in der der 8ter liegt

mlist = [0,1,'hallo']

id(mlist)

mlist2 = [0,1,'hallo']

id(mlist2) #id(mlist) != id(mlist2)

l1=[1,2]

l2=l1

l1.append(3)

#wir sehen, dass sowohl l1 als auch l2 mit 3 erweitert wurden (da l2 auf l1 verweist)

||||||||||||||||||||||||||||||||||||||

String ist Inmutable

Alle sachen mit new, listen, ... sind Mutable

a=6

b=a -Referenz

Veränderlich ~~ Mutable - List, Bytearray

man muss eine Instanz erzeugen

Unveränderlich ~~ Inmutable - tuple (liste die inmutable ist), str, bytes

primitiven Datentypen

wenn ich a=6 habe und b=a mache dann wird b auf a gelegt, aber wenn ich a nun ändere, dann

ändere ich nicht b (da inmutable) (im gegensatz zum mutable), da ich nicht auf den gleichen Speicherpunkt in

der Speicherstelle hinweise

wenn man bei mutable nicht anfangsding ändern möchte dann mache ich eine deepcopy

bei deepcopy werden neue instanzen erstellt und bei copy werden nur die refferenzen geändert

copy kopiert nur in der ersten ebene primitive datentypen werden richtig kopiert und referenzen werden normal kopiert

Rheinwerk Seite durchlesen

.append fügt etwas hinzu (z.B. an Liste)

#man muss global for mutable datentypen schreiben !!!HÜ!!!

KW41 hü Pokersimulator

Modulierung, Methode wo man 5 Objekte rausbekommt, array mit 52 stellen und immer über modolo farbe finden und über division farbe finden, 0-12 (13) 26%13(0,1,2,3(Farbe)) 26/13(Zahl)

---------- neue Stunde

Programm zu Maschinen Code (00101010)

Sprachen wie Python, Java, C#, erzeugen Bytecode (kein Maschinencode) .class Dateien -> lesbare Textpassagen und Hieroglyphen (Maschinencode-Teile) enthalten

Vorteile:

Maschinencode Ist am schnellsten(effektivsten) auf dieser einen Hardware

Interpretencode wird geschrieben & compiliert, wir können diese .class datei nehmen und auf jedem System dass ein Javainterpreten hat geben

und benutzen (Plattformunabhängig wo Java (od.a.) installiert ist), einmal neu Compilieren und es funktioniert

Nachteile:

Maschinen-Code funktioniert auf anderen Versionen/PC's nicht raspbpie Versionen, 32-bit versionen, dort

Interpretensprachen: Verwaltet vieles selber (Speicher z.B.) wird durch Garbage-Collector verwendet aber eben nicht auf 100%

effektiv auf dieser Hardware (-> etwas langsam), Python verwendet viel C code weshalb es trotzdem schnell ist da es native c code verwendet,

In welcher Programmiersprache sind Java, C#, Python programmiert?

in C und C ist in Assambler programmiert welcher zwar eine Programmiersprache ist aber hauptsächlich binär-code (sehr hardware nah spezifische Befehle)

{[(steht alles in der Powerpoint unter den Links)]}

__Seiteneffekte:__

Mutable

Objekte die verändert werden Können, Listen, Dictionaries

a=5 b=a // a=5 b=5 -> Id bleibt gleich da Python 5 im Speicher immer gleich speichert (keine neue Instanz)

Immutable

primitive Datentypen z.B. x=3 -> Instanz (es ganze), Referenz zeigt drauf, Objekt hat eindeutige Id

In-Place

nehmen wir an wir haben ein Array (bzw. Datenstruktur) und wir wollen diese Datenstruktur sortieren, z.B. wir schauen ein Objekt

durch und alle Objekte wo ist das Minimum -> wird nach vorne getauscht und es geht weiter (kleinsten Zahlen blubbern nach vorne)

Wir sortieren im Array im Zusatzspeicher -> In-Place

Ist wenn wir nur konstanten Zusatzspeicher benötigen aber nicht von der Anzahl abhängigen speicher

Python oder Java bibliotheken machen abhängige Operationen, z.B. Liste sorted, list2=sorted.list1 (nicht in-place)

Ob ein Verfahren in Datenstrukturen ohne zusätzlichen Speicher

bei der deepcopy mach ich alles in einer neuen Liste

Wenn ich eine Liste habe und in dieser Liste sind nur mehr Imutable objekte drinnen und wir wollen eine copy machen dann funktioniert die deepcopy

Aber wenn ich in einer liste wieder eine liste habe dann ist meine deepcopy keine echte deepcopy sondern nur eine Referenz

{[(Alles durchlesen und fragen in Powerpoint)]}

Style-Guides

sind wichtige Guides damit die Syntax leserlich ist und schön formatiert bleibt

File Encoding

basically Ascii code für die Zeichen bzw Tasten

-> wenn in China programmiere und in Österreich den Code abgebe dann wird es wegen den Verschiedenen

Werten des codes nicht erkannt -> muss also alles angepasst werden

{[(DelftStack -> Code)]}

{[(Rheinwerk-Verlag -> Eingebaute Funktionen)]}

Dunder Variablen

sind in Main bzw Einstiegspunkt

||||||||||||||||||||||||||||||||||||

```
if __name__ == "__main__":
    meine_function()
```

||||||||||||||||||||||||||||||||||||

Dunder = Doubleunderscore

wird von Python standardmäßig befüllt

ist Zeichen, dass eine besondere Variable ist

in jeder der 3 Pythondatei könnte ich if name == könnte dann aber nur dann ausgeführt werden in der Datei in der ich die Dunder benutzt habe

In test1 führe ich der Methoden in der Main aus

---------- neue Stunde

Es gibt wenn alles passt (aber auch nur passt), dann bekommt man einen Dreier

Man muss viele Extras einbauen, damit man in die Richtung eines Einsers kommt.

Von Int Float Double

Int hernehmen, da des Ganzzahlen sind und weniger Rechenleistung benötigen

Mainmethoden machen weil grundlegende funktionen importieren

immer in methoden einpacken zum importieren

::-1 dreht um

---Debuggen:---

Debuggen Schritt für Schritt

https://realpython.com/python-debugging-pdb/#essential-pdb-commands

ternäre operatoren sind if ähnlich

---

min = a < b and a or b

print(min)

a = 5

rval = True if a > 5 else False

---------- neue Stunde

if instance == 2;

  return 2

if instance ==3;

  return 3

--> des haben wir so, da wir mit else sonst nicht returnen könnten

Wann wird ein Objekt iterable? Es isch wenn zwei methoden implementiert sind (dunder-methode __iter__ __next__) macht es iterable

wenn wir zb next aufrufen dann bekommen wir das nächste

https://www.digitalocean.com/community/tutorials/how-to-use-the-python-map-function-de

mapped_numbers = list(map(lambda x: x * 2 + 3, numbers))

-> lambda funktion geht jede list durch und holt numbers heraus

lambda ist eine verkürzte Schreibweise für Funktionen

x ist parameter, nach doppelpunkt kommt methodenkörper, somit kann man ganz ganz schnell eine methode schreiben, sobald

es fertig ist, kommt des ausm stack wieder raus

map ist iterierbare Datenstruktur

powerfunktion pow nimmt 2 parameter -> man kann 2 iterables angeben

https://www.w3schools.com/python/python_iterators.asp

Klasse -> Kapselung ist dafür dass wir Sachen beschützen, Struct (sachen da drinnen sind beschützt und eine Kapsel)

Wir wollen unsere kostbaren schätze (variablen) schützen, da wir so des Fehlerpotential begrenzen

https://realpython.com/python3-object-oriented-programming/

__init__

ist die initialisierende Methode

Konstruktormethode

(self, aeckj, noece) verweist auf sich selbst

static -> variable/methode gehört zu der klasse, alles andere gehört zu einer Instanz

Static und bound methods

Klasse, plates stück metal, da drücken wir eine form raus, des isch dann a instanz

Static methoden und static variable

stat. var. sind Klassenatribute

Stat. hat this bzw self nichtmehr und wird deshalb instanzübergreifend

Instanz macht man indem wir den klassennamen aufrufen und () schreiben -> neue Instanz

im bsp-link ist Dog() mutable, weils ein Objekt ist, jedes Objekt ist Mutable

Wenn wir irgendwas instanzieren und einen konstruktor mit 2 parametern, müssen wir diese 2 parameter angeben

es funktioniert genau gleich wie java, wir haben keinen default konstruktor

wenn wir einen print haben, wollen wir öfers nicht die adresse der instanz speicherstelle sondern die repräsentation

__str__() ist eine art ToString

die Kindklassen sind immer vom Typ Elterns

miles ist dog aber dog ist kein miles

wenn wir jemals erben, dann nehmen wir alles von der Elternklasse mit

```
class JackRussellTerrier(Dog):
    def speak(self, sound="Arf"):
        return super().speak(sound)
```

arf ist default sound, wird aber nicht aufgerufen sobald wir etwas angeben

ab dem Zeitpunkt ab dem wir weiter schreiben sollte jeder einen Parameter haben

Speak benutzt man immer wenn man auf die Elternklasse kommen will, benutzt man also immer wenn wir in der Kinderklasse bin

Super macht eine Temporäre Instanz von elternklasse und deswegen können wir darauf zugreifen

https://realpython.com/python-super/#an-overview-of-pythons-super-function

mit Super können wir Konstruktor aufrufen

Multiple inheritance so viel wie nötig so wenig wie möglich

Er sucht immer von links nach rechts und bei klammern von innen nach außen

wenn er triangle area hat oder braucht

mro

---------- neue Stunde

Seiteneffekte ist wenn man von außen darauf zugreifen kann

Liste verändern, refferenz zeigt drauf und dadurch verändert man orginal und das verweißte dann das wird geändert - Seiteneffekte

Konstruktor heißt in Python __init__

Mehrfachvererbung ist in p möglich, in java nicht

Modifikatoren in python wie static gibt es nicht wirklich

es gibt instanz und klassen variablen

membervariablen in java und python, wenn wir in klasse außerhalb definieren wird es zu einer membervariable mit einer instanz, erst wenn wir

static dazuschreiben, dann wird es statisch

wenn eine klasse von ner anderen klasse erbt dann kann man in der erbenden klasse super schreiben, damit sie von mutterklasse erbt

was ist super? - eine Refferenz

Lambda ist speicherschonend da kürzer

Map sind wie for-schleifen und können iterable funktionen ausführen

iterable gehen jede stelle durch und ein objekt ist iterable wenn es zb eine liste ist, oder __iter__ und __next__ müssen implementiert werden dann ist es iterable (iter-anfang, next-nächstes)

In Java ist es keine echte Mehrfachvererbung

I muss die eindutigkeit vergeben, wenn man mehrfach vererbt dann muss man aufpassen wenn man eine struktur mit den selbenmethodennamen nimmt - achtung überlagerung (overload)

bei java muss ich super.super.super eingeben was ich in python nicht muss

In der init methode kann ich schauen wie viele argumente ich habe und nach der anz. der Argumente gehen

Wir könnten noch eine separate methode machen oder vererbung

Wir haben mehrere Konstruktoren

Counter für wie oft der Mensch was spielt und des soll der pc nehmen und er soll random

und das auf verschiedenen schwierigkeitsstufen

wie speichere ich daten? Txt datei oder sqlight datenbank

Daten gezählt, flask api im hintergrund der eine db hat und an den wir daten schicken -> 2 separate programme

Wir wollen eine Programmierumgebung die exakt so abgestimmt ist dass unser programm funktioniert

Bibliotheken können sehr sehr viel aber sie verändern sich laufend

Python rohinstallation erweitert man in einem Ordner auf unserer Festplatte, welche nur in einem Bestimmten Ordner sind

Python beim normalen rechner eingeben in cmd dann kommt systempython

mit virtual invironment "activate" kommt python

Wir haben viruatl environment installiert und bibliotheken importiert

-> es gibt zwei befehle pip (python installer) installiert zusätzlich bibliotheken

pip freeze -> |

wenn jemand meine repository klont: pip install -r

pip install pipreqs

bei pip freeze werden alle imports reingeschrieben

wenn wir pipreqs eingeben schaut es nach wer es wirklich verwendet im code und benutzt nur die erforderlichen (eigentlich eigenen librarys)

geht ordnermäßig vor

ist an unsere hardware gekoppelt, sobald wir an unserem neuen rechner sind müssen wir es erneut aufbauen

wir haben projekt, dann hab ich gitinit, dann gitignore python c*jpg oder so kein plan, pipreqs installieren (oder freeze)

---------- neue Stunde

Namespaces geben Scope an, in dem sie definiert sind (Sichtbarkeit)

-> Sammlung von Variablen

Es gibt global, enclosing, Built-in, und local namespace

Enclosing ist wenn in python in der methode eine andere Methode definiert wird (nested Methods)

local -> enclosing -> global -> built-in

enclosed ist die zweite def in einer def (enclosing)

https://realpython.com/python-namespaces-scope/

Wenn wir eine variable nicht finden bekommen wir eine name-error exception

Alles was ich global definiere ist überall Sichtbarkeit

built in, ist des was python schon direkt vordefiniert hat

Error: kann man nicht handlen

Exception: kann man handlen

python verwaltet refferenzen als Dictionaries

->

global()

{add:adfa,aadfa:hs,nrt:rmrr,...}

global vor variabel schreiben für Sichtbarkeit

x,y,z = 1,2,3

nonlocal isch um global zu umgehen

-Best Practices durchlesen zuhause-

---------- neue Stunde