مفاهیم کلی شیءگرایی در زبان برنامه‌نویسی پایتون

درس برنامه‌سازی پیشرفته – ترم ۱۴۰۲۱

استاد: آقای دکتر سید امیرحسین طباطبایی

دستیار تدریس: علیرضا برون

کلمات کلیدی Keywords

کلاس یا کلس= Class

روش یا متد = Method

خصیصه یا ویژگی= Attribute

متد سازنده= __init__

Contents

:OOP

برنامه‌نویسی شیءگرا، الگوی برنامه‌نویسی است که بر روی ایجاد اشیاء حاوی داده و متدهایی برای تغییر داده‌ها تمرکز دارد. پایتون یک زبان برنامه‌نویسی شیءگرا است و به طور کامل از مفاهیم OOP پشتیبانی می‌کند.

در پایتون، می‌توانید کلاس‌ها را برای تعریف اشیاء ایجاد کنید. یک کلاس مانند یک نقشه یا الگوی ایجاد اشیاء است. اشیاء نمونه‌هایی از یک کلاس هستند. در ادامه مثالی را بررسی می‌کنیم:

```
# تعریف یک کلاس به نام 'Person'
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def greet(self):
        print(f"Hi, I'm {self.name} and I have
{self.age} years old.")

# ایجاد یک شیء از کلاس 'Person'
person1 = Person("Ali", 25)

#دسترسی به ویژگیهای شیءخرو
print(person1.name)  #       : Ali
print(person1.age)   # 25 :خروجی

# فراخوانی یک متد شیء
person1.greet()  # خروجی: Hi, I'm Ali and I have 25 years
old.
```

توضیح:

-یک کلاس با نام `Person` تعریف می‌شود که دو ویژگی `name` و `age` و یک متد `greet` دارد.

متد `__init__` یک متد ویژه است که سازنده نامیده می‌شود و وقتی یک شیء (از کلاسی) ایجاد می‌شود، اجرا می‌شود.

-با فراخوانی سازنده و ارسال مقادیر برای `name` و`age` ، یک شیء از کلاس `Person` به نام `person1` ایجاد می‌شود.

- می‌توان با استفاده از نماد "شیء نقطه ویژگی" به ویژگی‌های شیء دسترسی پیدا کرد.

- متد `greet` شیء `person1` را فراخوانی می‌کند و پیامی با استفاده از ویژگی‌های شیء چاپ می‌شود.

نتیجه:

Ali

25

Hi, I'm Ali and I have 25 years old.

مشاهده میکنید که این مثال، ایجاد کلاس، نمونه‌گیری از اشیاء، دسترسی به ویژگی‌ها و فراخوانی متدها بر روی این اشیاء را نشان می‌دهد.

ویژگی‌ها:

در برنامه‌نویسی شیءگرا، ویژگی‌های یک شیء در داخل کلاسی که آن شیء را نمایندگی می‌کند، تعریف می‌شوند.

ویژگی‌ها، حالت یا ویژگی‌های یک شیء را نشان می‌دهند. آنها داده‌هایی را تعریف می‌کنند که یک شیء می‌تواند نگه دارد. هر شیء که از یک کلاس ساخته شده است، مجموعه‌ای از مقادیر ویژگی خود را دارد.

ویژگی‌ها با استفاده از اعلانات متغیر در داخل یک کلاس تعریف می‌شوند. این متغیرها می‌توانند انواع داده‌ای مختلفی مانند عدد صحیح، رشته‌ها، Boolean و غیره داشته باشند. به طور معمول، آنها در بالای تعریف کلاس، خارج از هر متدی اعلان می‌شوند.

در زیر مثالی از یک کلاس در زبان پایتون آورده شده است که ویژگی‌ها را تعریف می‌کند:

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

    def say_hello(self):
        print("Hi, I'm", self.name)

# creating objects of class Person
person1 = Person("Raha", 30)
person2 = Person("Ali", 25)
# accessing to attributes:
print(person1.name)  # خروجی: Raha
print(person2.age)   # خروجی: 25
```

در این مثال، کلاس `Person` دو ویژگی به نام‌های `name` و `age` دارد. متد `__init__` یک متد ویژه به نام سازنده است که برای مقداردهی اولیه ویژگی‌ها هنگام ایجاد یک شیء استفاده می‌شود. پارامتر `self` به نمونه شیء در حال ایجاد (درون کلاس) ارجاع دارد.

ویژگی‌ها با استفاده از نماد نقطه (dot notation) قابل دسترسی اند و تغییر داده می‌شوند، همانطور که در کد بالا نشان داده شده است هر شیء نسخه خودش از ویژگی‌ها را دارد که این امر باعث میشود تا اشیاء مقادیر مختلفی داشته باشند.

متدها:

متدها در کلاس‌ها، توابعی هستند که رفتار شیء را تعریف می‌کنند. آنها با شیءهایی که از یک کلاس ساخته شده‌اند مرتبط هستند و می‌توانند عملیات‌ها و فعالیت‌های مختلفی را بر روی داده‌های شیء انجام دهند. متدها یک راه برای تعامل و تغییر داده‌ها در شیء را فراهم می‌کنند.

متدها همیشه ضروری نیستند، اما به طور معمول برای انجام وظایف خاصی مربوط به شیءهای متعلق به کلاس استفاده می‌شوند.

متدها در داخل کلاس تعریف می‌شوند، به طور مشابهی که ویژگی‌ها (یا خصیصه‌ها) تعریف می‌شوند. آنها معمولاً پس از تعریف ویژگی‌ها نوشته می‌شوند. در ادامه مثالی آورده شده است:

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

    def say_hello(self):
        print("Hi, I'm", self.name)

    def get_age_in_months(self):
        return self.age * 12

person = Person("Raha", 30)
person.say_hello()  # خروجی: Hi, I'm Raha.

age_in_months = person.get_age_in_months()
print(age_in_months)  # خروجی: 360
```

در مثال بالا، کلاس `Person` دو متد به نام‌های `say_hello()` و `get_age_in_months()` دارد. متد `say_hello()` یک پیام با استفاده از نام فرد را چاپ می‌کند، در حالی که متد `get_age_in_months()` سن فرد را از سال به ماه تبدیل کرده و برمی‌گرداند.

برای فراخوانی یک متد روی یک شیء، از نماد نقطه (`شیء.نام_متد()`) استفاده می‌شود.

متدها راهی برای انجام عملیات مربوط به شیءهای یک کلاس را فراهم می‌کنند. آنها می‌توانند توسط خود شیء درون یا از خارج کلاس فراخوانی شوند. آنها جزئی اساسی از تعریف رفتار و قابلیت‌های شیء در برنامه‌نویسی شیءگرا هستند.

اشیاءِ کلاس:

در برنامه‌نویسی شیءگرا، اشیاء، نمونه‌هایی از یک کلاس هستند. یک کلاس الگویی است برای ایجاد اشیاء که ویژگی‌ها و رفتارهای آنها را تعریف می‌کند. هنگامی که یک نمونه از یک کلاس ایجاد می‌کنید، یک شیء ایجاد می‌شود که ویژگی‌ها و رفتارهای تعریف شده توسط کلاس را دارا می‌باشد.

برای تعریف یک شیء در پایتون، ابتدا باید یک کلاس را تعریف کرد. پس از داشتن یک کلاس، می توانید با فراخوانی سازنده کلاس (که خودش در هنگام معرفی شیء فراخوانی میشود (به صورت خودکار)) یک شیء ایجاد کنید. سازنده یک متد ویژه است که هنگام ایجاد یک نمونه از کلاس فراخوانی می‌شود. این متد شیء را با هر مقدار پیش فرض یا پارامترهایی که مشخص می‌کنید، مقداردهی اولیه می‌کند.

در زیر مثالی از چگونگی تعریف یک کلاس ساده در پایتون و ایجاد یک شیء از آن آمده است:

```
class Person:
    def __init__(self, name):
        self.n = name

person1 = Person("Ali")
person2 = Person("Negin")
person3 = Person("John")
```

در این مثال، یک کلاس `Person` با یک سازنده که یک پارامتر به نام `name` را می‌گیرد، تعریف شده است. سازنده ویژگی `n` شیء را با مقدار پارامتر `name` مقداردهی اولیه می‌کند.

سپس در کد با فراخوانی سازنده کلاسِ `Person` با آرگومان‌های مختلف مثل "Ali"، یک شیء به نام `person1` ایجاد می‌کنیم. (در این مثال، سه شیء به نام‌های ``person1``، ``person2`` و `person3` ایجاد شده‌اند، هرکدام با یک نام متفاوت هستند.)

ما میتوانیم تعداد دلخواهی از اشیاء از یک کلاس ایجاد کنیم و هر شیء میتواند مجموعه ای از ویژگی‌هایِ مستقل از ویژگی‌های دیگر نمونه‌های کلاس را داشته باشد. و اشیاء در دامنه‌ی برنامه تعریف می‌شوند. شما می‌توانید آنها را در هر جایی از کد خود که دسترسی به تعریف کلاس دارد، ایجاد کنید. با این حال، روش معمول ایجاد اشیاء در ابتدای است.

علاوه بر این، در پایتون، اشیاءِ یک کلاس می‌توانند فقط ویژگی‌هایی داشته باشند بدون اینکه روشی (متد) را داشته باشند.

مثالی از یک کلاس که فقط ویژگی‌های دارد بدون متدهایی:

```python
class Car:
    def __init__(self, brand, color, year):
        self.brand = brand
        self.color = color
        self.year = year

car1 = Car("Toyota", "Red", 2022)
print(car1.brand)  # خروجی: Toyota
print(car1.color)  # خروجی: Red
print(car1.year)   # خروجی: 2022
```

در این مثال، کلاس `Car` سه ویژگی`brand` ، `color`و `year` دارد. متد سازنده (`__init__`) این ویژگی‌ها را با مقادیری که در هنگام ایجاد شیء وارد می‌شوند، مقداردهی اولیه می‌کند. شیء یا (Object) با نام `car1` با مقادیر خاصی از ویژگی‌ها ایجاد شده است و می‌تواند با استفاده از شیءنقطه، (dot notation) به این ویژگی‌ها دسترسی پیدا کند.

خروجی:

Toyota

Red

2022

همچنین ممکن است کلاسی با فقط متدها و بدون ویژگی‌ها داشته باشید، اگرچه این کمتر رایج است. متدها عملکرد یا آعمالی را فراهم می‌کنند که روی اشیاء کلاس قابل انجام است.

همچنین این موضوعِ داشتن یا نداشتن ویژگی یا متد، به تبع طراحی کلاس توسط شما برمیگردد.

اما در مورد روش‌ها (متدها) در یک کلاس، اینکه می‌توانند با ویژگی‌هایِ یک شیء تعامل داشته باشند و آن‌ها را تغییر دهند یا ندهند. و در حالت کلی، لزوما روش‌ها نیازی به تغییر ویژگی‌ها ندارند.

آنها می‌توانند عملیات‌های مختلفی را بر روی ویژگی‌ها انجام دهند، مانند دسترسی به مقادیر آن‌ها، تغییر آن‌ها یا استفاده از آن‌ها برای انجام محاسبات. همچنین می‌توانند اقداماتی را انجام دهند که به صورت مستقیم با ویژگی‌ها سر و کار ندارند، مانند نمایش اطلاعات یا اجرای رفتاری مشخص.

مثال زیر نشان می‌دهد که چگونه روش‌ها می‌توانند با ویژگی‌ها تعامل داشته باشند:

```python
class Circle:
    def __init__(self, radius):
        self.radius = radius

    def get_area(self):
        return 3.14 * self.radius ** 2

    def set_radius(self, new_radius):
        self.radius = new_radius

circle1 = Circle(5)
print(circle1.get_area())  # خروجی: 78.5

circle1.set_radius(7)
print(circle1.get_area())  # خروجی: 153.86
```

در این مثال، کلاس `Circle` یک ویژگی به نام `radius` دارد. روش `get_area()` محاسبه‌ی مساحت دایره را انجام میدهد که بر اساس ویژگی شعاع است و مقدار مساحت را برمی‌گرداند (که خود مساحت نیز میتوانست ویژگی باشد اما در اینجا در متد سازنده تعریف نشده است). روش `set_radius()` امکان تغییر مقدار ویژگی شعاع را فراهم می‌کند.

هنگامی که `circle1` با شعاع ۵ ایجاد می‌شود، فراخوانی روش `get_area()` مساحت دایره را برمی‌گرداند (78.5). سپس، از روش `set_radius()` استفاده می‌شود تا شعاع را به ۷ تغییر دهد. دوباره فراخوانی `get_area()` نشان می‌دهد که شعاع به روز شده و مساحت جدید برگشت داده می‌شود (153.86).

به یاد داشته باشید که اینکه یک روش ویژگی‌ها را تغییر می‌دهد یا خیر، به توجه به نحوه پیاده‌سازی آن روش است. ممکن است روش‌هایی وجود داشته باشند که به طور کامل ویژگی‌ها را تغییر دهند یا ندهند و به جای آن عملیات‌ها یا محاسبات دیگری را بر اساس ویژگی‌ها انجام دهند (مانند مثال بالا).

تابع یا متد؟

در پایتون، کلاس‌ها خود حاوی توابعی به نام روش‌ها اند. روش‌ها در یک کلاس به توابع معمولی شبیه هستند، اما در داخل کلاس تعریف می‌شوند و برای عملیات روی ویژگی‌ها و رفتارهای آن کلاس طراحی شده‌اند.

روش‌ها ممکن است شامل دستورات return یا print باشند، همانند توابع معمولی. انتخاب میان استفاده از return و print درون یک روش، بستگی به رفتار مورد نظر دارد. اگر می‌خواهید روش محاسباتی را انجام دهد و یک مقدار را برگرداند، از دستور return استفاده می‌کنید. اگر می‌خواهید روش اطلاعاتی یا خروجی را نمایش دهید، از دستور print استفاده می‌کنید. بنابراین میتوان گفت که کلاس‌ها در پایتون می‌توانند به عنوان یک خانواده از توابع در نظر گرفته شوند که هر تابع (روش) وظیفه‌ی خاصی را مرتبط با کلاس و اشیاء آن اجرا می‌کند.

و درنهایت ما می‌توانیم ویژگی‌ها و روش‌های یک شیء را در هر قسمتی از برنامه خود در صورت دسترسی به شیء داشته باشیم. همچنین از نماد نقطه برای دسترسی به ویژگی‌ها و روش‌های یک شیء در داخل کلاس خود استفاده میکنیم.

نماد نقطه جدا کننده‌ای است که شیء را از ویژگی‌ها و روش‌های آن جدا می‌کند؛ زیرا این کار به ما کمک می‌کند تا کد را بیشتر خوانا و قابل فهم‌تر کنیم. این یک توافق عمومیِ استفاده شده در بسیاری از زبان‌های برنامه‌نویسی است.

Created by Alireza Boroon on 15 December 2023.

And please feel free to ask any questions or share your thoughts by contacting me.

@. Telegram.

Made by ❤