بسم الله الرحمن الرحيم

الوراثة المتعددة (Multiple Inheritance) في بايثون

الوراثة المتعددة تعني أن الصنف (Class) يمكنه أن يرث الخصائص والدوال من أكثر من صنف أب في نفس الوقت. هذا يشبه أن ترث صفات من والديك وأجدادك معاً. بايثون تدعم هذه الميزة، مما يسمح ببناء أصناف معقدة تجمع سلوكيات متعددة.

(MRO) آلية البحث عن الدوال

عندما يكون هناك دالتان بنفس الاسم في الأبوين، تستخدم بايثون خوارزمية تسمى Method Resolution Order (MRO)لتحديد أي دالة تُستدعى. ببساطة، تبحث في الأبناء أولاً، ثم في الآباء حسب الترتيب المكتوب في تعريف الابن. يمكنك رؤية هذا الترتيب باستخدام .ClassName._mro_ .

مثال توضيحي:

Python

```
class Product:
    def _init_(self, name, price, category):
        self.name = name
        self.price = price
        self.category = category

    def display_info(self):
        print (f"المنتج: {self.name}")
        Print (f"السعر: {self.price} ريال")
        Print (f"التصنيف: {self.category}")
```

```
    def apply_discount(self, discount_percent):
        self.price = self.price * (1 - discount_percent / 100)
        print (f"تم تطبيق خصم {discount_percent}%. السعر الجديد: {self.price:.2f} ريال")

class Shippable:
    def _init_(self, weight, length, width, height):
        self.weight = weight
        self.length = length
        self.width = width
        self.height = height

    def calculate_shipping_cost(self, cost_per_kg=5, cost_per_volume=0.1):
        volume = self.length * self.width * self.height
        shipping_cost = (self.weight * cost_per_kg) + (volume * cost_per_volume)
        return round(shipping_cost, 2)

    def display_shipping_info(self):
        print (f"الوزن: {self.weight} كجم")
        Print (f"الأبعاد: {self.length} × {self.width} × {self.height} سم")

#الابن: منتج قابل للشحن) يرث من Product و(Shippable
class ShippableProduct(Product, Shippable):
    def _init_(self, name, price, category, weight, length, width, height, sku):
```

```
        Product._init_(self, name, price, category)
        Shippable._init_(self,  weight, length,  width, height)
        self.sku = sku

    def display_full_info(self):
        self.display_info()          # من Product
        self.display_shipping_info()  # من Shippable
        print(f"رمز SKU: {self.sku}")
        Print (f"تكلفة الشحن التقديرية: {self.calculate_shipping_cost()} ريال")

laptop = ShippableProduct(
    name="لابتوب XPS 15",
    price=4500,
    category="إلكترونيات",
    weight=2.5,
    length=35,
    width=24,
    height=2,
    sku="LPT-XPS-15-2026"
)

laptop.display_info()
print ("-" * 30)
```

```
laptop.apply_discount(10)  # 10 خصم%

print ("-" * 30)


laptop.display_shipping_info()

print ("-" * 30)


laptop.display_full_info()
```

الحزم والمكتبات المضمنة(Built-in Packages & Modules)

عند تثبيت بايثون على جهازك، تأتي معه مجموعة كبيرة من الحزم (Packages) والوحدات (Modules)الجاهزة للاستخدام مباشرة. نطلق عليها "مضمنة" لأنها جزء من لغة بايثون نفسها ولا تحتاج إلى تثبيت إضافي.

·الوحدة :(Module) هي ملف بايثون يحتوي على دوال، كلاسات، ومتغيرات يمكنك استيرادها والاستفادة منها. مثال : math.pyهي وحدة تقدم دوال رياضية.

·الحزمة :(Package) هي مجلد يحتوي على مجموعة من الوحدات المنظمة، وغالباً ما يتضمن ملف .__init__.py مثال: حزمة datetime تحتوي على وحدات للتعامل مع التواريخ والأوقات.

باستخدام هذه المكتبات الجاهزة، يمكنك إنجاز مهام معقدة بكتابة أسطر قليلة من الكود، بدلاً من برمجة كل شيء من الصفر.

أهم الوحدات المضمنة مع شرح مبسط لكل منها

1 وحدة – math العمليات الرياضية.

توفر دوال رياضية متقدمة مثل الجذر التربيعي، الدوال المثلثية، اللوغاريتمات، والثوابت الرياضية.(π, e)

.مثال: حساب مساحة دائرة نصف قطرها 5

python

```
import math

r = 5

area = math.pi * r ** 2

print ("المساحة:", area)
```

.2وحدة – random الأرقام العشوائية

تستخدم لإنشاء أرقام عشوائية، اختيار عناصر عشوائية من قائمة، أو خلط قائمة.

مثال: محاكاة رمي عملة معدنية.

python

```
import random

result = random.choice(['صورة', 'كتابة'])

print ("نتيجة الرمي:", result)
```

3 وحدة – datetime التعامل مع الزمن.

تتيح إنشاء التواريخ والأوقات، إجراء العمليات الحسابية عليها، وتنسيقها.

مثال: حساب عدد الأيام بين تاريخين.

python

```
import datetime

start = datetime.date(2025, 1, 1)

end = datetime.date(2026, 2, 20)

delta = end - start

print ("الأيام بين التاريخين:", delta.days)
```

4 وحدة – os التفاعل مع نظام التشغيل.

توفر دوالاً للتعامل مع الملفات والمجلدات، متغيرات البيئة، وتنفيذ أوامر النظام.

مثال: معرفة مسار المجلد الحالي.

python

```
import os

print("المجلد الحالي:", os.getcwd())
```

5 وحدة – json التعامل مع بياناتJSON .

تستخدم لتحويل البيانات بين صيغة) JSON نص) وهياكل بايثون (قاموس، قائمة).

مثال: تحويل قاموس بايثون إلى نصJSON .

python

```
import json

", "age": 30}أحمدdata = {"name": "

json_str = json.dumps(data, ensure_ascii=False)

print(json_str)
```

6 وحدة – re التعبيرات النمطية.

تسمح بالبحث عن أنماط محددة داخل النصوص واستبدالها.

مثال: التحقق من صيغة البريد الإلكتروني.

python

```
import re

email = "test@example.com"

if re.match(r'^[\w\.-]+@[\w\.-]+\.\w+$', email):

 print ("بريد إلكتروني صالح")
```

7 وحدة – statistics الإحصاء.

توفر دوالاً إحصائية بسيطة مثل المتوسط، الوسيط، الانحراف المعياري.

مثال: حساب متوسط درجات طلاب.

python

```
import statistics

grades = [85, 90, 78, 92, 88]

avg = statistics.mean(grades)

print ("المتوسط:", avg)
```

8 وحدة – sys متغيرات ووظائف النظام.

تتيح الوصول إلى معاملات سطر الأوامر، الخروج من البرنامج، والتعامل مع مسار البحث عن الوحدات.

مثال: قراءة معاملات سطر الأوامر.

python

```
import sys

print ("عدد المعاملات:", len(sys.argv))

Print ("المعاملات:", sys.argv)
```

فوائد المكتبات المضمنة

.توفير الوقت والجهد لأنها توفر حلولاً جاهزة لمشاكل شائعة.

.موثوقة ومختبرة لأنها جزء من التوزيعة الرسمية لبايثون.

.متعددة الاستخدامات تغطي مجالات متنوعة: الرياضيات، النصوص، النظام، الشبكات، وغيرها.

.توثيق ممتاز حيث توفر بايثون شرحاً مفصلاً لكل وحدة في موقعها الرسمي.

خاتمة

تعد الوراثة المتعددة أداة قوية عند استخدامها بحذر، وتوفر المكتبات المضمنة زخماً هائلاً للإنتاجية. آمل أن يكون هذا الشرح والأمثلة قد أوفت بالمطلوب.