1.
Beschrijving:
Mesa is agent based dit is erg makkelijk te zien doordat je 2 classes moet aanmaken. 1 voor het model en 1 voor de agents. Een groot voordeel van mesa is dat mesa erg gestructureerd is / wordt geschreven in python waardoor iedereen wel begrijpt wat er staat. Dit is ook wat mesa erg anders maakt dan andere talen. Daarbuiten is Mesa vrij ongelimiteerd in mogelijkheden. Het enige nadeel dat ik zag was dat je soms lange lines krijgt als:
self.model.grid.get_neighborhood(self.pos, moore=True, include_center=False)
Maar dit is vooral wennen met het noemen van alle punten. Verder blijft hoe dieper je in de stof gaat het steeds ingewikkelder worden maar dat is natuurlijk logisch.

2.

1. In mijn tutorial is de initiële staat de __init__ hier in krijgt de agent zijn id en het model waar het in zit. Verder wordt hier ook de start wealth aangegeven waar elke agent mee begint.
2. De see functie zit verstopt in de move functie. In de move functie kijkt de agent naar welke mogelijke cellen het heen bewegen (maar niet wat er in zit).
3. De act functie is in dit geval de give_money functie. Hier kijkt het naar de hoeveelheid andere agents in dezelfde cel waarna als andere agents in de cel zitten hij een van hun extra wealth geeft. als de cel buiten de agent leeg is doet het niets.
4. Volgens mij heeft mijn tutorial geen update functie. Het enige wat ik er op vind lijken is de step functie die de move en give_money laat uitvoeren.

3.
Allereerst is het erg makkelijk om agents te maken om in het model te opereren. Dit komt doordat de agent alle accurate informatie kan krijgen van het model. Dit is te zien in de give_money waar het makkelijk een actuele lijst van de agents op dezelfde (of andere) cel kan opvragen. Verder hebben alle acties een gegarandeerd effect. Dit zorgt ook weer voor een makkelijker process van het maken van de agents. Verder bepaald de agent elke step wat het doet zonder te kijken naar voorgaande steps. Wel ziet het of hij *op het moment* nog wealth heeft voor het verder gaat. Ook wordt het model alleen aangepast door de agent. Er is geen andere invloed buiten dat van de agents. Tot slot lijkt het model enorm op een schaakspel. De agents hebben namelijk alleen de keuze om naar de vakjes om hem heen te gaan en daar 1 actie uit te voeren. Dus al met al zorgt dit voor een erg aangename programmeerervaring.

4.
Een voorbeeld waarbij minimaal 3 dichotomies tegenovergesteld zijn is een simulatie voor zweefvliegtuigen. Zweefvliegtuigen zijn namelijk erg afhankelijk van luchtstromen waar ze zelf geen invloed op hebben. Verder is de route / de afstand die ze kunnen afleggen per step ook niet bepaald. Verder hebben de acties van een zweefvliegtuig ook niet een specifiek effect.