**Mesa**

**Tutorial von Mesa:** <https://colab.research.google.com/github/projectmesa/mesa/blob/main/docs/tutorials/0_first_model.ipynb>

Cmd-Konsole öffnen:

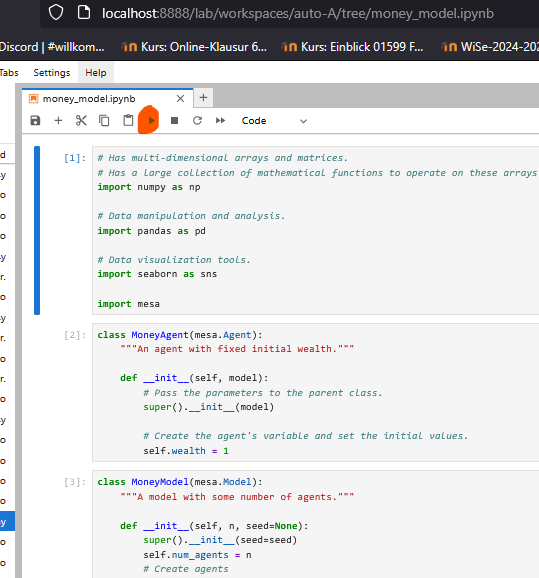
pip install mesa[rec]

pip install jupyter

pip install seaborn

jupyter lab

- Code von Colab in Jupyter kopieren und ausführen lassen



**Video Tutorial bei YouTube:**  
<https://www.youtube.com/watch?v=fUrUWnWGHEQ&list=PLF0b3ThojznRpQOd7iFukqXybbMV_vwZn>

Beginnt ab Video 3 mit der Programmierung

**Codebeispiele:** https://mesa.readthedocs.io/latest/examples.html

**Was ist Mesa (ABM Framework)?**

Mesa ist ein **Python-Framework**, das speziell dafür entwickelt wurde, **Agentenbasierte Modelle (ABMs)** zu erstellen, auszuführen, zu analysieren und zu visualisieren. Es bietet eine Sammlung von Werkzeugen und vordefinierten Komponenten, die den Aufbau solcher Modelle erheblich vereinfachen.

**Was ist Agentenbasierte Modellierung (ABM)?**

Bevor wir tiefer in Mesa eintauchen, kurz zu ABM:

Agentenbasierte Modellierung ist eine Simulationsmethode, bei der ein System durch das Verhalten und die Interaktionen vieler einzelner, autonomer "Agenten" modelliert wird. Jeder Agent folgt einfachen Regeln, und das komplexe Verhalten des Gesamtsystems ("emergentes Verhalten") ergibt sich aus diesen Interaktionen innerhalb einer definierten Umgebung. Beispiele sind die Simulation von Epidemieausbreitungen (Agenten = Personen), Verkehrsflüssen (Agenten = Fahrzeuge) oder sozialen Dynamiken (Agenten = Individuen).

**Was bietet Mesa konkret?**

Mesa stellt die Bausteine zur Verfügung, um solche Modelle in Python zu implementieren:

1. **Modell-Klasse (Model):** Der Hauptcontainer für deine Simulation. Hier definierst du die globale Logik, erstellst die Agenten und die Umgebung und legst fest, wie die Simulation abläuft.
2. **Agenten-Klasse (Agent):** Eine Vorlage, um deine individuellen Agenten zu definieren. Jeder Agent hat Eigenschaften (seinen Zustand) und Verhaltensregeln (typischerweise in einer step-Methode definiert, die in jedem Simulationsschritt aufgerufen wird).
3. **Scheduler:** Bestimmt die Reihenfolge, in der die Agenten in jedem Zeitschritt aktiviert werden. Mesa bietet verschiedene Scheduler (z. B. alle gleichzeitig, zufällig, nacheinander).
4. **Raum/Grid (Space):** Definiert die Umgebung, in der sich die Agenten befinden und interagieren. Mesa bietet verschiedene Typen:
   * Grid: Ein Gitter (wie ein Schachbrett).
   * ContinuousSpace: Ein kontinuierlicher Raum.
   * NetworkGrid: Agenten interagieren über ein Netzwerk (soziale Netzwerke etc.).
5. **Datensammler (DataCollector):** Ein sehr nützliches Werkzeug, um systematisch Daten während der Simulation zu sammeln. Du kannst festlegen, welche Eigenschaften der Agenten oder des Modells in jedem Zeitschritt aufgezeichnet werden sollen (z. B. Anzahl infizierter Agenten, durchschnittliches Vermögen).
6. **Visualisierung:** Mesa integriert eine **webbasierte Visualisierungskomponente**. Damit kannst du deine Simulation live im Browser beobachten. Du siehst, wie sich Agenten bewegen, wie sich ihre Zustände ändern, und kannst dir gleichzeitig Diagramme der gesammelten Daten anzeigen lassen. Das ist extrem hilfreich zum Verständnis und Debugging des Modells.

**Warum Mesa verwenden?**

* **Einfachheit:** Basiert auf Python, einer weit verbreiteten und relativ leicht zu lernenden Sprache.
* **Struktur:** Gibt eine klare Struktur für ABMs vor, was die Entwicklung erleichtert.
* **"Batteries Included":** Bietet viele notwendige Komponenten (Scheduler, Raum, Datensammlung, Visualisierung) direkt an.
* **Modularität:** Die Komponenten sind gut voneinander getrennt, was Anpassungen und Erweiterungen erleichtert.
* **Open Source:** Kostenlos verfügbar, der Code ist einsehbar und eine Community entwickelt es weiter.

**Zusammenfassend:**

Mesa ist ein leistungsstarkes und benutzerfreundliches Python-Framework, das Forschern und Entwicklern hilft, Agentenbasierte Modelle zu bauen und zu analysieren. Es nimmt einem viel von der grundlegenden Programmierarbeit ab, sodass man sich stärker auf die Definition der Agenten, ihrer Regeln und der Analyse der Simulationsergebnisse konzentrieren kann. Es ist ein Werkzeugkasten für die Erstellung "künstlicher Welten" im Computer zur Untersuchung komplexer Systeme.