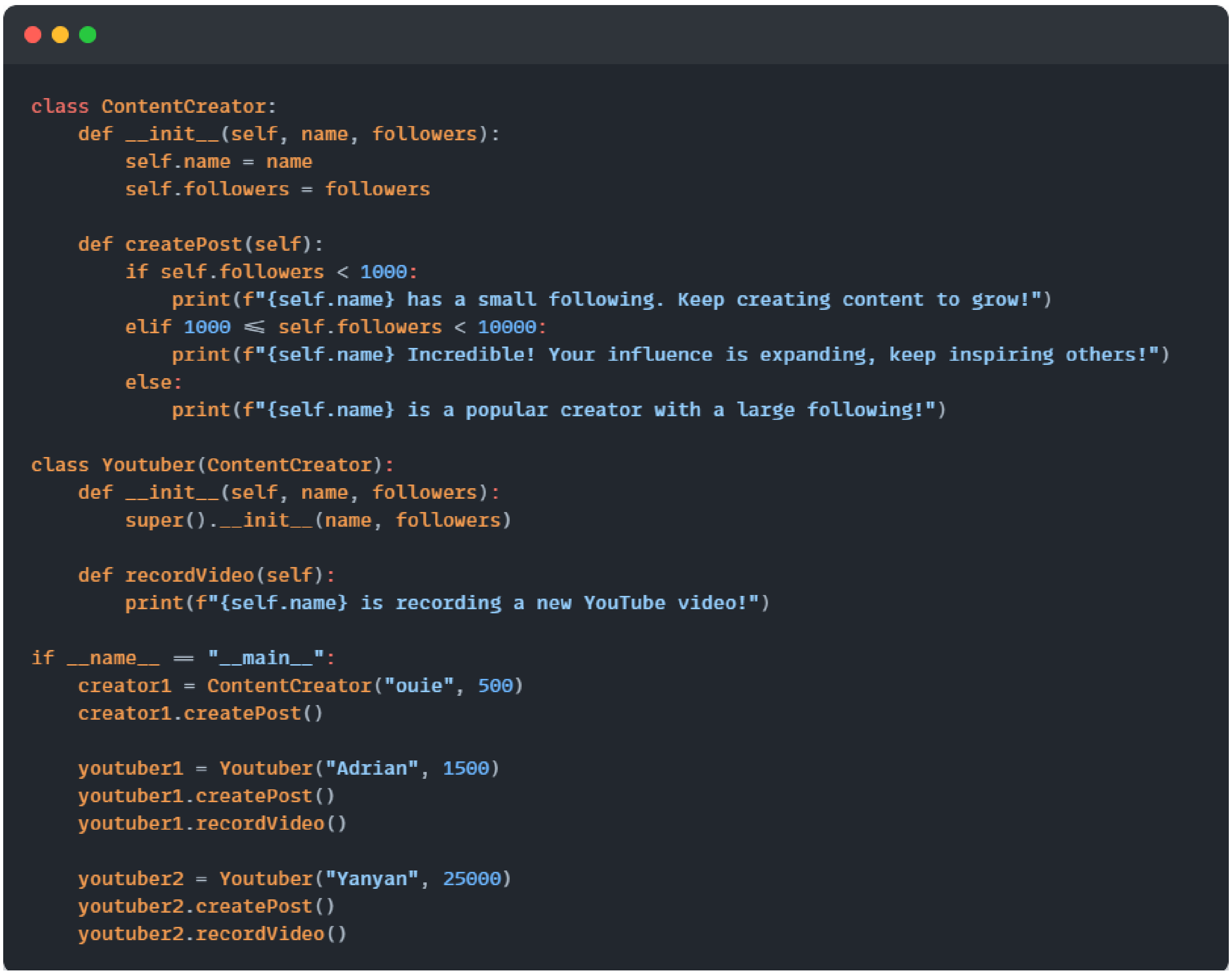
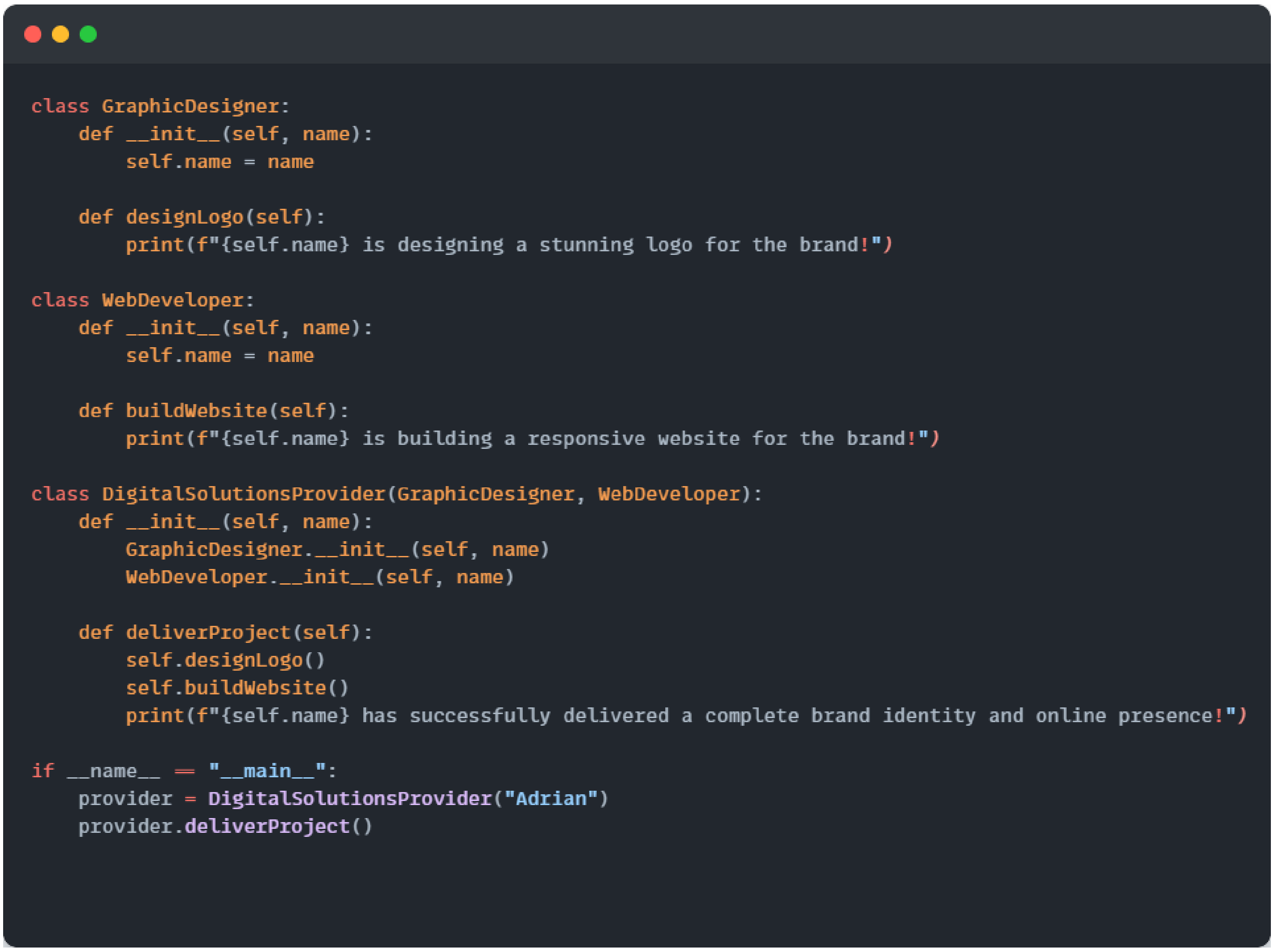
Single Inheritance



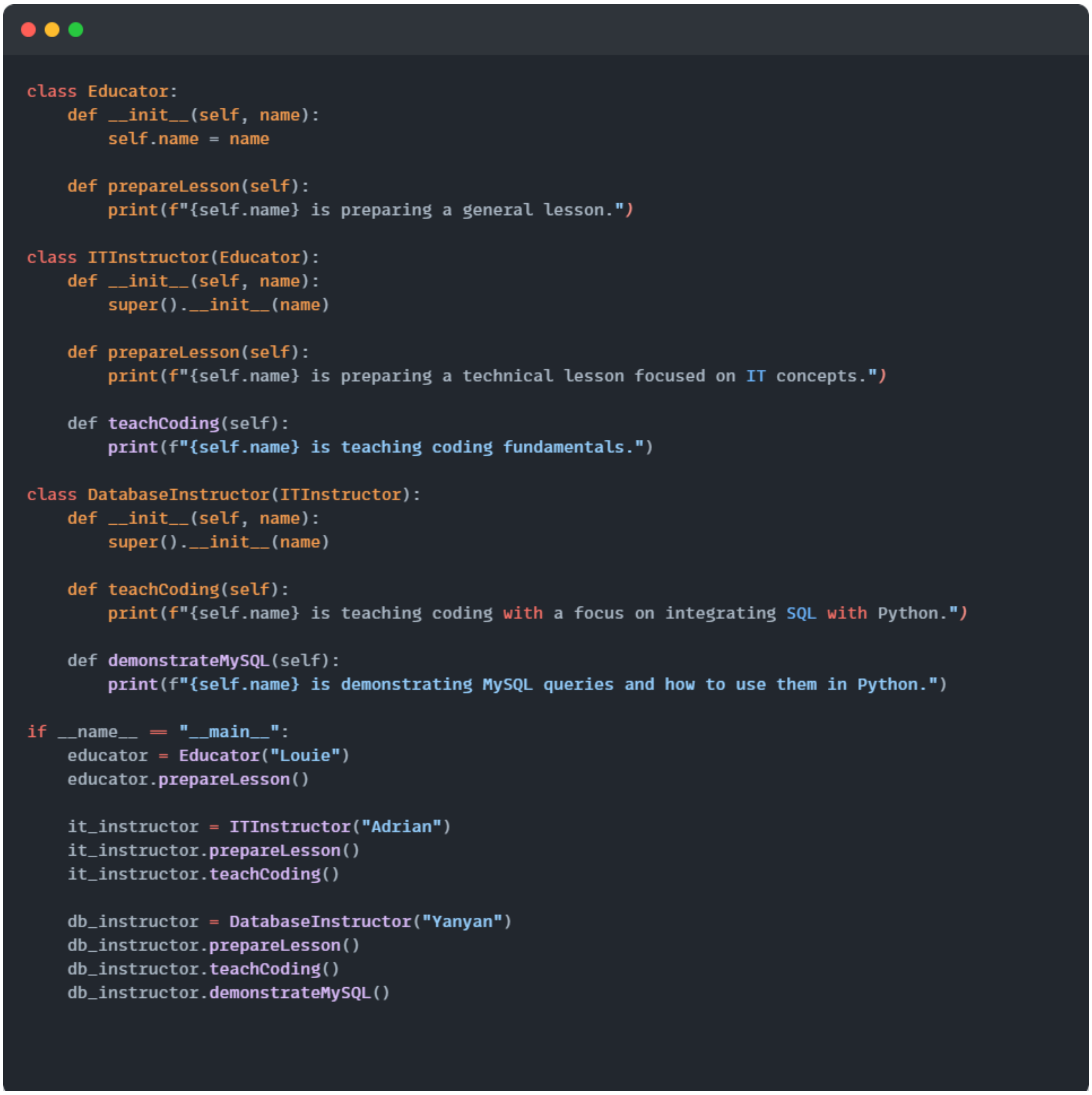
```python
class ContentCreator:
    def __init__(self, name, followers):
        self.name = name
        self.followers = followers

    def createPost(self):
        if self.followers < 1000:
            print(f"{self.name} has a small following. Keep creating content to grow!")
        elif 1000 <= self.followers < 10000:
            print(f"{self.name} Incredible! Your influence is expanding, keep inspiring others!")
        else:
            print(f"{self.name} is a popular creator with a large following!")

class Youtuber(ContentCreator):
    def __init__(self, name, followers):
        super().__init__(name, followers)

    def recordVideo(self):
        print(f"{self.name} is recording a new YouTube video!")

if __name__ == "__main__":
    creator1 = ContentCreator("ouie", 500)
    creator1.createPost()

    youtuber1 = Youtuber("Adrian", 1500)
    youtuber1.createPost()
    youtuber1.recordVideo()

    youtuber2 = Youtuber("Yanyan", 25000)
    youtuber2.createPost()
    youtuber2.recordVideo()
```

Multiple Inheritance



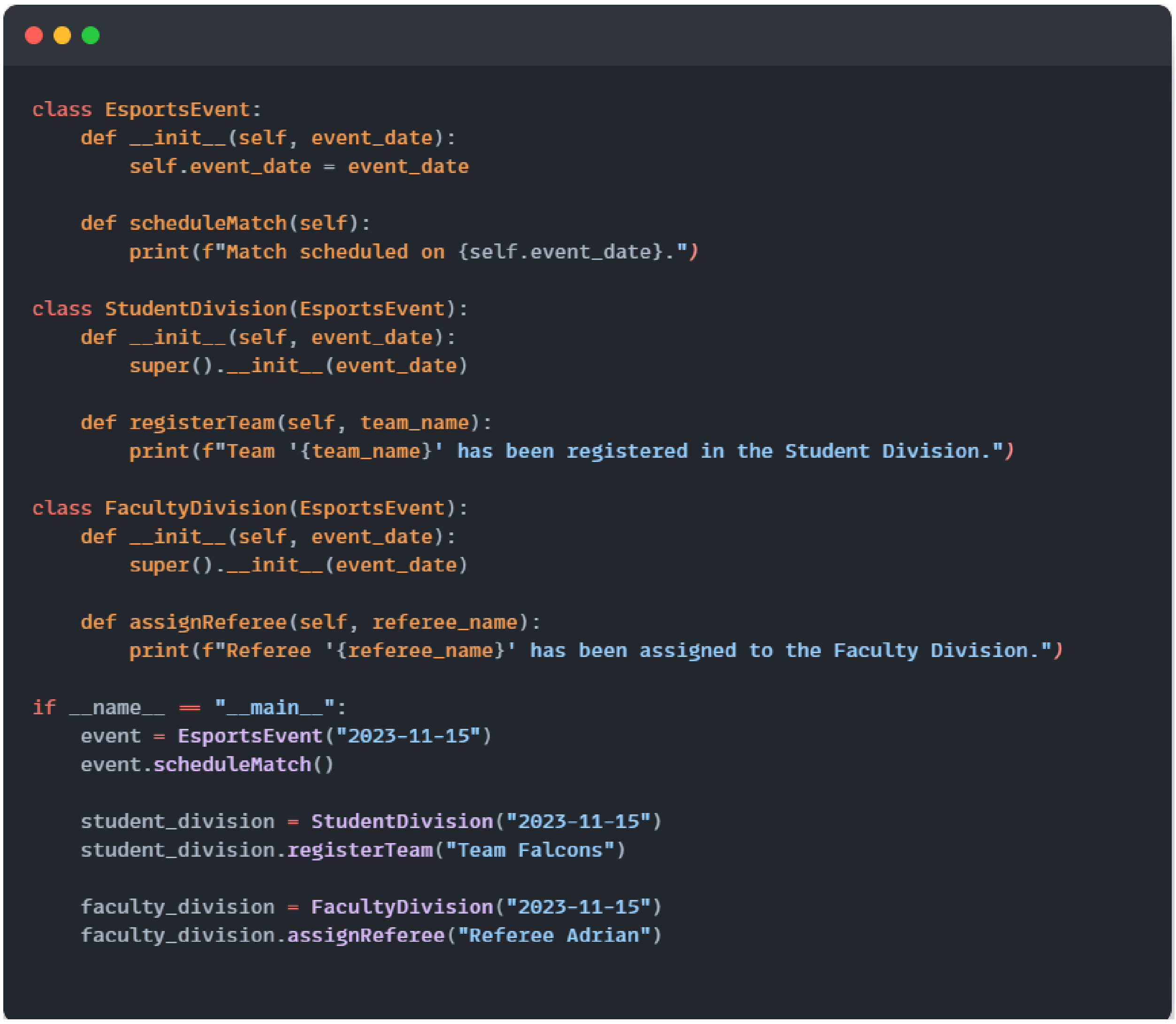
```python
class GraphicDesigner:
    def __init__(self, name):
        self.name = name

    def designLogo(self):
        print(f"{self.name} is designing a stunning logo for the brand!")

class WebDeveloper:
    def __init__(self, name):
        self.name = name

    def buildWebsite(self):
        print(f"{self.name} is building a responsive website for the brand!")

class DigitalSolutionsProvider(GraphicDesigner, WebDeveloper):
    def __init__(self, name):
        GraphicDesigner.__init__(self, name)
        WebDeveloper.__init__(self, name)

    def deliverProject(self):
        self.designLogo()
        self.buildWebsite()
        print(f"{self.name} has successfully delivered a complete brand identity and online presence!")

if __name__ == "__main__":
    provider = DigitalSolutionsProvider("Adrian")
    provider.deliverProject()
```

Multilevel Inheritance



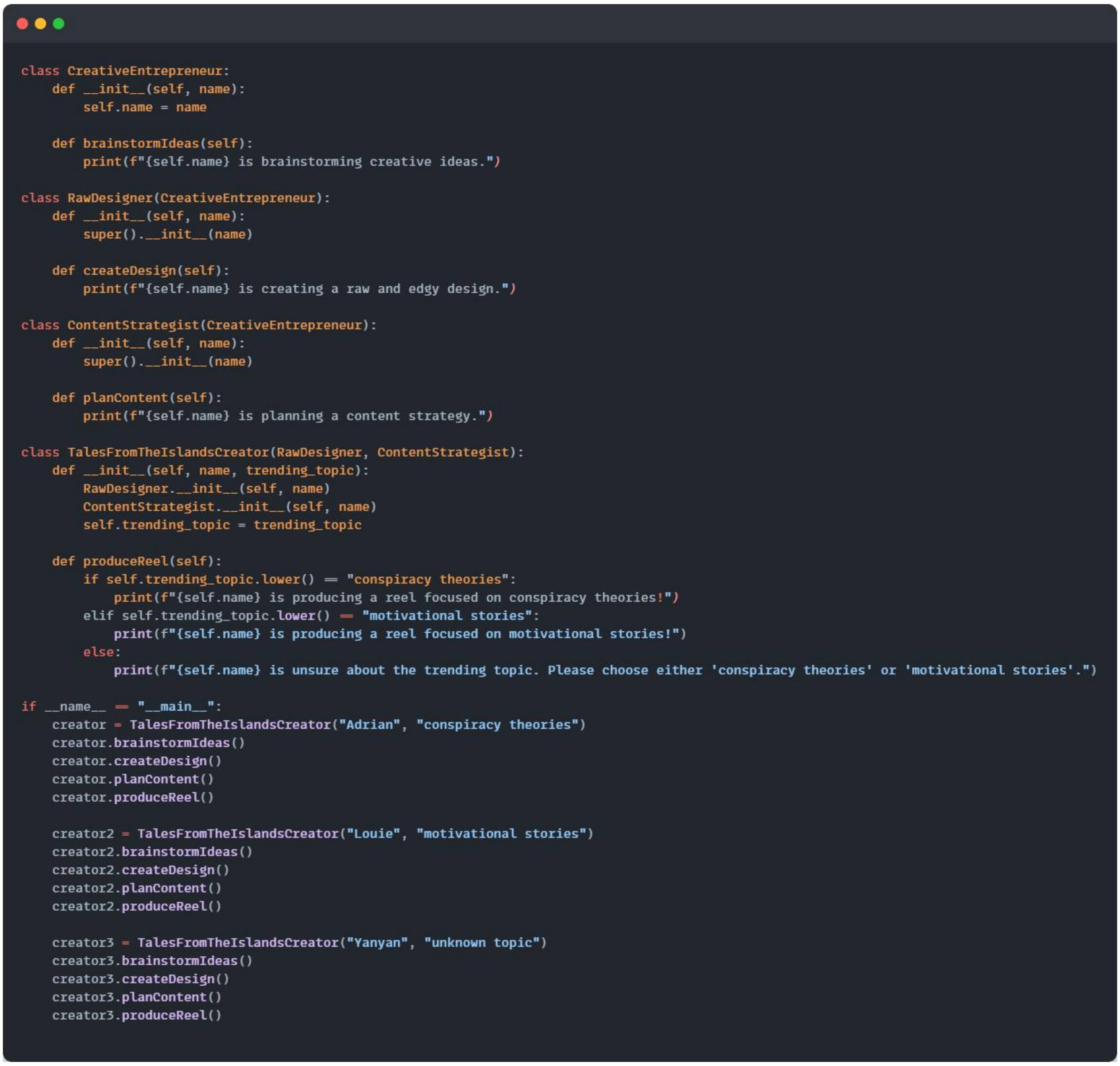
```python
class Educator:
    def __init__(self, name):
        self.name = name

    def prepareLesson(self):
        print(f"{self.name} is preparing a general lesson.")

class ITInstructor(Educator):
    def __init__(self, name):
        super().__init__(name)

    def prepareLesson(self):
        print(f"{self.name} is preparing a technical lesson focused on IT concepts.")

    def teachCoding(self):
        print(f"{self.name} is teaching coding fundamentals.")

class DatabaseInstructor(ITInstructor):
    def __init__(self, name):
        super().__init__(name)

    def teachCoding(self):
        print(f"{self.name} is teaching coding with a focus on integrating SQL with Python.")

    def demonstrateMySQL(self):
        print(f"{self.name} is demonstrating MySQL queries and how to use them in Python.")

if __name__ == "__main__":
    educator = Educator("Louie")
    educator.prepareLesson()

    it_instructor = ITInstructor("Adrian")
    it_instructor.prepareLesson()
    it_instructor.teachCoding()

    db_instructor = DatabaseInstructor("Yanyan")
    db_instructor.prepareLesson()
    db_instructor.teachCoding()
    db_instructor.demonstrateMySQL()
```

Hierarchical Inheritance



```python
class EsportsEvent:
    def __init__(self, event_date):
        self.event_date = event_date

    def scheduleMatch(self):
        print(f"Match scheduled on {self.event_date}.")

class StudentDivision(EsportsEvent):
    def __init__(self, event_date):
        super().__init__(event_date)

    def registerTeam(self, team_name):
        print(f"Team '{team_name}' has been registered in the Student Division.")

class FacultyDivision(EsportsEvent):
    def __init__(self, event_date):
        super().__init__(event_date)

    def assignReferee(self, referee_name):
        print(f"Referee '{referee_name}' has been assigned to the Faculty Division.")

if __name__ == "__main__":
    event = EsportsEvent("2023-11-15")
    event.scheduleMatch()

    student_division = StudentDivision("2023-11-15")
    student_division.registerTeam("Team Falcons")

    faculty_division = FacultyDivision("2023-11-15")
    faculty_division.assignReferee("Referee Adrian")
```

Hybrid Inheritance



```python
class CreativeEntrepreneur:
    def __init__(self, name):
        self.name = name

    def brainstormIdeas(self):
        print(f"{self.name} is brainstorming creative ideas.")

class RawDesigner(CreativeEntrepreneur):
    def __init__(self, name):
        super().__init__(name)

    def createDesign(self):
        print(f"{self.name} is creating a raw and edgy design.")

class ContentStrategist(CreativeEntrepreneur):
    def __init__(self, name):
        super().__init__(name)

    def planContent(self):
        print(f"{self.name} is planning a content strategy.")

class TalesFromTheIslandsCreator(RawDesigner, ContentStrategist):
    def __init__(self, name, trending_topic):
        RawDesigner.__init__(self, name)
        ContentStrategist.__init__(self, name)
        self.trending_topic = trending_topic

    def produceReel(self):
        if self.trending_topic.lower() == "conspiracy theories":
            print(f"{self.name} is producing a reel focused on conspiracy theories!")
        elif self.trending_topic.lower() == "motivational stories":
            print(f"{self.name} is producing a reel focused on motivational stories!")
        else:
            print(f"{self.name} is unsure about the trending topic. Please choose either 'conspiracy theories' or 'motivational stories'.")

if __name__ == "__main__":
    creator = TalesFromTheIslandsCreator("Adrian", "conspiracy theories")
    creator.brainstormIdeas()
    creator.createDesign()
    creator.planContent()
    creator.produceReel()

    creator2 = TalesFromTheIslandsCreator("Louie", "motivational stories")
    creator2.brainstormIdeas()
    creator2.createDesign()
    creator2.planContent()
    creator2.produceReel()

    creator3 = TalesFromTheIslandsCreator("Yanyan", "unknown topic")
    creator3.brainstormIdeas()
    creator3.createDesign()
    creator3.planContent()
    creator3.produceReel()
```