وراثت

- وراثت راهی برای تشکیل کلاس های جدید با استفاده از کلاس هایی است که قبلاً تعریف شده اند
- کلاس های تازه تشکیل شده کلاس های مشتق شده نامیده می شوند، کلاس هایی که ما از آنها استخراج می کنیم کلاس های پایه نامیده می شوند
- مزایای مهم وراثت، استفاده مجدد از کد و کاهش پیچیدگی یک برنامه است
- کلاس های مشتق شده (فرزندان) عملکرد کلاس های پایه (اجداد) را override یا extend می دهند.

وراثت

Inheritance 1

```python
class Person:

    def __init__(self,first_name="",last_name=""):
        self.first_name = first_name
        self.last_name = last_name

    def hello(self,fname,lname):
        self.first_name = fname
        self.last_name = lname
        print(f"Hello {fname} {lname}")
```

وراثت

Inheritance 2

```
class Student(Person):
    def __init__(self,role="student",score=None):
        self.role = role
        self.score = score

    def avrage(self,*args):
        self.score = args
        result = 0
        for num in args:
            result += num
        print(f"avrage = {result/len(args)}")
```

چندریختی

- میدانیم توابع می‌توانند آرگومان‌های مختلفی را دریافت کنند، متدها متعلق به اشیایی هستند و روی آنها عملیاتی را اجرا می‌کنند.
- در پایتون، چندریختی به روشی اشاره دارد که در آن کلاس‌های شیء مختلف می‌توانند نام متد یکسانی را به اشتراک بگذارند، و آن متدها را می‌توان از یک مکان فراخوانی کرد، حتی اگر اشیاء مختلفی به آن منتقل شوند
- abstract class کلاسی است که هرگز انتظار ندارد نمونه سازی شود. به عنوان مثال، ما هرگز یک شی Person نخواهیم داشت، فقط اشیاء Student و Teacher را خواهیم داشت که از کلاس Person مشتق شده اند

چندریختی

Polymorphism 1

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

    def hello(self):
        return self.name + " is student"


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

    def hello(self):
        return self.name + " is teacher"

ali = Student("Ali")
reza = Teacher("Reza")
```

چندریختی

Polymorphism 2

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

    def hello(self):
        pass


class Student(Person):

    def hello(self):
        return self.name + " is student"


class Teacher(Person):

    def hello(self):
        return self.name + " is teacher"
```

متدهای خاص

- توسط متدهای خاص میتوان عملیات های خاصی را در کلاس ها پیاده سازی کرد
- این متدها در واقع مستقیماً فراخوانی نمی‌شوند، بلکه توسط syntax زبان پایتون فراخوانی می‌شوند

special method 2

```
book = Book("Python Crash Course", "Eric Matthes", 546)

#Special Methods
print(book)
print(len(book))
del book
```

متدهای خاص

special method

```
class Book:
    def __init__(self, title, author, pages):
        print("A book is created")
        self.title = title
        self.author = author
        self.pages = pages

    def __str__(self):
        return f"Title: {self.title}, author: {self.author}, pages: {self.pages}"

    def __len__(self):
        return self.pages

    def __del__(self):
        print("A book is destroyed")
```