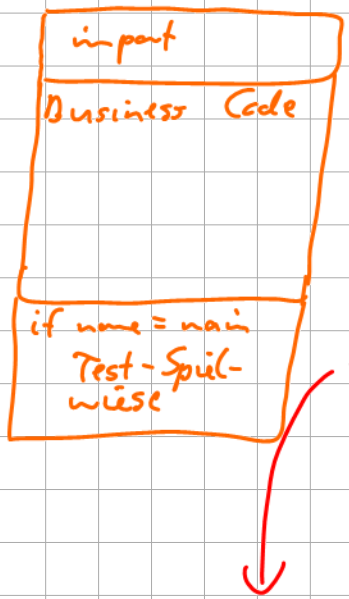
Integrationstests

Klasse 1 : $x \rightarrow H \rightarrow y$

Klasse 2 : $y \rightarrow H^{-1} \rightarrow \tilde{x}$

↳ Problem der Integrationstests:
- welche Abweichung zwischen x und $\tilde{x}$ sind akzeptabel aufgrund der Algorithmik

Python Skript:



→ wenn hier Code auftaucht, der nirgendwo anders im Python Projekt benötigt wird, folgt: verschieben in „if name = main" Block

→ In Business Code sind assert zulässig

→ „aufwändigere" Tests müsse in den Block „if name = main"

Anmerkung: schnelle Tests sind besser ⇔ auf Laufzeit-probleme achten

Tipp für alle, die Dateioperationen haben:

a) Dateioperationen wenn möglich in eigene Klasse auslagern.

b) Dummy-Instanz dieser Dateioperations-Klasse anlegen, ausführen, Testen/Prüfen ob alle Dateien korrekt angelegt sind, anschließend diese temporären Dateien wieder löschen.

⇒ wie bei Tests (Unittests / Integrationstests)

nach dem Test muss das Testsystem (wie Notebook) wieder in identischen Zustand sein, wie vor dem Test.

c) Save / Load danach muss Klasse wieder in Ursprungszustand sein.

Alle Definitionen von Unittests in „if name = main" Block

Ordnerstruktur MOPS:
a) flach im ORdner liegen alle Python-Skripte. Alles was flach drin liegt und kein Python-Skript ist, entspricht einer temporären Datei und kann ignoriert/gelöscht werden.
b) Alle größeren Dateiansammlungen / benutzte Projekte (bspw. Whisper) liegen in Unterordnern.

Aufruf der MOPS-Software:
Train.py um das Training zu starten.
- In VOCABULARY und HMMVOCABULARY lege ich die Wörter, die detektiert werden sollen, fest.
- Der Resultcontroller wird konfigurert: welcher detektierte String gehört zu welcher Python-Prozedur.
SelfTest.py um den Zustand des Projektes zu testen.
main.py um den MOPS zu starten.

main.py (praktisch alle Python-Projekte haben im Startordner eine main.py, diese ist fast immer der Einstiegspunkt der Software):
a) Controller starten: MVC = Model View Controller
View: User Interface (hier: Kommandozeile); wichtig: es darf nur eine Prozedur/Klasse im Businesse Code aufgerufen werden.
Model: Dateien, die im Hintegrrund die Software konfigurieren
Controller: zentrale Konfiguration des Business Code
b) Hier: die main.py legt einen Controller an und startet diesen.

Der Controller verwaltet (Anlegen (über eine Factory), Start, Stop) einen Signalfluss (über eine Liste von SignalFlowBlock)

In der Factory ist der Signalfluss des MOPS definiert.
Factory ist ein Design Pattern der Objektorientierten Programmierung (Gang of four)
Factorys liefern konfigurierte Klassen.