یادداشت‌های درسی (۱)

از نگاه من روح آدمی بدون تعلیم و ترتیب همچون سنگ مرمری در معدن است که هیچیک از زیباییهای ذاتی خود را نشان نمی‌دهد تا هنگامی که دست هنرمند حجار آن رنگها را با صیقلی هویدا کند و صفحه‌ای درخشان پدید آورد که آن ابرهای شکوهمند و خط و خال و رگه‌های پنهان در جسم سنگ را در پیش چشم نهد.
جوزف اَدیسن (۱۷۱۹-۱۶۷۲)

- نوشتن برنامه‌های کوچک نیازمند رعایت اصول و نکات خاصی در نحوه نگارش خطوط برنامه نیست. برنامه‌نویس کافی است که خط به خط ، دستورات مورد نظر را بنویسد. اما نوشتن برنامه‌های بزرگ به این شیوه، حتی اگر منجر به تولید برنامه‌ای درست شود، احتمالاً منجر به تولید برنامه‌ای خوانا و مفید نخواهد شد.
- برنامه‌نویس باید برای نگارش برنامه‌های بزرگ، مطابق با سبک برنامه‌نویسی خاصی عمل کند. دو سبک مرسوم و محبوب برنامه‌نویسی عبارتند از سبک **روالی** (procedural) و سبک **شیءگرایی** (object-oriented) .
- در سبک روالی، برنامه به تعدادی ماژول (واحد) تقسیم می‌شود؛ و هر ماژول، خود به تعدادی روال (تابع) تقسیم می‌شود. هر ماژول نقش ویژه‌ای را در برنامه بازی می‌کند و هر تابع در یک ماژول نیز، کار خاصی می‌کند. نهایتاً با فراخوانی و تعامل ماژول‌ها و تابع‌ها با یکدیگر است که آنها خروجی مورد نظر برنامه را تولید می‌کنند.
- گرچه می‌توان با سبک روالی، هر الگوریتمی را به برنامه تبدیل کرد و مسأله را حل کرد، اما این سبک همیشه بهترین سبک برنامه‌نویسی نیست.
- امروزه مرسوم‌ترین و محبوب‌ترین سبک برای تولید نرم‌افزار (برنامه‌های بزرگ) ، سبک شیءگرایی است. در واقع، چهار زبان برنامه‌نویسی C++ ، C# ، Java و Python (که مرسوم‌ترین و اصلی‌ترین زبان‌ها برای تولید نرم‌افزار هستند) همه بر مبنای سبک برنامه‌نویسی شیءگرایی ایجاد شده‌اند؛ گرچه زبان‌های C++ و Python امکان نوشتن برنامه‌ها به سبک روالی را نیز به برنامه‌نویس می‌دهند.

برنامه‌نویسی شیءگرایی (Object-Oriented Programming)

- طبق این سبک، برنامه‌نویس باید **اشیاء** (objects) یا چیزهایی (things) را که هر یک از آنها نقشی در برنامه او دارند، مشخص کند.
- هر شیءای که مد نظر برنامه‌نویس باشد، باید مشخصه‌هایی (characteristics) داشته باشد و قادر به انجام رفتارهایی (behavior) نیز باشد. برای مثال، اگر شیء مورد نظر سگ باشد، می‌توان بسته به نیاز، مشخصه‌هایی چون «نام، رنگ، سرعت، قد، وزن و نژاد» و رفتارهایی چون «پارس کردن، خوابیدن، راه رفتن، دویدن، خوردن، بازی کردن و گاز گرفتن» را برای آن در نظر گرفت.

- برنامه‌نویس باید قالبی (رده یا دسته‌ای) برای هر شیء مشخص کند. در دنیای برنامه‌نویسی، به این قالب class گفته می‌شود. (کلمه class یکی از کلمات کلیدی زبان برنامه‌نویسی پایتون است.) در بدنه هر کلاس، باید مشخصه‌ها و رفتارهای اشیاء عضو آن توصیف شوند.
- به مشخصه‌های اشیاء، attribute (صفت) و گاهی نیز field یا instance variable یا data member گفته می‌شود.
- رفتارهای اشیاء را باید با تعدادی function توصیف و پیاده کرد. به چنین تابع‌هایی member function یا method نیز گفته می‌شود.
- نوشتن یک برنامه شیءگرا، مستلزم نوشتن حداقل یک کلاس است. برنامه‌نویس باید ابتدا کلاس خود را با تعیین «صفات و رفتارهای اشیاء» تعریف کند و آنگاه، به تعداد لازم، اشیایی را از آن کلاس تولید کند و از طریق آن اشیاء، به حل مسأله بپردازد. در واقع، بازیگران اصلی برنامه‌های شیءگرا، اشیاء هستند. تا اشیایی ایجاد نشوند و نقش خود را ایفا نکنند، تعریف کلاسی چیزی نخواهد بود جز یک تعریف بی‌حاصل!

مثال ۱: class Mobile

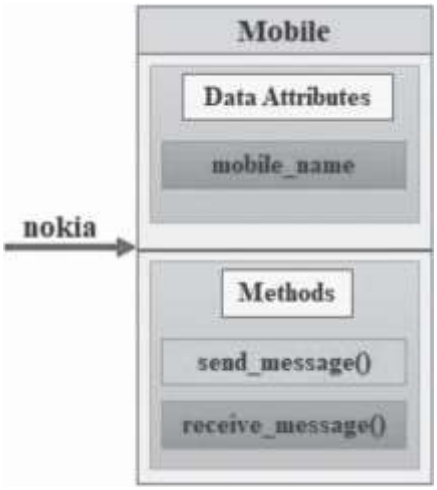
فرض کنید شیء مورد نظر یک mobile باشد و رفتارهای آن «دریافت پیام» و «ارسال پیام» باشند. در این برنامه، کلاسی به نام Mobile برای توصیف رفتارهای این شیء تعریف شده است و آنگاه با ایجاد یک شیء از کلاس، عملیات «دریافت پیام و ارسال پیام» انجام شده‌اند.

```
1.  class Mobile:
2.      def __init__(self):
3.          print("This message is from Constructor Method")
4.      def receive_message(self):
5.          print("Receive message using Mobile")
6.      def send_message(self):
7.          print("Send message using Mobile")
8.  def main():
9.      nokia = Mobile()
10.     nokia.receive_message()
11.     nokia.send_message()
12. if __name__ == "__main__":
13.     main()
```

- از خط ۱ تا خط ۷ کلاس Mobile تعریف شده است. در اولین خط از تعریف هر کلاس، کلمه کلیدی class و سپس نام کلاس و سپس : آورده می‌شود. (قرارداد این است که نام کلاس، یک اسم مفرد باشد، با حرف بزرگ آغاز شود و اگر چند کلمه‌ای است، هر کلمه آن با حرف بزرگ آغاز شود.)
- در این کلاس، صفتی (attribute) برای اشیاء در نظر گرفته نشده است.
- کلاس Mobile شامل سه متد (تابع) است. تابع __init__() (خطوط ۲ و ۳) تابع ویژه‌ای است به نام **سازنده** (constructor) . این تابع، همان گونه که از نام آن بر می‌آید، برای ساخت اشیاء مورد استفاده قرار می‌گیرد. فراخوانی این تابع (در خط ۹) ، یک شیء از کلاس Mobile ایجاد می‌کند. هر کلاس تنها می‌تواند یک تابع سازنده داشته باشد.
- تابع receive_message(self) (در خطوط ۴ و ۵) و تابع send_message(self) (در خطوط ۶ و ۷) رفتارهای اشیاء عضو کلاس را توصیف می‌کنند.
- تنها ورودی (آرگومان) هر سه تابعی که در بدنه کلاس تعریف شده‌اند، self است. در حالت کلی، اولین ورودی چه تابع سازنده و چه هر یک از دیگر توابع عضو یک کلاس self است، گرچه هر یک از آنها می‌توانند علاوه بر self ، ورودی‌های دیگری هم داشته باشند.
- self چیزی نیست جز متغیری که به شیء‌ای که اخیراً استفاده شده است، اشاره می‌کند. از آنجا که از یک کلاس ممکن است به هر تعدادی شیء ایجاد شود. با استفاده از متغیر self می‌توان مشخص کرد که کدام شیء (کدام نمونه از کلاس) متدی از کلاس را فراخوانی کرده است. مفسر پایتون، شیء‌ای را که باعث فراخوانی متد شده است، با پارامتر self متناظر می‌کند.
- self یک کلمه کلیدی در زبان پایتون نیست و می‌توان از کلمه دیگری به جای آن استفاده کرد. اما استفاده از این کلمه در میان برنامه‌نویسان پایتون عرف شده است و بهتر است برنامه‌نویسان تازه‌کار نیز به این عرف احترام بگذارند!
- از خط ۸ تا خط ۱۱ ، تابعی تعریف شده است به نام main() . این تابع، جایی است که تابع سازنده و دیگر توابع عضو کلاس در بدنه آن فراخوانی می‌شوند. در واقع، تابع main() را می‌توان درگاه ورود به برنامه دانست.
- در خط ۹ ، تابع سازنده فراخوانی شده است و شیء‌ای از کلاس Mobile ایجاد شده است به نام nokia . تابع سازنده همیشه با نام __init__() تعریف می‌شود، اما با نام کلاس فراخوانی می‌شود.
- در خطوط ۱۰ و ۱۱ ، شیء nokia به دریافت پیام و ارسال پیام مشغول شده است! برای فراخوانی توابع عضو کلاس، ابتدا باید نام شیء آورده شود و سپس عملگر . (نقطه) و سپس نام تابع.
- توجه کنید که در فراخوانی تابع سازنده و دو تابع دیگر در خطوط ۹ و ۱۰ و ۱۱ ، پارامتری به نام self استفاده نشده است. ولی مفسر، در پشت صحنه، شیء‌ای را که یک تابع را فرا خوانده است، به عنوان اولین پارامتر (متناظر با آرگومان self در امضای تابع) به تابع فرا خوانده شده می‌فرستد.
- در خطوط ۱۲ و ۱۳ ، مفسر پایتون، برابری متغیر ویژه‌ای به نام __name__ (که یک متغیر built-in است؛ یعنی قبلاً در خود زبان تعریف شده است) و رشته "__main__" را بررسی می‌کند. در صورت درست بودن شرط if ، تابع main() فراخوانی خواهد شد و از آنجا توابع دیگر.

مثال ۲: class Mobile

- این برنامه مانند برنامه قبل است، جز آنکه برای تابع سازنده __init__() ، علاوه بر self ، ورودی (آرگومان) دیگری نیز به نام name در نظر گرفته شده است و اشیاء عضو کلاس، علاوه بر دو متد، یک صفت نیز به نام mobile_name دارند. در خط ۳ ، با استفاده از کلمه self ، صفت mobile_name با آرگومان name تابع سازنده، متناظر شده است؛ یعنی وقتی شیءای ایجاد شد، مقدار صفت mobile_name آن، برابر با مقداری (پارامتری) که هنگام فراخوانی تابع سازنده تعیین شده است، قرار گیرد. بنابراین، فراخوانی تابع سازنده در خط ۹ با پارامتر "Nokia" باعث می‌شود که self.mobile_name = "Nokia" شود.




```
1.  class Mobile:
2.      def __init__(self, name):
3.          self.mobile_name = name
4.      def receive_message(self):
5.          print(f"Receive message using {self.mobile_name} Mobile")
6.      def send_message(self):
7.          print(f"Send message using {self.mobile_name} Mobile")
8.  def main():
9.      nokia = Mobile("Nokia")
10.     nokia.receive_message()
11.     nokia.send_message()
12. if __name__ == "__main__":
13.     main()
```