Color: Name of changed file, code added, code revised

1. Extension1: Add a fourth category of the population for exposed (but not yet infectious) agents and a new parameter for the incubation period.
   Changes:

- Agent_Based_Simulation_of_COVID_19_Health_and_Economical_Effects.ipynb → second cell of simulation1 → amplitudes={
  Status.Susceptible: 5,
  Status.Exposed: 5,
  Status.Recovered_Immune: 5,
  Status.Infected: 5
  }
- abs.py →

```
def contact(self, agent1, agent2):
    """
    Performs the actions needed when two agents get in touch.

    :param agent1: an instance of agents.Agent
    :param agent2: an instance of agents.Agent
    """

    if agent1.status == Status.Susceptible and agent2.status == Status.Infected:
        contagion_test = np.random.random()
        agent1.infection_status = InfectionSeverity.Exposed
        if contagion_test <= self.contagion_rate:
            # agent1.status = Status.Infected
            agent1.status = Status.Exposed
            # agent1.infection_status = InfectionSeverity.Asymptomatic
            agent2.number_infected += 1
```

- abs.py →

```
def update(self, agent):
    """
    Update the status of the agent

    :param agent: an instance of agents.Agent
    """

    if agent.status == Status.Death:
        return

    # calculate incubation days of an agent
    if agent.status == Status.Exposed:
        agent.incubation_days += 1
        if agent.incubation_days == agent.incubation_period:
            agent.incubation_days = 0
            agent.status = Status.Infected
            #agent.infected_status = InfectionSeverity.Exposed
        return
```

- graphhics.py →

```
def color1(s):
    """Plotting colors by status string"""
    if s == 'Susceptible':
        return 'lightblue'
    elif s == 'Exposed':
        return 'yellow'
    elif s == 'Infected':
        return 'gray'
    elif s == 'Recovered_Immune':
        return 'lightgreen'
    elif s == 'Death':
        return 'black'
    elif s == 'Hospitalization':
        return 'orange'
    elif s == 'Severe':
        return 'red'
    else:
        return 'white'
```

- experiments.py →

```
def plot_graph_batch_results(df, health_metrics=('Susceptible', 'Exposed', 'Infected',
'Hospitalization', 'Severe', 'Recovered_Immune', 'Death'),
 ecom_metrics=('Q1', 'Q2', 'Q3', 'Q4', 'Q5','Business','Government'), **kwargs):
```

→

```
def plot_batch_results(df, health_metrics=('Susceptible', 'Exposed', 'Infected', 'Hospitalization',
'Severe', 'Recovered_Immune', 'Death'),
                       ecom_metrics=('Q1', 'Q2', 'Q3', 'Q4', 'Q5')):
```

- agents.py →

```
class Status(Enum):
    """
    Agent status, following the SIR model
    """
    Susceptible = 's'
    Exposed = 'e'
```

- ```
  class Agent(object):
   """
   The container of Agent's attributes and status
   """
   def __init__(self, **kwargs):
    …...
        self.environment = kwargs.get('environment', None)
            """The number of agents infected by this agent"""
            self.number_infected = 0
            self.incubation_days = 0
            self.incubation_period = 7
  ```
-

2. Extension2:
  ```