فى الدرس دا هنتكلم عن ال **Abstract base Classes** واختصارا **ABCs**.

دا نوع من ال classes بيرغم اي class هيورث منه انه يمشي على pattern محدد والا مش هيسمحله انه يورث منه.

طيب اولا انا هيكون عندي class وهحدده انه abstract class وكمان هخلي methods جواه تكون abstract وبكدا هيكون ال abstract class بيقول لل class اللى هيورث منه لازم تعمل ال method دي عندك علشان اسمحلك بانك تورث مني.

علشان احدد ان ال class يكون abstract هستدعي الاول **ABCMeta** و **abstractmethod** من موديول **abc**، بعد كدا بين اقواس ال class هكتب **metaclass=ABCMeta**، وفوق كل method عايزها تكون abstract هكتب decorator اسمه **@abstractmethod**.

لسه مش فاهم ؟ بص ع الكود، انا عندي class اسمه programming، دا حددته انه يكون abstract class وحددت جواه اتنين methods يكونوا abstract وواحده عاديه، بعد كدا جه ورث منه اتنين class اسمهم python و pascal، ف programming قال ل python و pascal علشان اسمحلكوا تورثوا مني لازم الاتنين methods اللى عندي اللى محددهم abstract تعملوهم عندكوا والتالته براحتكم سواء لو عايزين تعملوها او لا، غير كدا مفيش وراثه.

فانا جوا pascal عرفت الاتنين methods اللى متحددين يكونوا abstract جوا programming اللى pascal وارث منه، فكدا pascal سمحله بالوراثه عادي.

فى ال class اللى اسمه python بقا عرفت method واحده من الاتنين فلما جيت اعرف منه object اداني error علشان انا معملتش ال method التانيه اللى هي شرط اساسي اني اعملها علشان اقدر اورث.

```
# ----------------------------------------------------------------
# -- Object Oriented Programming => ABCs => Abstract Base Class --
# ----------------------------------------------------------------
# - Class Called Abstract Class If it Has One or More Abstract Methods
# - abc module in Python Provides Infrastructure for Defining Custom Abstract Base Cl
asses.
# - By Adding @absttractmethod Decorator on The Methods
# - ABCMeta Class Is a Metaclass Used For Defining Abstract Base Class
# ----------------------------------------------------------------

from abc import ABCMeta, abstractmethod

class Programming(metaclass=ABCMeta):

    @abstractmethod
    def has_oop(self):

        pass

    @abstractmethod
    def has_name(self):

        pass
```

```python
    def has_life(self):

        pass

class Python(Programming):

    def has_oop(self):

        return "Yes"

class Pascal(Programming):

    def has_oop(self):

        return "No"

    def has_name(self):

        return "Pascal"

one = Pascal()
print(one.has_oop())
print(one.has_name())

two = Python()
# print(two.has_oop())
# print(one.has_name())
```

CODE

```
No
Pascal
Traceback (most recent call last):
  File "c:\Users\Muhammad\Documents\Python Course\first.py", line 48, in <module>
    two = Python()
TypeError: Can't instantiate abstract class Python with abstract method has_name
```

OUTPUT