

مكتب التدريب المهني وترقية العمل

قسم هندسة البحث والتدريب



القطاع: الرقمي والذكاء الاصطناعي

دليل الدورة

M103: البرمجة الموجهة بالموضوع

السنة الأولى -

قطاع :

تطوير
رقمي
(جذع مشترك)





فهرس



01 - فهم نموذج OOP

- أعرض OOP
- تحديد فئة
- إنشاء كائن
- تعرف على التغليف
- التلعب بالأساليب

02 - تعرف على الأساسيةات

- ركائز OOP
- تعريف الميراث
- تعريف تعدد الأشكال
- تميز التجريد
- التعامل مع الواجهات

03 - الحلول الموجهة للكائنات البرمجية

- كود حل موجه للكائنات
- معالجة البيانات
- استخدم التعبيرات العادية
- إدارة الاستثناءات

04 - التعامل مع الوحدات والمكتبات

- لتعامل مع الوحدات
- لتعامل مع المكتبات

طرق التدريس



1

الدليل لـ

فوج الإيتاريين المهني المصطلح المترافق

يتم نشر نسخة PDF



2

نسخة PDF

يمكن تحميله على

المتعلمين والمدرسين

WebForce Life



3

محفوبيات قابلة للتثبيت

يمكن تنزيل أوراق

الملخص أو التمارين من

WebForce Life



4

نسخة PDF

يتم نشر نسخة

عبر الإنترنت في منطقة

المتعلمين والمدرسين

WebForce Life



5

الموارد متصلة

الموارد هي

متاحة بشكل متزامن

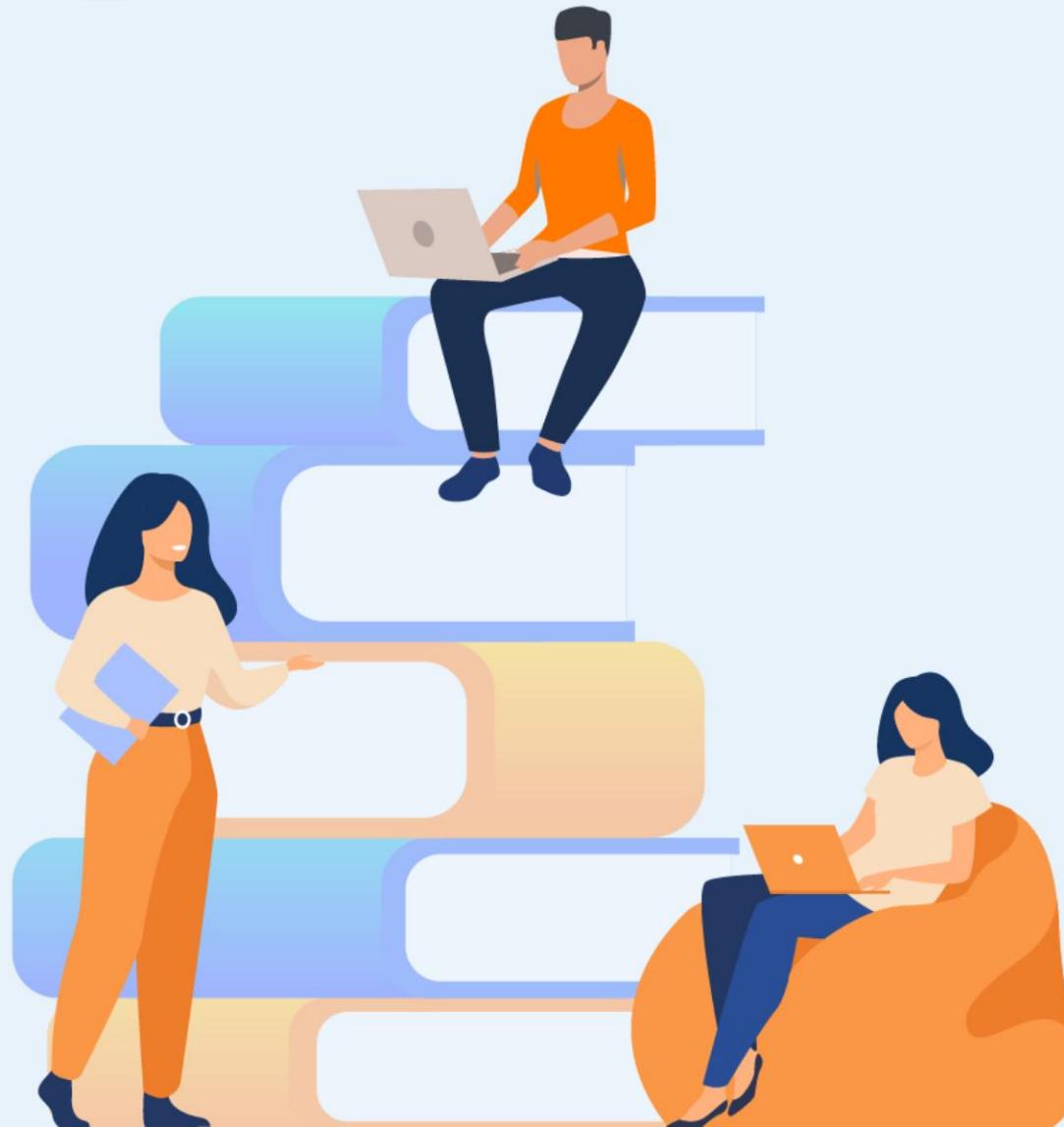
وغير متزامن للتكييف مع

وتيرة التعلم



الجزء 1

فهم نموذج OOP



في هذه الوحدة، سوف تقوم بما يلي:

- فهم مبدأ OOP
- إتقان تعريف الفصل
- إتقان مفهوم الكائن
- معرفة مبدأ التغليف. • معرفة كيفية التعامل مع طرق التغليف



05 ساعات



الفصل 1

أعرض OOP

ما ستعلم في هذا الفصل:

- اكتساب فهم لأسلوب OOP
- التعرف على مبدأ OOP. ذكر مميزاته مقارنة بنماذج البرمجة الأخرى



ساعة 01



الفصل 1

أعرض OOP

1. مقدمة في البرمجة الشيئية

2. تاريخ موجز لتطور لغات البرمجة الشيئية

3. معرفة مميزات OOP مقارنة بالنماذج الأخرى

OOP - 01 تقديم

مقدمة في البرمجة الشيئية



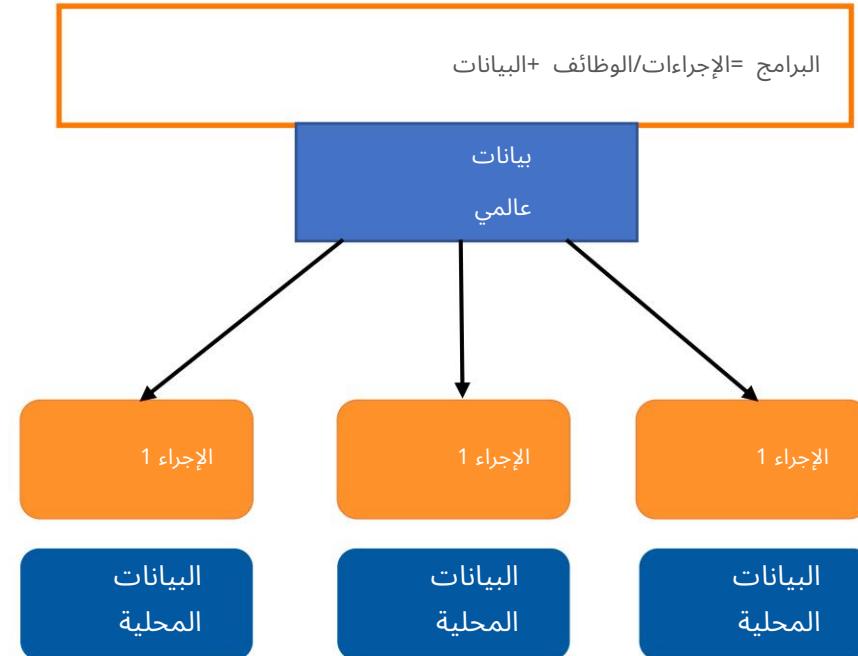
البرمجة الإجرائية

- في البرمجة الإجرائية، البرنامج مقسمة إلى كتل من التعليمات، تسمى الإجراءات أو المهام.

- ثم، لحل كل مشكلة، واحدة أو يتم استخدام العديد من الإجراءات/الوظائف

- في البرمجة الإجرائية والبيانات و يتم فصل معالجة هذه البيانات

البرامج = الإجراءات/الوظائف + البيانات



OOP - 01 تقديم

مقدمة في البرمجة الشيئية



البرمجة الإجرائية

في البرمجة الإجرائية:

- يقوم البرنامج بإنشاء البيانات

- إجراءات ووظائف معالجة هذه البيانات. • البرمجة في هذه الحالة بلغت :

- تحديد عدد معين من المتغيرات من أنواع مختلفة (عدد صحيح، سلسلة أحرف، بنيات، جداول، إلخ)

- كتابة إجراءات أو المهام للتعامل معها

سلبيات:

- صعوبة إعادة استخدام الكود

- صعوبة صيانة التطبيقات الكبيرة

حساب مجموع الأرقام الأول
فار

ط,S: عدد صحيح

البدء /*بدء الإجراء*/

س:= 0

لأن: 1 = إلى 100 افعل

س:=س+1

EndFor

"مجموع أول 100 رقم هو: ;S"

النهاية /*نهاية الإجراء*/

ابداً /*بدء الخوارزمية*/

مجموع

النهاية /*خوارزمية النهاية*/



OOP - 01 تقديم

مقدمة في البرمجة الشيئية



البرمجة الشيئية

- هل الفصل (البيانات والإجراءات) مفيد؟
- لماذا نعطي الأولوية للإجراءات على البيانات، بدلاً من الوظائف؟
- لماذا لا نعتبر أن البرامج يجب أن تعامل فقط مع الكائنات التي تحتوي على ملف العمليات (الإجراءات) التي حددها (في هذه الكائنات)؟

- تقدم اللغات الموجهة للكائنات إجابات على هذه الأسئلة.
- وهي مبنية على نوع واحد من الكيان: هذه كائنات
- وبالتالي، فإن الكائن هو كيان يجمع بين الخصائص الثابتة والخصائص الديناميكية الأخرى .
- يتكون البرنامج من مجموعة من الكائنات لكل منها جزء إجراءات (معالجة) وجزء بيانات.

OOP 01 - مقدمة في البرمجة الشيئية



WEBFORCE
BE THE CHANGE



البرمجة الإجرائية

ماذا يجب أن يفعل النظام ككل؟



البرمجة الشيئية

ما هي الكيانات التي سيتعامل
معها برمجي؟

ما هي الوحدات التي يجب أن يتكون منها برمجي؟

OOP01 - مقدمة في البرمجة الشيئية



البرمجة الإجرائية



البرمجة الشيئية

ما هي الإجراءات التي ينبغي على
تنفيذها؟
الائتمان حساب؟
الخصم من الحساب؟

برنامج إدارة الحسابات البنوكية

ما هو مظهر (هيكل) الحساب البنكي؟



الفصل 1

أعرض OOP

1. مقدمة في البرمجة الشيئية

2. تاريخ موجز لتطور لغات البرمجة الشيئية

3. معرفة مميزات OOP مقارنة بالنماذج الأخرى

OOP01 - تاریخ موجز لتطور OOP



السبعينيات

الثمانينيات

التسعينيات

في الوقت الحاضر

• ظهرت مفاهيم OOP في أواخر السبعينيات.

• أولى لغات البرمجة الموجهة للكائنات كانت .

• سيمولا : (1966) كانت أول لغة تجمع بين البيانات والإجراءات.

• أضفى الطابع الرسمي على مفاهيم الموضوع والطبقة. Simula I (1972):

• تعريف مفهوم الموضوع. Smalltalk (1972):

OOP01 - تاریخ موجز لتطور OOP



السبعينيات

الثمانينيات

التسعينيات

في الوقت الحاضر

وبعد ما يزيد قليلاً عن عقد من الزمن (80) دقيقة، تم إصدار العديد من اللغات المعتمدة على OOP مثل:

- برج إيفل أنشأه برتراند ماير
- لغة C++ التي أنشأها بيارن ستروسرب
- Objective C، والذي يستخدم بشكل خاص في تطوير iOS.

OOP01 - تاریخ موجز لتطور OOP



السبعينيات

الثمانينيات

التسعينيات

في الوقت الحاضر

شهد يوم 23 مايو 1995 ميلاد لغة Java الشهيرة من قبل شركة Sun Microsystems. تم شراؤها في عام 2009 من قبل شركة أوراكل العملاقة.

OOP01 - تاریخ موجز لتطور OOP



السبعينيات

الثمانينيات

التسعينيات

في الوقت الحاضر

اليوم، لدينا الاختيار بين مجموعة واسعة من اللغات بناءً على مبادئ OOP مثل:

PHP 5، (من الإصدار

#ضد،

روبي،

بايثون، الخ.



الفصل 1

أعرض OOP

1. مقدمة في البرمجة الشيئية

2. تاريخ موجز لتطور لغات البرمجة الشيئية

3. معرفة مميزات OOP مقارنة بالنماذج الأخرى

01 - تقديم OOP

معرفة مزايا OOP مقارنة بالنماذج الأخرى



الغرض والفوائد من OOP

الهدف: تسهيل تصميم وتشغيل وصيانة البرامج الكبيرة.

المزايا: • النمطية: تمثل كل فئة وحدة مدمجة تحتوي على بيانات ومجموعة من السلوكيات. وهذا

يؤدي إلى تطبيقات أقل تعقيداً.

• إمكانية إعادة الاستخدام:

- يسهل إعادة استخدام مكونات البرنامج. على سبيل المثال، يمكن إعادة استخدام فئة ما في عدة برامج، دون الحاجة إلى تعريفها في كل مرة.

- إن تحديد علاقة الميراث بين الفئات يكون مدفوعاً في المقام الأول بالحاجة إلى تجنب تكرار التعليمات البرمجية

• تبسيط الصيانة وزيادة الإنتاجية .

• التجزيد:

- الكائنات هي كيانات مشابهة لاثنين نواجههما في العالم الحقيقي (حساب، طالب، شركة، وما إلى ذلك).

- يتم إخفاء التفاصيل غير المفيدة للمستخدم. هذا المكون مضمون من قبل OOP.

مثال: لتعيين درجة لطالب، ما عليك سوى إدخال الدرجة و اختيار الطالب المعنى.

يتم إخفاء كيفية تعيين درجة معينة لطالب معين عن المستخدم.



الفصل 2

تحديد فئة

ما سنتعلمه في هذا الفصل:

- فهم مفهوم الفصل
- معرفة كيفية تصميم نموذج للفصل الدراسي
- معرفة كيفية تحديد مكونات الفصل



الفصل 2

تحديد فئة

1.تعريف الفصل

2.نمذجة الفصل

3.مكونات الفصل

- 02-تعريف الفصل

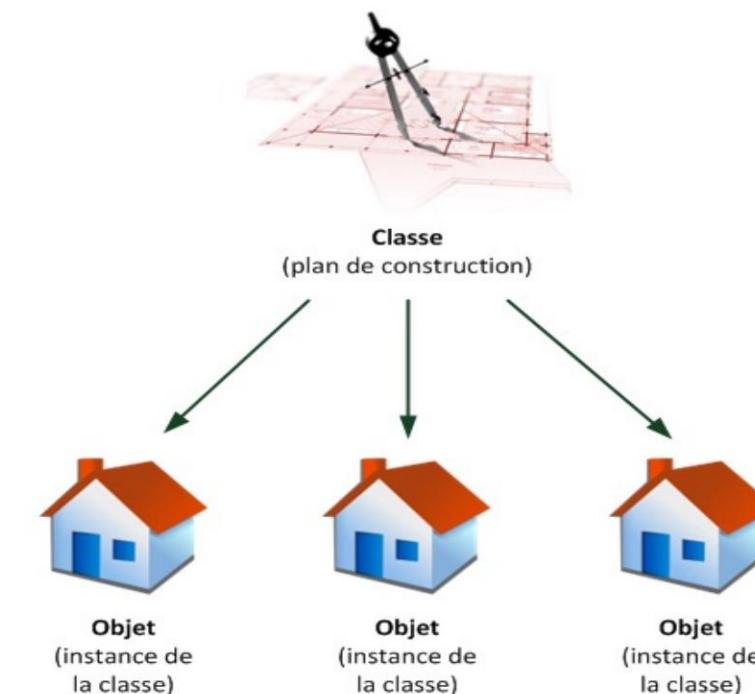
تعريف الطبقة



مفهوم الطبقة

- يمكن مقارنة الفصل بمفهوم النوع الذي نراه في لغات البرمجة الإجرائية.

- يعتبر الفصل بمثابة نموذج سيتم من خلاله إنشاء مجموعة من الكائنات. تحتوي هذه الكائنات على بيانات أو خصائص مشتركة ونفس السلوكيات.





الفصل 2

تحديد فئة

1.تعريف الفصل

2.نمذجة الفصل

3.مكونات الفصل

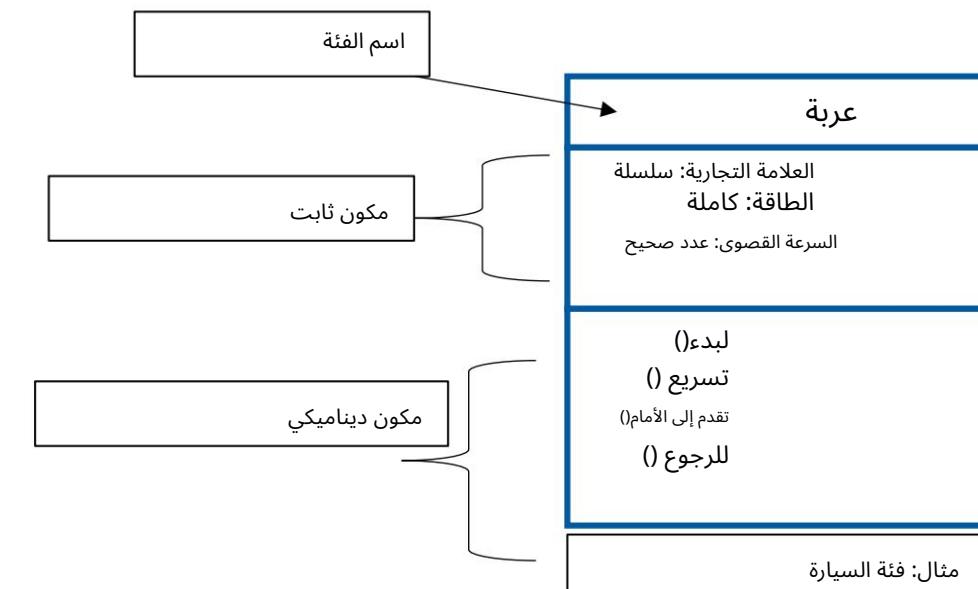
- 02-تعريف الفصل

نمذجة الطبقة



نمذجة الطبقة

- تتميز الطبقة بما يلي:
- اسم
- مكون ثابت: الحقول (أو السمات). وهي تميز حالة الكائنات أثناء تنفيذ البرنامج
- مكون ديناميكي: طرق تمثل سلوك كائنات هذه الفئة . إنهم يتعاملون مع مجالات الكائنات ويميزون الإجراءات التي يمكن أن تؤديها الكائنات





الفصل 2

تحديد فئة

- 1.تعريف الفصل
- 2.نمذجة الفصل
- 3.مكونات الفصل

- 02-تعريف الفصل

مكونات الصف



صف

- سمة تسمى أيضًا عضو الحقل أو البيانات يتواافق مع خاصية الفئة
- يتم تعريف السمة بواسطة:

• اسم.

• نوع البيانات • قيمة أولية (اختياري)

• يمكن أن تكون السمة من النوع:

- بسيطة: عدد صحيح، حقيقي، سلسلة أحرف، حرف، إلخ.
- كائن من نوع الفصل الدراسي: طالب، سيارة، إلخ.



سمات
نوع بسيط



- 02-تعريف الفصل

مكونات الصف



الوصول إلى السمات

- للوصول إلى سمة كائن ما، نشير إلى اسم مرجع الكائن متىوغاً باسم السمة في الكائن بالطريقة التالية:

اسم الكائن.
nameAttribute.

رؤية السمات

تحدد رؤية السمة حقوق الوصول إلى بيانات الفئة:

• يستطيع أي فصل الوصول إلى بيانات الفصل المحدد بمستوى الرؤية العامة.

• محمي: (#)

• الوصول إلى البيانات محجوز لطرق وراثة الطبقات

(سنرى لاحقاً في قسم التراث)

• خاص: (-)

• يقتصر الوصول إلى البيانات على أساليب الفصل نفسه

• اسم الكائن: اسم المرجع إلى الكائن

• اسم السمة
attributeName =

اسم_الفئة

- السمة: # النوع
+ السمة: 2 النوع
السمة: 3 النوع

+ الطريقة1)
- الطريقة2)

- 02-تعريف الفصل

مكونات الصف

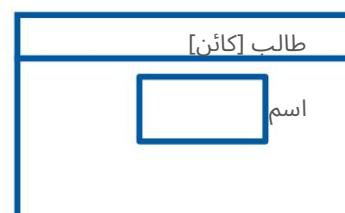


- قد يكون من الضروري تحديد سمة يتم مشاركة قيمتها بين جميع الكائنات **الموجودة في الفصل الدراسي**. نحن نتحدث عن سمات الطبقة

علاوة على ذلك، يتم تخزين هذه البيانات مرة واحدة فقط لمدة
جميع مثيلات الفصل
وصول:

منذ ذلك الحين، طريقة للفئة كما هو الحال بالنسبة لأي سمة أخرى

- عن طريق مثيل للفئة
- استخدام اسم الفئة



- 02-تعريف الفصل

مكونات الصف



منشئ

- المنشئ هو أسلوب معين يتم استدعاؤه ضمنياً عند إنشاء كائن
- يسمح لك المنشئ بتهيئة بيانات الكائنات (السمات) الخاصة بالفئة التي يعتمد عليها
- يجب ألا يكون للمنشئ نوع إرجاع
- يمكن أن تحتوي الفئة على عدة مُنشئات، ولكن لا يمكن إنتاج كائن معين إلا بواسطة مُنشئ واحد

أنواع البناء

- المنشئ الافتراضي
- المنشئ الذي لا يحتوي على أي معلمات يسمى المنشئ الافتراضي
- إذا لم نقم بإنشاء مُنشئ، فسيقوم الفصل تلقائياً باستدعاء المنشئ الافتراضي عند إنشاء كائن
- منشئ المعلمات:
 - يُطلق على المنشئ الذي يحتوي على معلمة واحدة على الأقل مُنشئ ذو معلمات
 - نسخ المنشئ:
 - المنشئ الذي يقوم بإنشاء كائن عن طريق نسخ البيانات من كائن آخر يسمى مُنشئ النسخ

- 02-تعريف الفصل

مكونات الصف



مدمرة

المدمر هو أسلوب خاص يسمح بتدمير كائن غير مرجعي.

- اللغات التي تستخدم جامعي البيانات المهملة (مثال بايثون) لا تقدم آلية التدمير لأن المبرمج لا يدير الذاكرة بنفسه
- يجب ألا يكون للمدمر نوع إرجاع

نتيج لك أداة التقطيع ما يلي:

- التعامل مع الأخطاء
- تحرير الموارد المستخدمة بطريقة معينة
- التأكد من إغلاق أجزاء معينة من الكود.

جامع البيانات المهملة هو برنامج لإدارة الذاكرة تلقائياً. وهي مسؤولة عن إعادة تدوير الذاكرة المخصصة مسبقاً ثم الذاكرة غير المستخدمة.



الفصل 3

إنشاء كائن



ما سنتعلمه في هذا الفصل:

- فهم مفهوم الكائن • إتقان مفاهيم إنشاء فئة وتدمير كائن



الفصل 3

إنشاء كائن

1.تعريف الكائن

2.المثيل

3.التدمير الصريح للكائن

03 - إنشاء كائن

تعريف الكائن

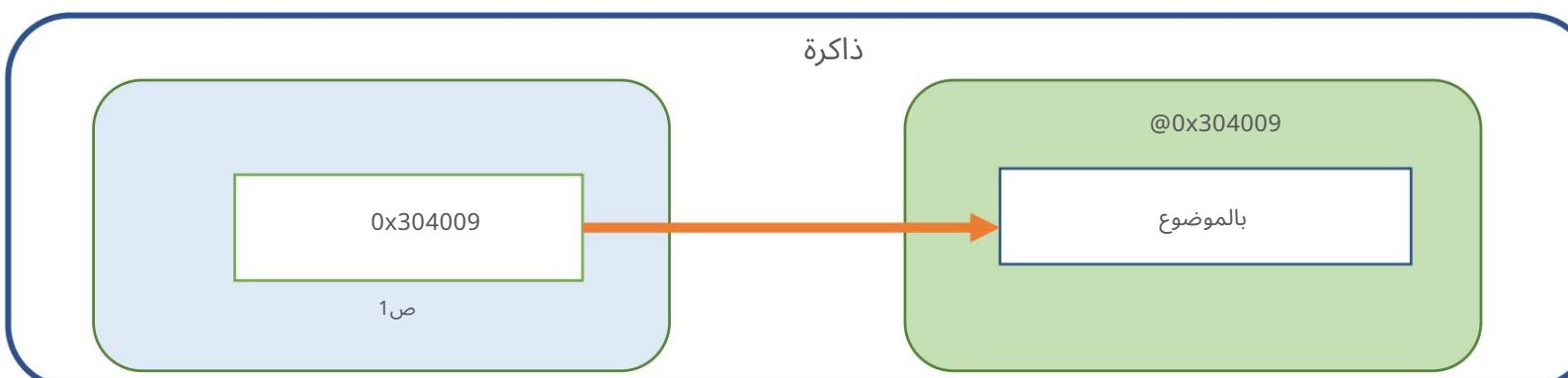


مفهوم الكائن

الكائن = المرجع + الحالة + السلوك

• يجب أن يكون لكل كائن اسم (مرجع) "فريد له" لتعريفه. • يوفر المرجع إمكانية الوصول إلى الكائن، ولكنه ليس الكائن نفسه. أنه يحتوي على عنوان الموقع

الذاكرة التي يتم تخزين الكائن فيها.



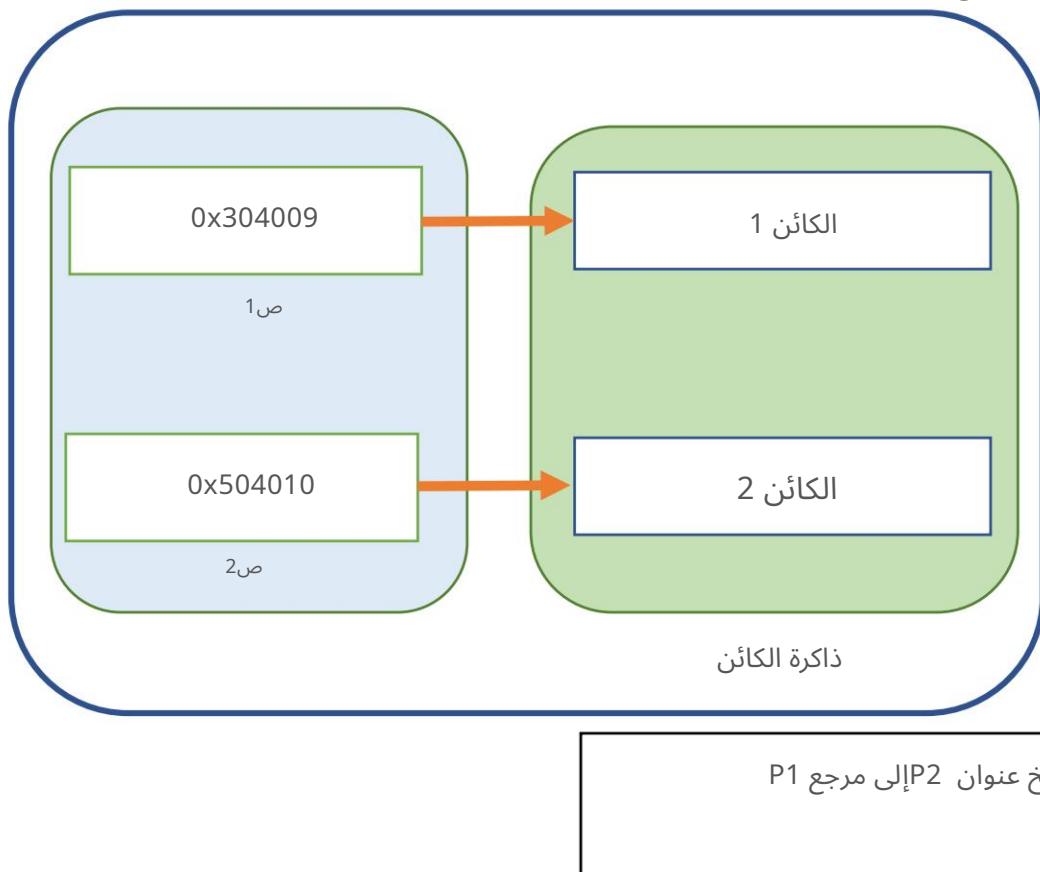
03 - إنشاء كائن

تعريف الكائن

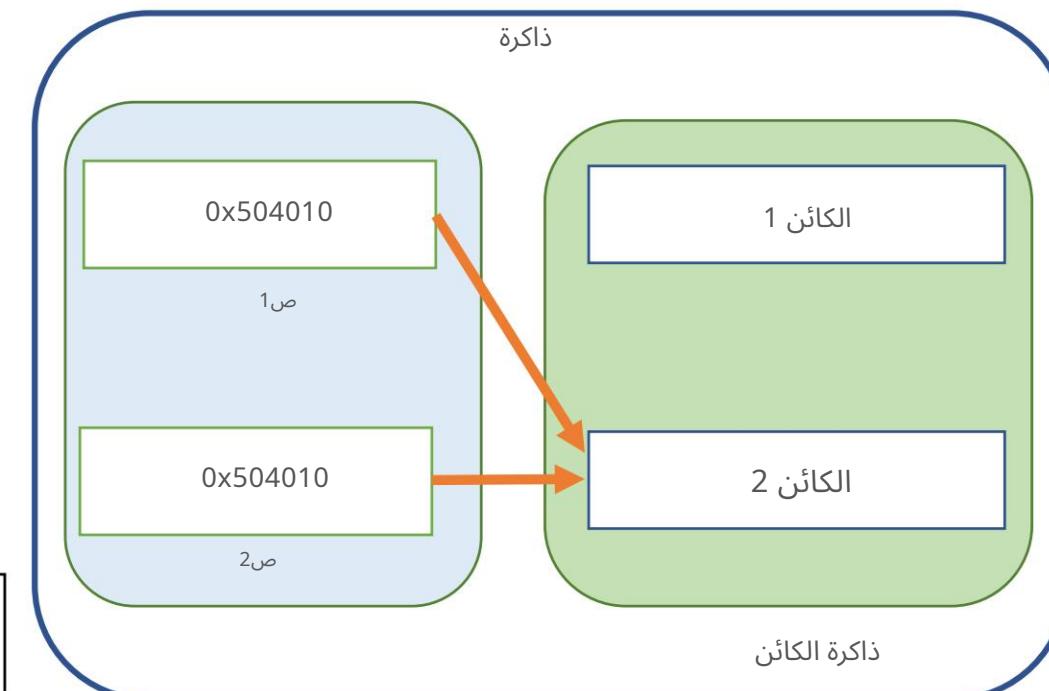


مفهوم الكائن

- يمكن لعدة مراجع إخالة نفس الكائن إلى العنونة غير المباشرة



ذاكرة

 $P1=P2$ 

ذاكرة

الكائن 1

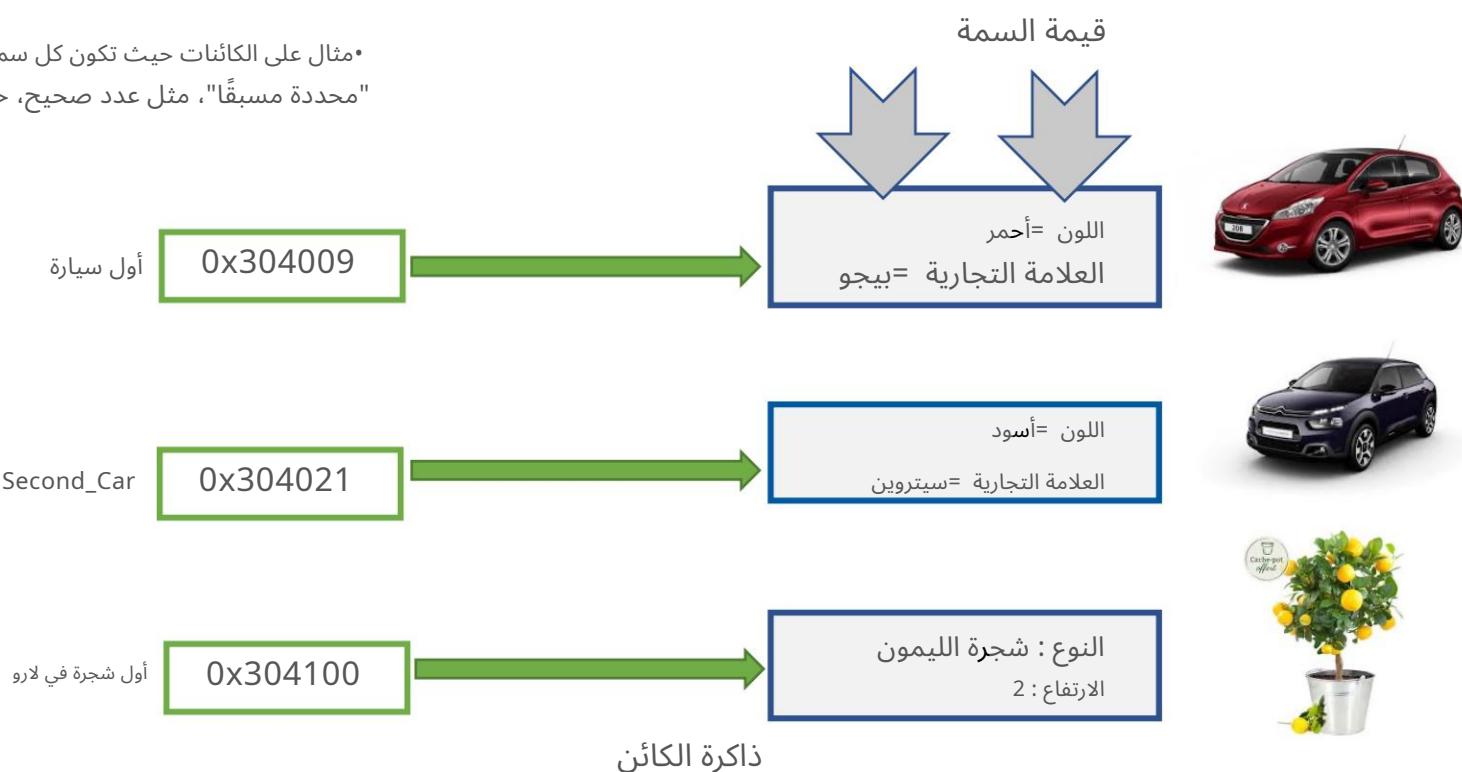
الكائن 2

ذاكرة الكائن

مفهوم الكائن

- الكائن قادر على حفظ الحالة، أي مجموعة من المعلومات في السمات
السمات هي كافة المعلومات المستخدمة لتمثيل حالة الكائن

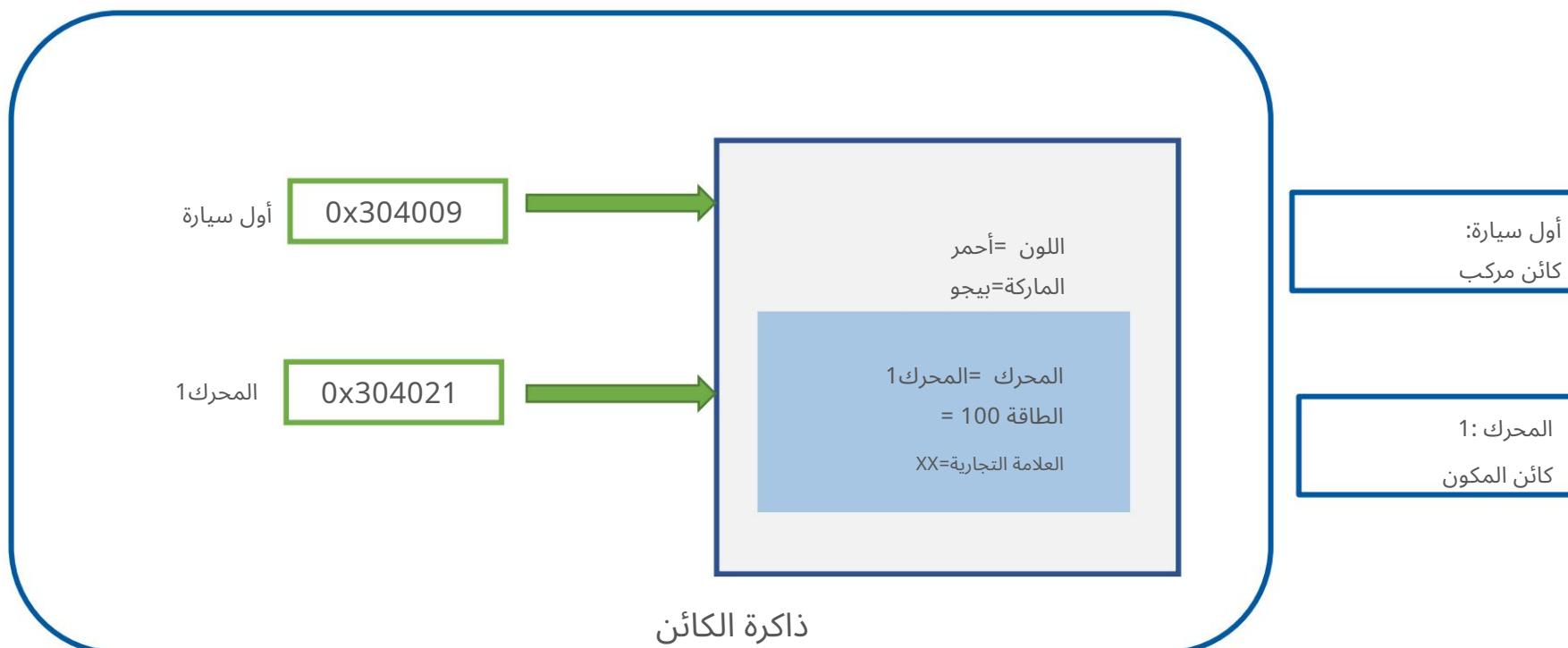
السلوك + الحالة + المرجع = الكائن



03 - إنشاء كائن

تعريف الكائن

يمكن وضع الكائن المخزن في الذاكرة داخل مساحة الذاكرة المخصصة لشخص آخر. هذا هو كائن مركب



03 - إنشاء كائن

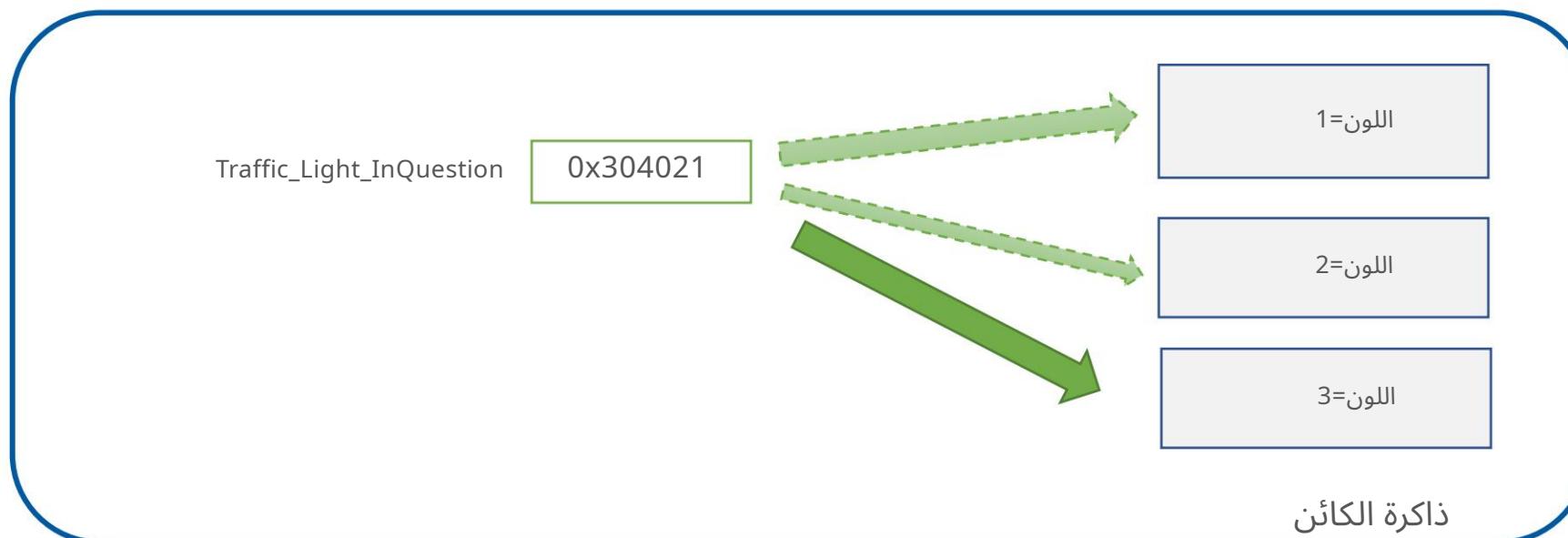
تعريف الكائن



*تغيير حالة الكائنات

*تقتصر دورة حياة الكائن على سلسلة من التغيرات في الحالة

*ضع في اعتبارك الكائن Feu_de_signalisation_EnQuestion الذي يأخذ لونه ثلاثة قيم

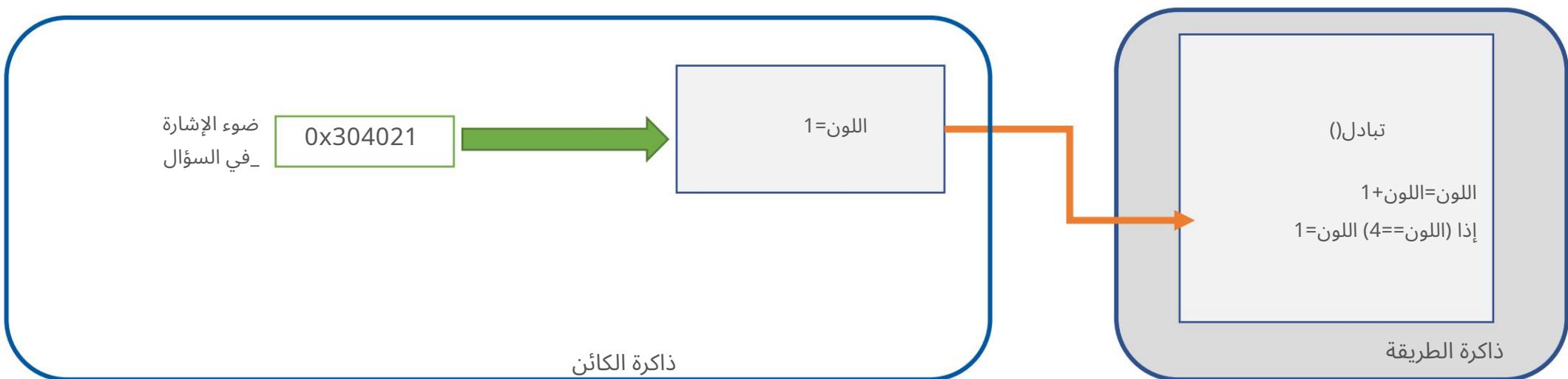


03 - إنشاء كائن

تعريف الكائن

الكائن = المرجع + الحالة + السلوك

ولكن ما هو سبب تغيرات الدولة؟
-طرق-





الفصل 3

إنشاء كائن

1.تعريف الكائن.

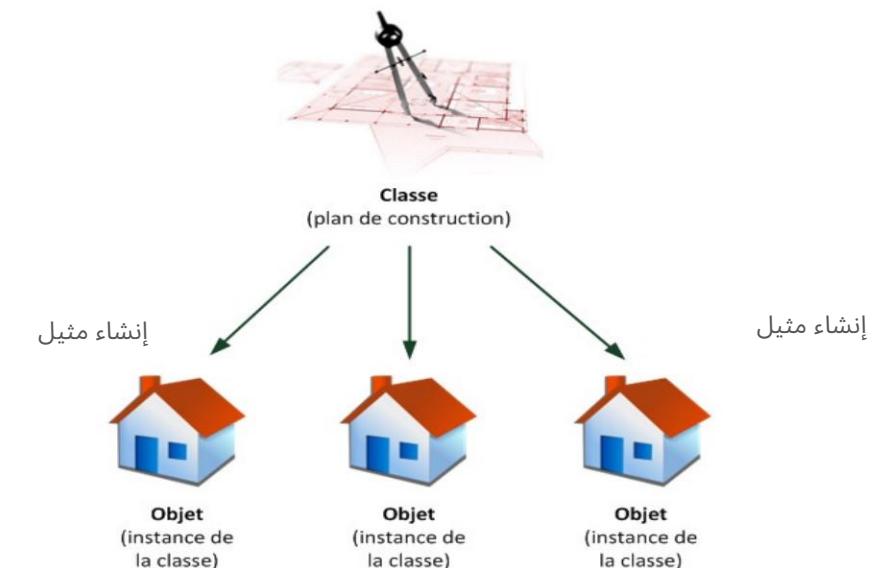
2.المثيل

3.التدمير الصريح للكائن

- 03 إنشاء كائن

إنشاء مثيل

- يتكون إنشاء كائن من مرحلتين:
- مرحلة الربع من الفصل: إنشاء الكائن في الذاكرة
- مرحلة من نطاق الكائن: تهيئة السمات
- يتم إنشاء مثيل صريح للكائن عبر المُنشئ. يتم استدعاؤه تلقائيًا عند إنشاء الكائن في الذاكرة.





الفصل 3

إنشاء كائن

1.تعريف الكائن.

2.المثيل

3.التدمير الصريح للكائن

03 - إنشاء كائن

التدمير الصريح للكائن



• تذكير: أداة التدمير هي طريقة معينة تسمح بتدمير كائن غير مرجعي

• يتم تنفيذ التدمير الصريح للكائن عن طريق استدعاء مدر المفأة



تعرف على التغليف



ما سنتعلم في هذا الفصل:

- فهم مبدأ التغليف
- تعرف على المعدلات والملحقات الخاصة بالفئة:
- بناء الجملة والمنفعة



تعرف على التغليف



1. مبدأ التغليف.
2. المعدلات والوصلات (الحروف، أدوات الضبط، وما إلى ذلك)

04- التعرف على التغليف

مبدأ التغليف



مبدأ التغليف

- التغليف هو عملية تجميع الكود (الطرق) + البيانات (السمات) داخل نفس الكيان (الكائن)

- يتكون التغليف من حماية المعلومات الموجودة في الكائن

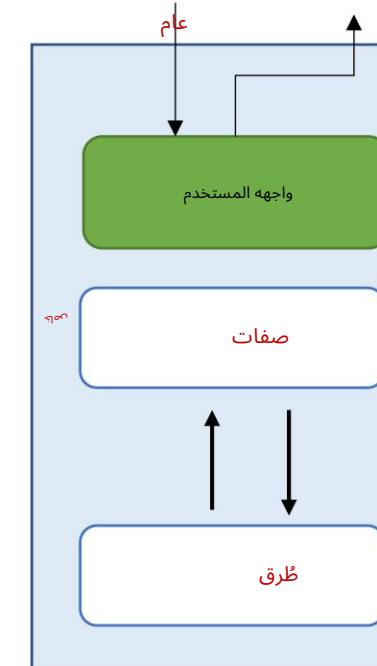
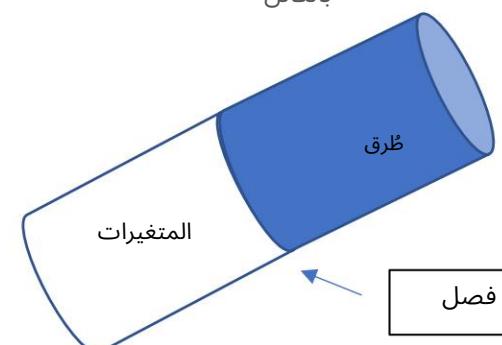
- لذلك من الممكن إخفاء معلومات كائن ما عن كائنات أخرى.

- يتكون التغليف وبالتالي من إخفاء تفاصيل تنفيذ كائن ما، من خلال تحديد واجهة. • الواجهة هي العرض الخارجي للكائن، وهي تحدد الخدمات التي يمكن الوصول إليها

(السمات والأساليب) لمستخدمي الكائن

• الواجهة = قائمة التوقيعات الخاصة بأساليب الوصول. • الواجهة = بطاقة العمل الخاصة

بالكائن



04- التعرف على التغليف

مبدأ التغليف



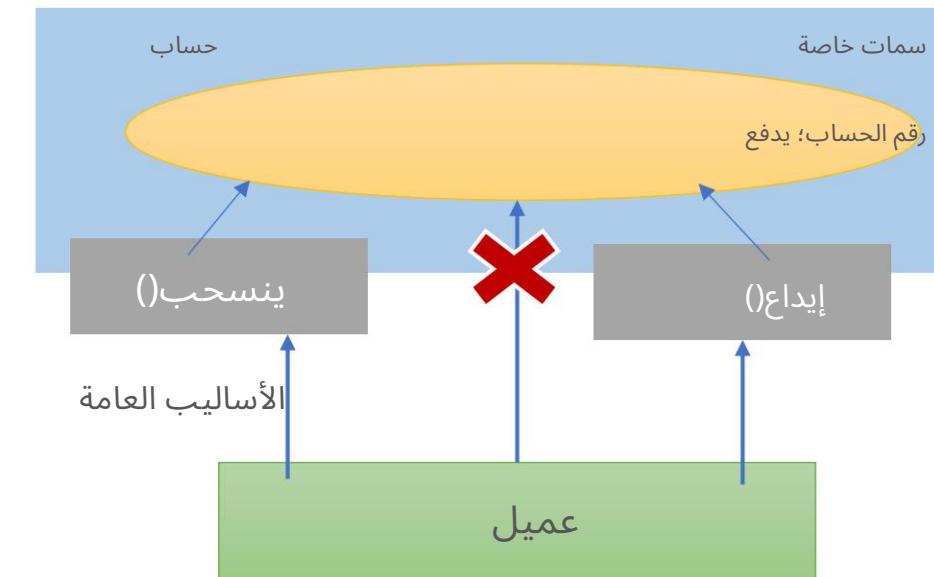
- الكائنات تقييد فقط وصولهم إلى أساليب صفهم

أنها تحمي سماتها • وهذا يسمح لك بالتحكم في كل عمليات الوصول

مثال :

- سمات فئة الحساب خاصة. وبالتالي، لا يمكن للعميل الوصول إلى رصيد الحساب إلا من خلال طريقي السحب () والإيداع () العامتين

لا يمكن للعميل تعديل رصيده إلا عن طريق إجراء معاملة إيداع أو سحب





تعرف على التغليف



1. مبدأ التغليف .2. المعدلات والوصلات (الحروف، أدوات الضبط، وما إلى ذلك)

04- التعرف على معدّلات التغليف وملحقاتها (الحروف، والمحددات، وما إلى ذلك)

المعدلات والملحقات

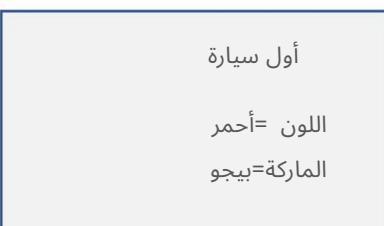
• للتفاعل مع سمات كائن من الخارج، ما عليك سوى إنشاء أساليب عامة في فئة تسمى **Accessors and Modifiers**

• **الموصيل (الخاصية)**: الأساليب التي ترجع قيمة سمة لكتاب ما (السمة خاصة بشكل عام) • الترميز المستخدم هو `getXXX` السمة التي تم إرجاعها



- **المعدلات (المحددات)**: الأساليب التي تقوم بتعديل قيمة إحدى سمات الكائن

الترميز المستخدم هو `setXXX` مع السمة المعدلة



الحصول على اللون ()

ما هو لون؟ Première_Voiture?

أحمر



اللون المحدد (أزرق)

اللون = أزرق





اللاعب بالأساليب



ما ستعلم في هذا الفصل:

- طرق التعامل: التعريف والدعوة
- التفريق بين أسلوب المثيل وأسلوب الفصل



الفصل 5

التلعب بالأساليب

- 1.تعريف الطريقة
- 2.رؤيه الطريقة
- 3.معلومات الطريقة
- 4.استدعاء الطريقة
- 5.طرق الفصل

- 05 التلاعب بالأساليب

تحديد الطريقة

تعريف الطريقة مشابه لتعريف الإجراء أو الوظيفة. وهو يتكون من رأس وكتلة.

يحدد الرأس ما يلي:

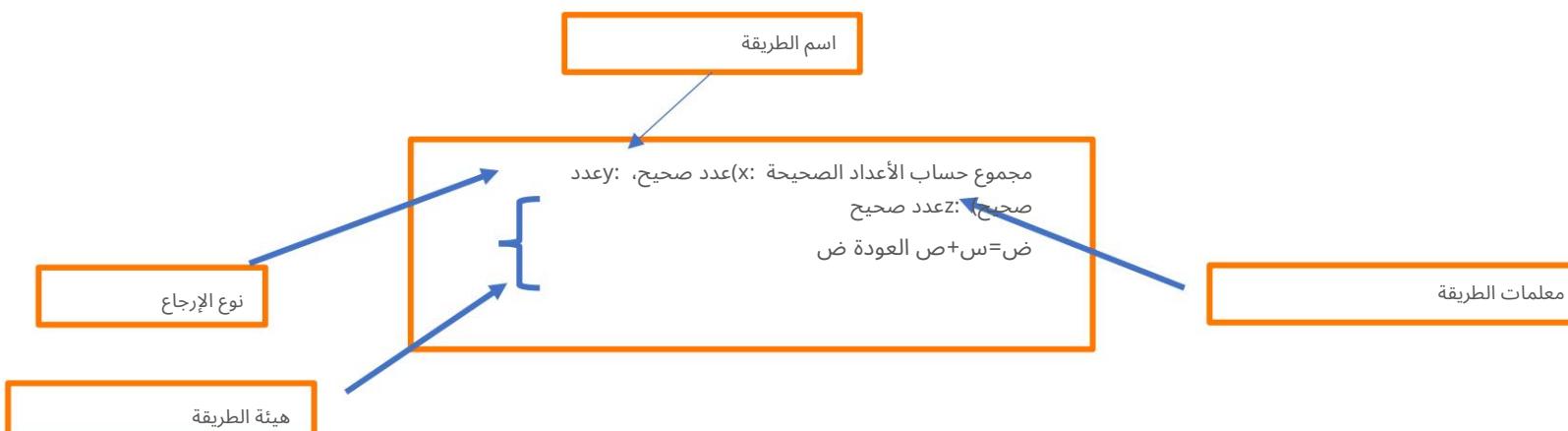
اسم

نوع من العودة

قائمة (ربما فارغة) بمعلومات الإدخال المكتوبة

ت تكون الكتلة من سلسلة من التعليمات (كتلة من التعليمات) التي تشكل نص الطريقة

مثال بناء الجملة:





الفصل 5

التلعب بالأساليب

- 1.تعريف الطريقة
- 2.رؤية الطريقة
- 3.معلومات الطريقة
- 4.استدعاء الطريقة
- 5.طرق الفصل

- 05 التلاعب بالأساليب

رؤية الطريقة

إن رؤية الطريقة تحدد حقوق الوصول التي تسمح باستدعاء أساليب الفئة

عامة (+):

- يُسمح باستدعاء الأسلوب للطرق من جميع الفئات

محمي (#):

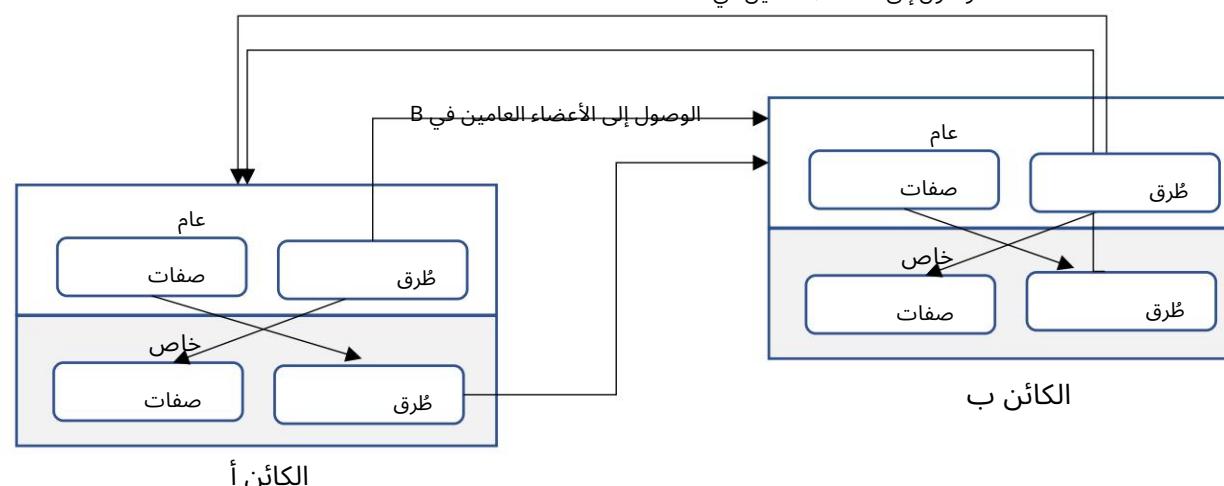
- استدعاء الأسلوب محظوظ لطرق وراثة الفئات

(سني لاحقاً في قسم التراث)

خاص (-):

- يقتصر استدعاء الأسلوب على أساليب الفئة نفسها

الوصول إلى الأعضاء العامين في A





الفصل 5

التلعب بالأساليب

- 1.تعريف الطريقة
- 2.رؤيه الطريقة
- 3.معلومات الطريقة
- 4.استدعاء الطريقة
- 5.طرق الفصل.

- 05 التلاعب بالأساليب

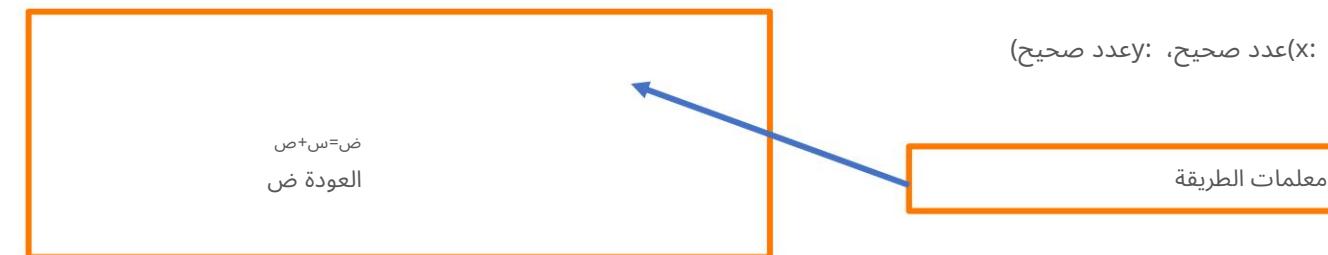
معلومات الطريقة



من الممكن تمرير الوسائط (وتسمى أيضًا المعلومات) إلى طريقة ما، أي تزويدها باسم متغير بحيث يمكن للطريقة إجراء عمليات على هذه الوسائط أو بفضل هذه الوسائط.

مجموع حساب الأعداد الصحيحة :
 ض = ص + ص
 العودة ض

مجموع حساب الأعداد الصحيحة :
 ض: عدد صحيح، : (عدد صحيح)





الفصل 5

التلعب بالأساليب

- 1.تعريف الطريقة
- 2.رؤيه الطريقة
- 3.معلومات الطريقة
- 4.استدعاء الطريقة**
- 5.طرق الفصل

- 05 التلاعب بالأساليب

استدعاء طريقة



لا يمكن تعريف الأساليب كمكونات للفئة.

• يتم استدعاء الأسلوب لكائن ما. تسمى هذه الطريقة طريقة المثيل.

• يتم استدعاء أسلوب لكائن ما عن طريق تسمية الكائن متبوعاً بنقطة متبوعة باسم الأسلوب وقائمة الوسائط الخاصة به.

اسم الكائن.طريقة الاسم (arg1, arg2,...)

: او

• اسم الكائن: اسم المرجع إلى الكائن

• اسم الطريقة: اسم الطريقة

Exemple : la classe Véhicule

Véhicule
<i>Marque :string</i>
<i>Puissance :integer</i>
<i>Vitesse-max :integer</i>
<i>Vitesse-courante :int</i>
<i>Démarrer()</i>
<i>Accélérer()</i>
<i>Avancer()</i>
<i>Reculer()</i>

٧ هو كائن نوع المركبة

(ابدأ.٧



الفصل 5

التلعب بالأساليب

- 1.تعريف الطريقة
- 2.رؤيه الطريقة
- 3.معلومات الطريقة
- 4.استدعاء الطريقة
- 5.طرق الفصل

- 05 التلاعب بالأساليب

طرق الصنف



- طريقة الفئة هي طريقة معلنة لا يعتمد تنفيذها على كائنات الفئة . نظرًا لأنها مرتبطة بالفئة وليس بمثيلاتها، فإننا نسبق اسم الطريقة باسم الفئة

```
classname.methodname(arg1, arg2,..)
```

مثال :



X

مجموع حساب الأعداد الصحيحة :X(عدد صحيح، :عدد صحيح)

X.حساب المجموع (2.5)

- بما أن أساليب الفصل تنتهي إلى الفصل، فلا يمكنها الوصول إلى سمات المثيل التي تنتهي إليها
تنتمي إلى مثيلات الفئة

- أساليب المثيل الوصول إلى سمات المثيل وأساليب المثيل

- أساليب مثيل الوصول إلى سمات الطبيعة وأساليب الطبيعة

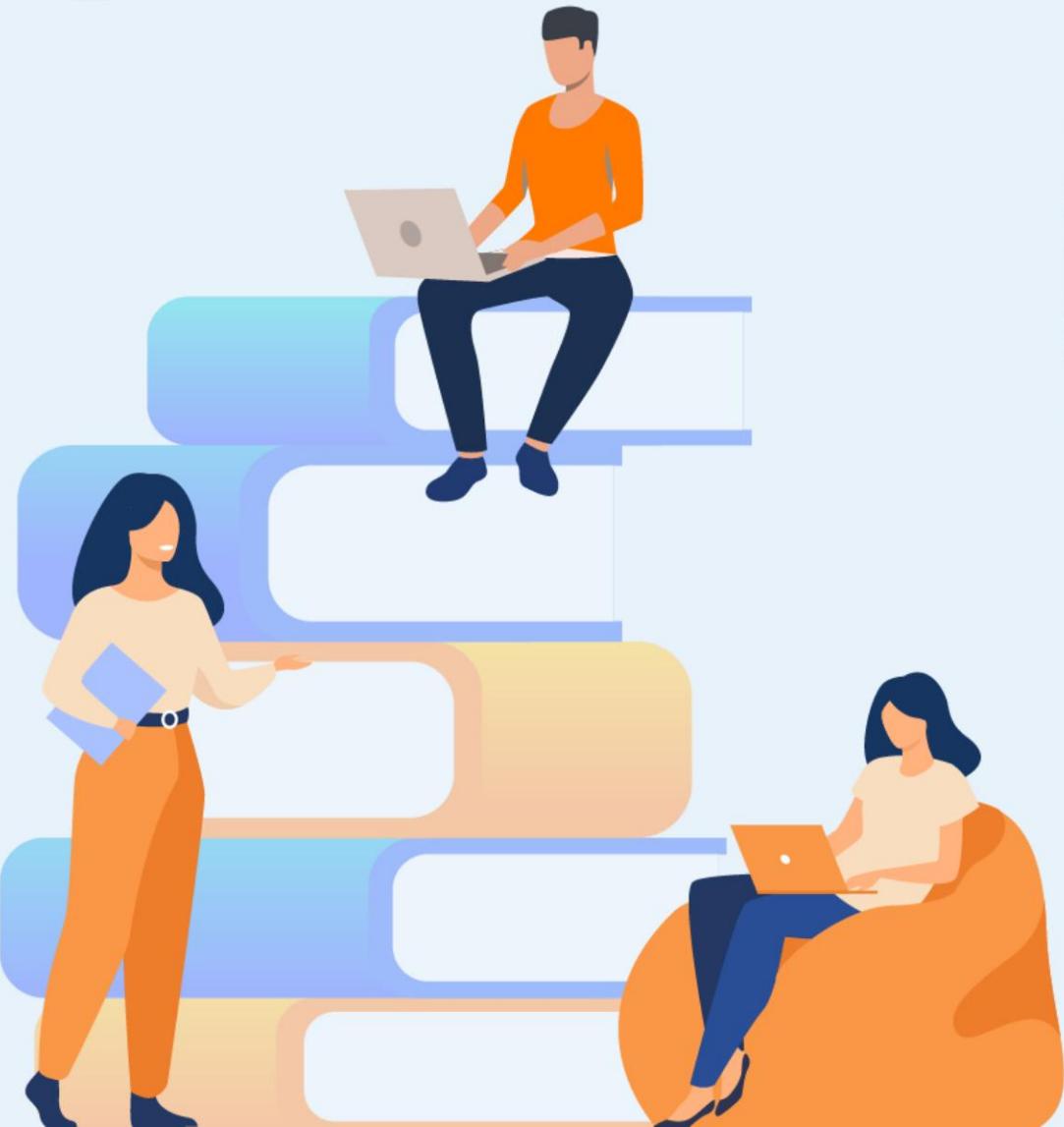
- أساليب فئة الوصول إلى سمات الطبيعة وأساليب الطبيعة

- لا تصل أساليب الفئة إلى سمات المثيل وأساليب المثيل



الجزء 2

تعرف على الركائز الأساسية لـ OOP



في هذه الوحدة، سوف تقوم بما يلي:

• إتقان مبدأ الوراثة في OOP • إتقان مفهوم تعدد الأشكال في OOP • إتقان مبدأ التجزير في OOP وفائدة

• التعامل مع الواجهات (التعريف والتنفيذ)



ساعة 15



تعريف الميراث



ما ستتعلم في هذا الفصل:

• فهم مبدأ الميراث والتمييز بين أنواعه • فهم مبدأ التحميل الزائد للمنشئ • تسلسل المنشئ الرئيسي

• التحكم في رؤية أعضاء فئة مشتقة



الفصل 1

تعريف الميراث

1. مبدأ الميراث. 2. أنواع الميراث. 3. التحميل الزائد للمنشئ والتسلاسل.

4. رؤية سمات وأساليب فئة الطفل.

01 - تعريف الميراث

مبدأ الميراث

مبدأ الميراث

الميراث هو المفهوم الذي يحدد علاقة التخصص أو التعميم بين الطبقات المختلفة (Exp: علاقـة توريـث بين فـئـة الموظـف والمـديـر وفـئـة الموظـف).

- عندما ترث الفئة "A" من الفئة "B":
- يمنح (أ) لنفسه جميع ممتلكات (ب) بالإضافة إلى ممتلكاته الخاصة.
- يعتبر A بمثابة تخصص لـ B. وفي الوقت نفسه، يمكن اعتبار B بمثابة تعميم لـ A.
- يُطلق على A فئة فرعية من ، أو فئة مشتقة من B.
- يُطلق على B الفئة الأصلية أو الفئة الفائقة لـ A.
- يمكن اعتبار أي مثيل (كائن) لـ A كائناً لـ B.
- لا يمكن دائماً اعتبار مثيل B كائناً لـ A.

01 - تعريف الميراث

مبدأ الميراث

مبدأ الميراث

مثال :

- النظر في تعريف الفئات الثلاث التالية: الشخص والطالب والموظف:



مشكلة :

الازدواجية رمز.

- يجب تكرار أي تعديل على سمة أو طريقة في كل فئة تم الإعلان عنها. إذا كنت ترغب في تعديل نوع سمة ، ExpBaissanne فيجب عليك القيام بذلك في كل فئة من الفئات الثلاثة.

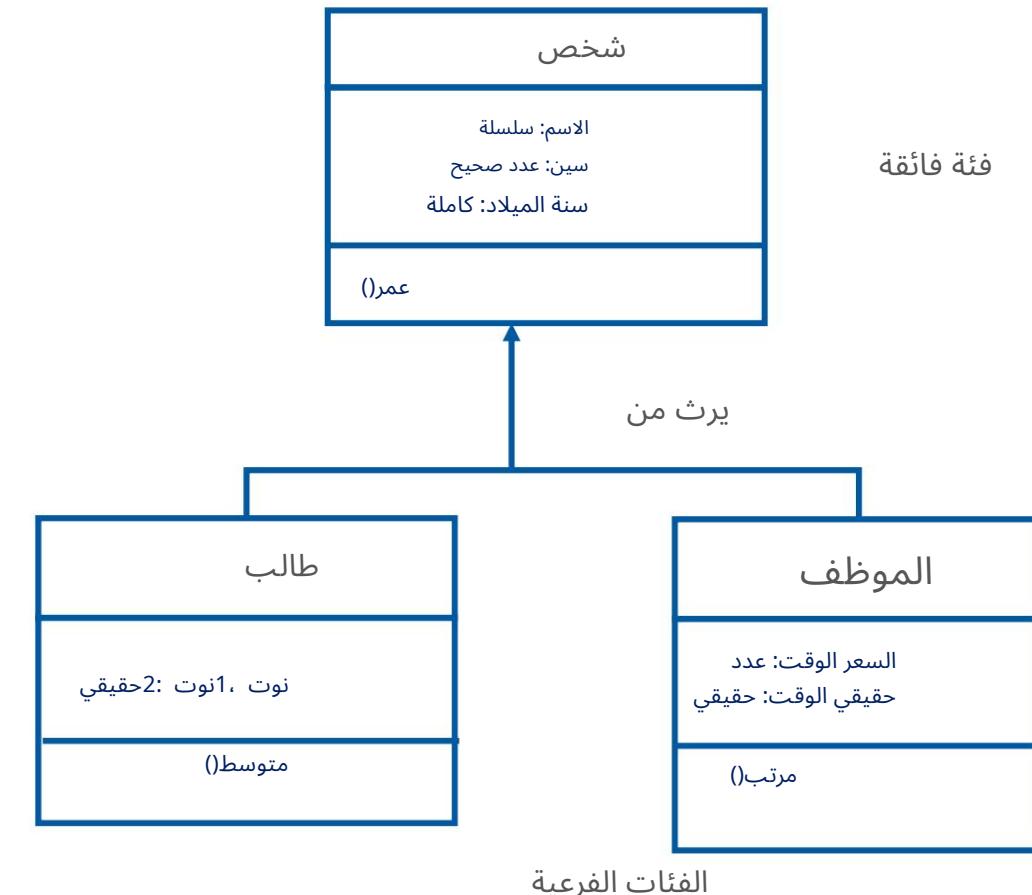
01 - تعريف الميراث

مبدأ الميراث



حل :

- وضع خصائص مشتركة مع كافة الفئات الأخرى في الفئة الأصل .
- نحن نحتفظ فقط في فصول الطفل بالسمات أو أساليب خاصة بهم.
- ترث الفئات المشتقة تلقائياً السمات (والطرق) التي لا تحتاج إلى إعادة كتابتها.



01 - تعريف الميراث

مبدأ الميراث



مصلحة في الميراث

- يجعل الوراثة من الممكن تحليل الكود عن طريق تجميع الخصائص المشتركة بين عدة فئات داخل فئة واحدة (الفئة الأم).
- أصبح إنشاء فئات جديدة أسهل بفضل التسلسل الهرمي للفئات المحدد جيداً.



الفصل 1

تعريف الميراث

1. مبدأ الميراث. 2. أنواع الميراث. 3. التحميل الزائد للمنشئ والتسلاسل

4. رؤية سمات وأساليب فئة الطفل

01 - تعریف المیراث

أنواع الميراث



تعدد الميراث

يمكن أن يرث الفصل من عدة فئات

مثال :

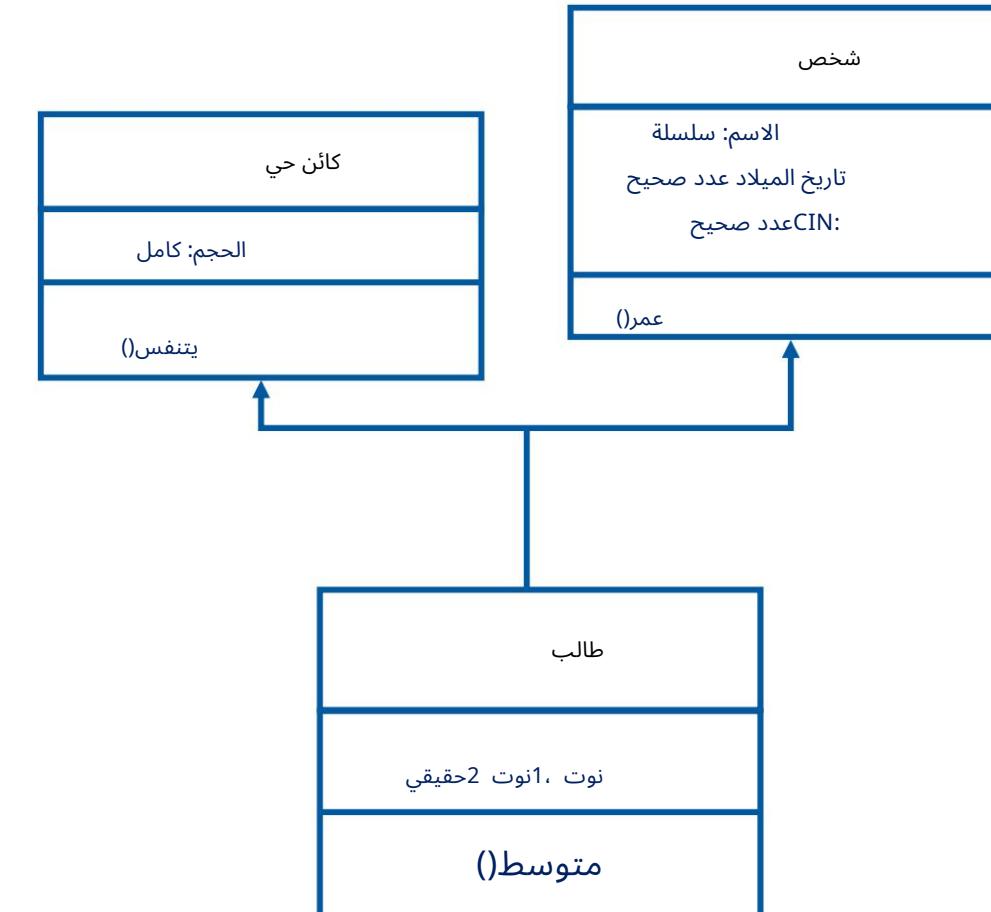
الطالب هو شخص وكائن حي في نفس الوقت

يرث فصل الطالب السمات والأساليب

التابع

فتين

ليصبح له سماته وأساليبه



- 01تعريف الميراث

أنواع الميراث

الميراث المتتالية

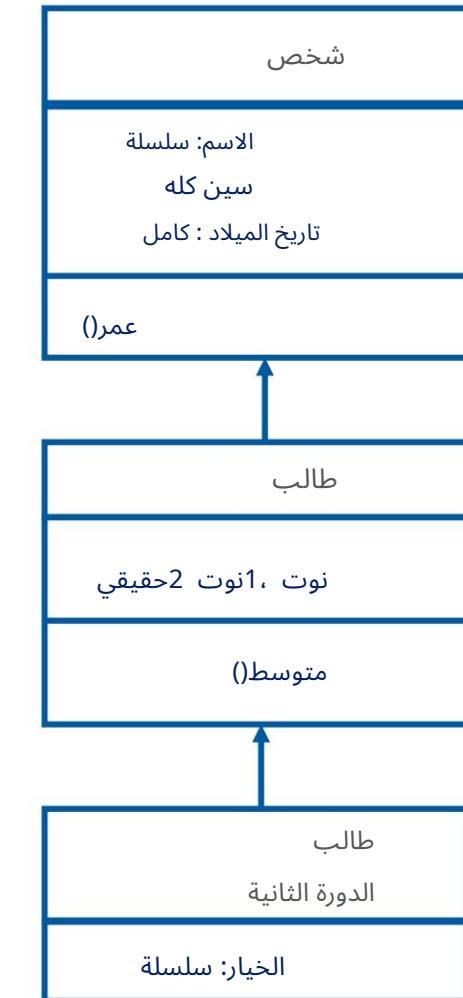
يمكن أن تكون الفئة الفرعية بحد ذاتها فئة فائقة.

مثال :

يرث الطالب من الشخص

يرث إلى StudentSecondCycle من Student

يرث من الشخص StudentSecondCycle





الفصل 1

تعريف الميراث

1. مبدأ الميراث .2. أنواع الميراث .3. التحميل الزائد **للمنشئ والتسلسل**

4. رؤية سمات وأساليب فئة الطفل

- 01تعريف الميراث

التحميل الزائد للمنشئ والتسلسل

التحميل الزائد للمنشئ

- يُطلق على المنشئين الذين يحملون نفس الاسم لكن التوقيع مختلف اسم المنشئين المحمليين بشكل زائد

- يتضمن توقيع المنشئ أنواع المعلمات وعدد المعلمات

- داخل الفصل الدراسي، يمكن أن يكون للمنشئين أعداد مختلفة من الوسائط، أو تسلسلات مختلفة من الوسائط، أو مختلفة أنواع الحجة

- يمكن أن يكون لدينا مُنشئات متعددة ولكن أثناء إنشاء كائن، يختار المترجم الطريقة التي يجب استدعاؤها اعتماداً على عدد ونوع الحجج

التحميل الزائد للمنشئ

كريت بيرسون

شخص

الاسم: سلسلة

CIN: تاريخ الميلاد الصحيح:

عدد صحيح

(عمر)

كريت بيرسون (سلسلة، إنت، إنت)

كريت بيرسون (سلسلة، عدد صحيح)

كريت بيرسون (عدد صحيح، سلسلة، عدد صحيح)

- 01تعريف الميراث

التحميل الزائد للمنشئ والتسلسل

تسلسل الشركات المصنعة

- تسلسل المنشئ هو أسلوب لاستدعاء مُنشئ الفئة الأصلية من مُنشئ الفئة التابعة
- يؤدي تسلسل المنشئ إلى تجنب تكرار رمز التهيئة للسمات التي ورثتها الفئة الفرعية



• استدعاء منشئ الميراث هو أسلوب لاستدعاء مُنشئ الفئة الأصلية من مُنشئ الفئة التابعة (الاسم: سلسلة، عدد صحيح ،تاريخ الميلاد: عدد صحيح، note1: صحيح، note2: حقيقى)

• يؤدي تسلسل المنشئ إلى تجنب تكرار رمز التهيئة للسمات التي ورثتها الفئة الفرعية (الاسم، CreatePerson note1=15 note2=13.5 ،CIN، CreatePerson note1=15 note2=13.5)

استدعاء منشئ الشخص

بناء الجسم الطالب



الفصل 1

تعريف الميراث

1. مبدأ الميراث. 2. أنواع الميراث. 3. التحميل الزائد للمنشئ والتسلسل

4. رؤية سمات وأساليب فئة الطفل

- 01 تعريف الميراث

رؤية سمات وأساليب فئة الطفل

رؤية الأعضاء وتراثهم

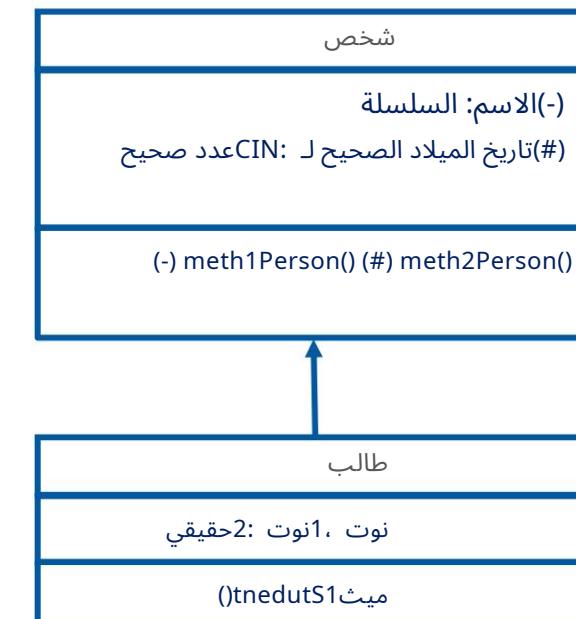
- إذا قامت فئة أصل بتعريف بعض (أو كل) سماتها وأساليبها على أنها خاصة، فلن يتمكن أي من فئاتها الفرعية من الوصول إليها أو تعديلها.
- للسماح لفئة فرعية بقراءة أو تعديل سمة موروثة، يجب أن تعلن الفئة الأصلية أنها محمية.

مثال :

()tnedutS1 ميث

الاسم="X" غير صحيح لأن الاسم هو سمة خاصة في الشخص
 صحيح لأنه تم إعلان CIN محمياً في الشخص (à) meth1Person() غير صحيح لأن الطريقة
 خاصة (à) صحيحة لأن الطريقة محمية meth2Person()

نص طريقة ()meth1Student()





تعريف تعدد الأشكال



ما ستعلم في هذا الفصل:

- فهم مبدأ تعدد الأشكال.
- معرفة كيفية إعادة تعريف الأساليب والتحميل

الزائد

ساعات 03



تعريف تعدد الأشكال



1. مبدأ تعدد الأشكال

2. إعادة تعريف الأساليب

3. طريقة التحميل الزائد

-02-تعريف تعدد الأشكال

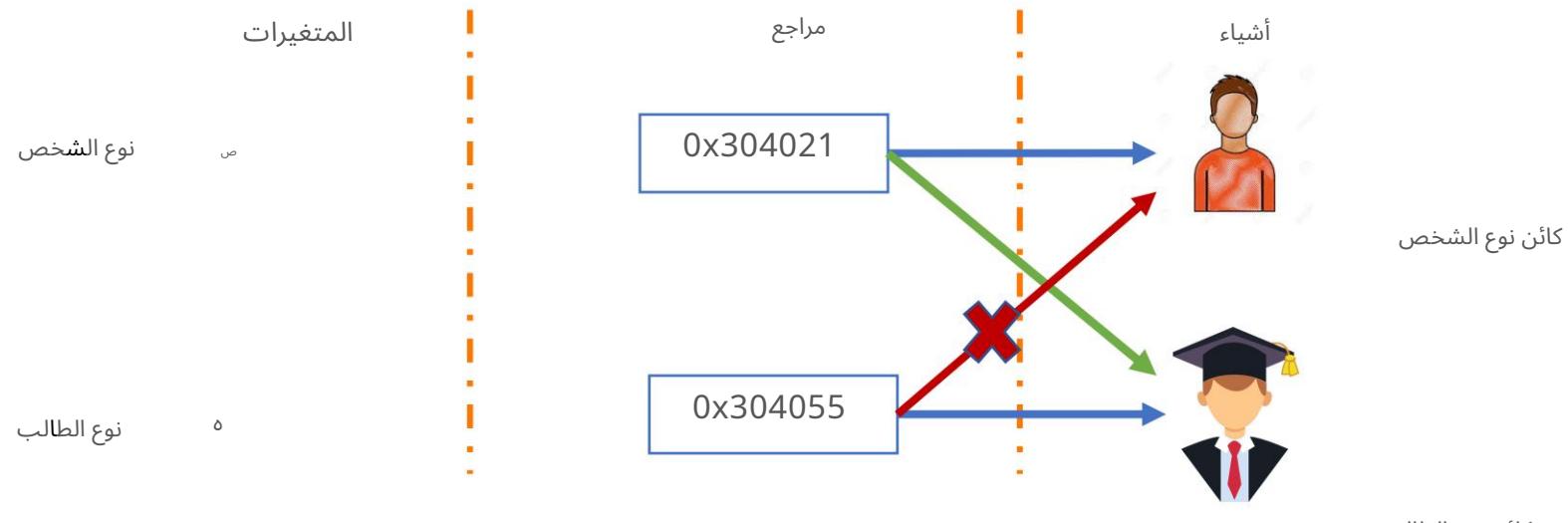
مبدأ تعدد الأشكال

مبدأ تعدد الأشكال

- يشير تعدد الأشكال إلى مفهوم في نظرية النوع، والذي بموجبه يمكن لاسم الكائن أن يحدد مثيلات الفئات مختلفة عن نفس الشجرة

مثال :

يمكن "رؤية" مثيل الطالب على أنه مثيل للشخص (وليس العكس!!)



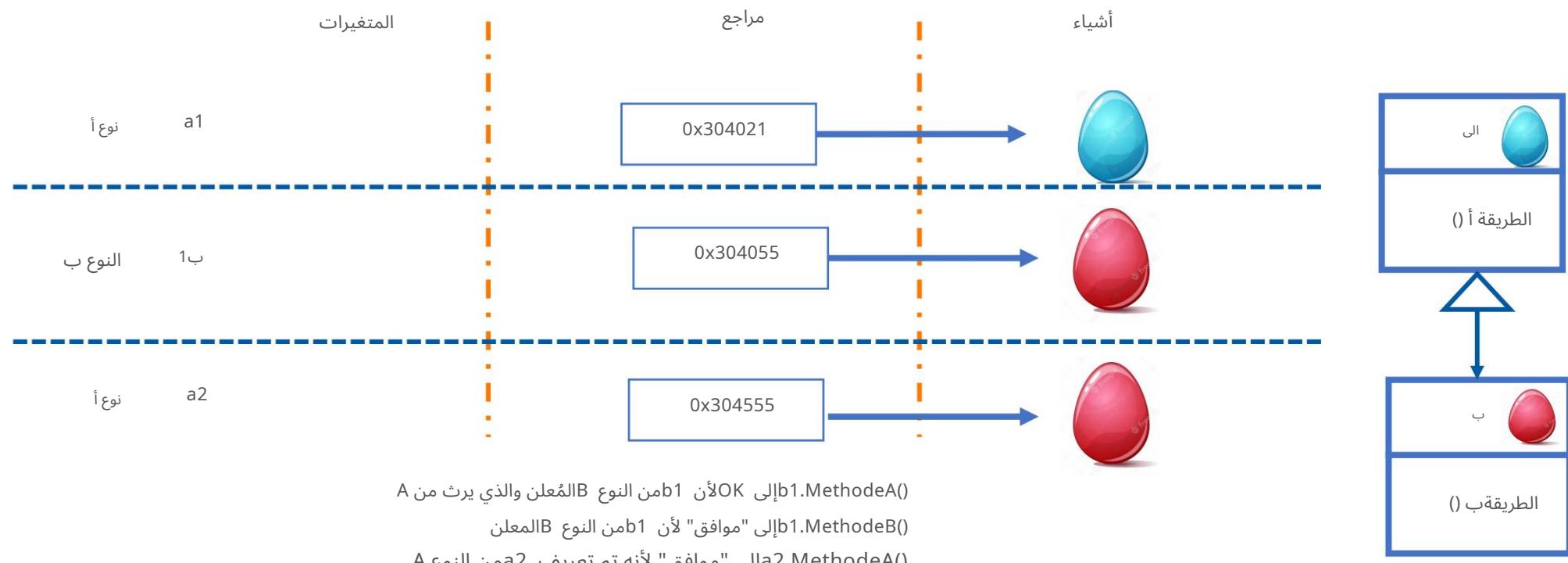
e = مسموح به لأن نوع p أكثر عمومية من نوع e

e = غير مصرح به لأن نوع e أكثر تحديداً من نوع p

-02-تعريف تعدد الأشكال

مبدأ تعدد الأشكال

نوع المتغير الذي يستخدمه المترجم لتحديد ما إذا كان قد تم الوصول إلى عضو صالح (السمة أو الأسلوب).





الفصل 2

تعريف تعدد الأشكال

1. مبدأ تعدد الأشكال

2. إعادة تعريف الأساليب

3. طريقة التحميل الزائد

- 02-تعريف تعدد الأشكال

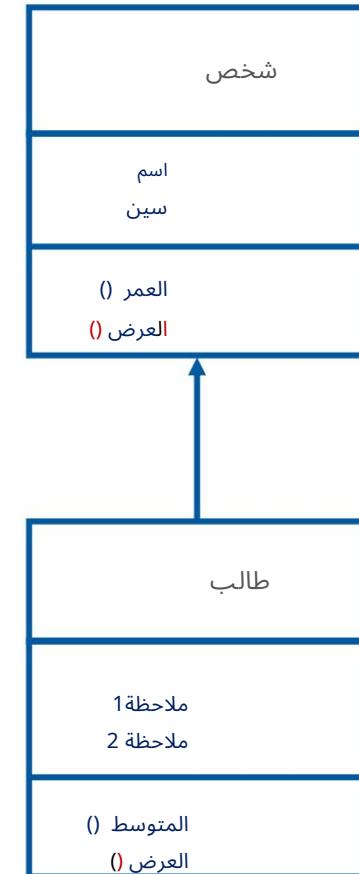
إعادة تعريف الأساليب



إعادة تعريف الأساليب القديمة

- الدافع وراء إعادة تعريف الطريقة الموروثة هو أن نسختها، في الفئة الأصلية، لا تتوافق مع احتياجات الفئة الفرعية.
- إعادة التعريف تجعل من الممكن اقتراح تعليمات برمجية مختلفة لطريقة موروثة مع الاحتفاظ برأسها. وإنما لا يمكن أن يكون إعادة تعريف بل أسلوب جديد مختلف تماماً عن الأسلوب الموروث.
- من فئة فرعية، من الممكن، في أي وقت، استدعاء طريقة تم إعادة تعريفها، في نسختها الأولية، المعلن عنها في الفئة الأصلية.

مثال: دع فئة الطالب ترث من فئة الشخص



• تعرض طريقة العرض (لفئة الشخص الاسم وسمات CIN للشخص. ترث فئة الطالب طريقة العرض (لفئة الشخص و

يعيد تعريفه

يقترح رمزاً جديداً (يضيف فصل الطالب العرض

• سمات الطالب Grade1 و Grade2

- 02-تعريف تعدد الأشكال

إعادة تعريف الأساليب

آلية تأخر الاتصال • بالنظر إلى 3 فئات A وB وC، يرث B من C وA يرث من B.

• ليكن `m` مثيلاً (كائناً) للفئة A. نرغب في استدعاء الأسلوب `m()` من كما يلي:

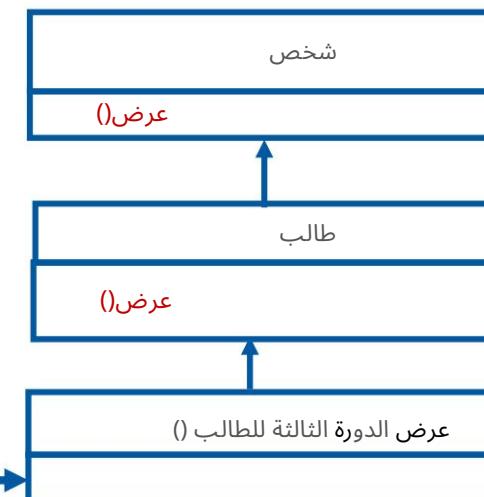
• يمكن استدعاء `m()` الذي يمثل الطريقة التي تنتهي إلى الفئة A. ويمكن استدعاء الطريقة من خلال الفئات التابعة لـ A، وهي C وB.

• افترض أن الفئة C تحدد أو تعيّد تعريف طريقة `m()`. إذا كان كائناً من الفئة C يستدعي الأسلوب `m()` فهو كائن من الفئة C. سيتم تنفيذه، وفي حالة الفشل يستمر البحث على مستوى الفئة B ثم A وهكذا.

مثال: أجعل `etc` كائناً من النوع `StudentThirdCycle`

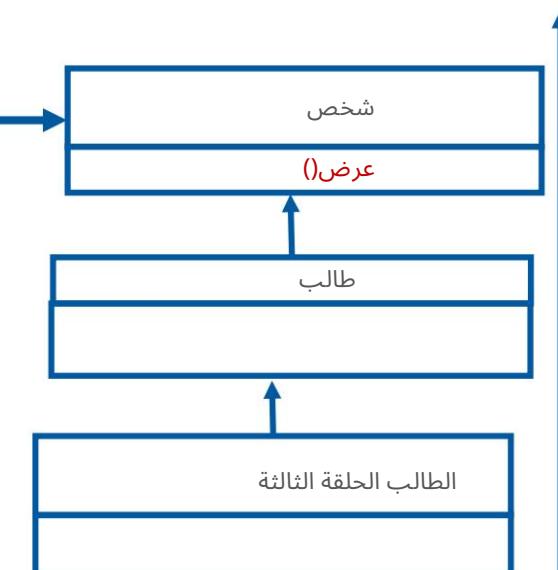
الحالة الأولى

سيتم تنفيذ طريقة العرض (لفئة `StudentThirdCycle`)



الحالة الثانية

يتم تنفيذ طريقة العرض (لفئة الشخص)
إلح.عرض ()





الفصل 2

تعريف تعدد الأشكال

1. مبدأ تعدد الأشكال

2. إعادة تعريف الأساليب

3. طريقة التحميل الزائد

-02-تعريف تعدد الأشكال

طريقة التحميل الزائد

• يسمح التحميل الزائد للطريقة بتعريف نفس الطريقة عدة مرات باستخدام وسائط مختلفة.

• يختار المترجم الطريقة التي يجب استدعاؤها بناءً على عدد الوسائط ونوعها.

• يتم تحميل الأسلوب بشكل زائد عندما يقوم بتنفيذ إجراءات مختلفة اعتماداً على نوع المعلمات التي تم تمريرها وعدها.





الفصل 3

تميز التجريد

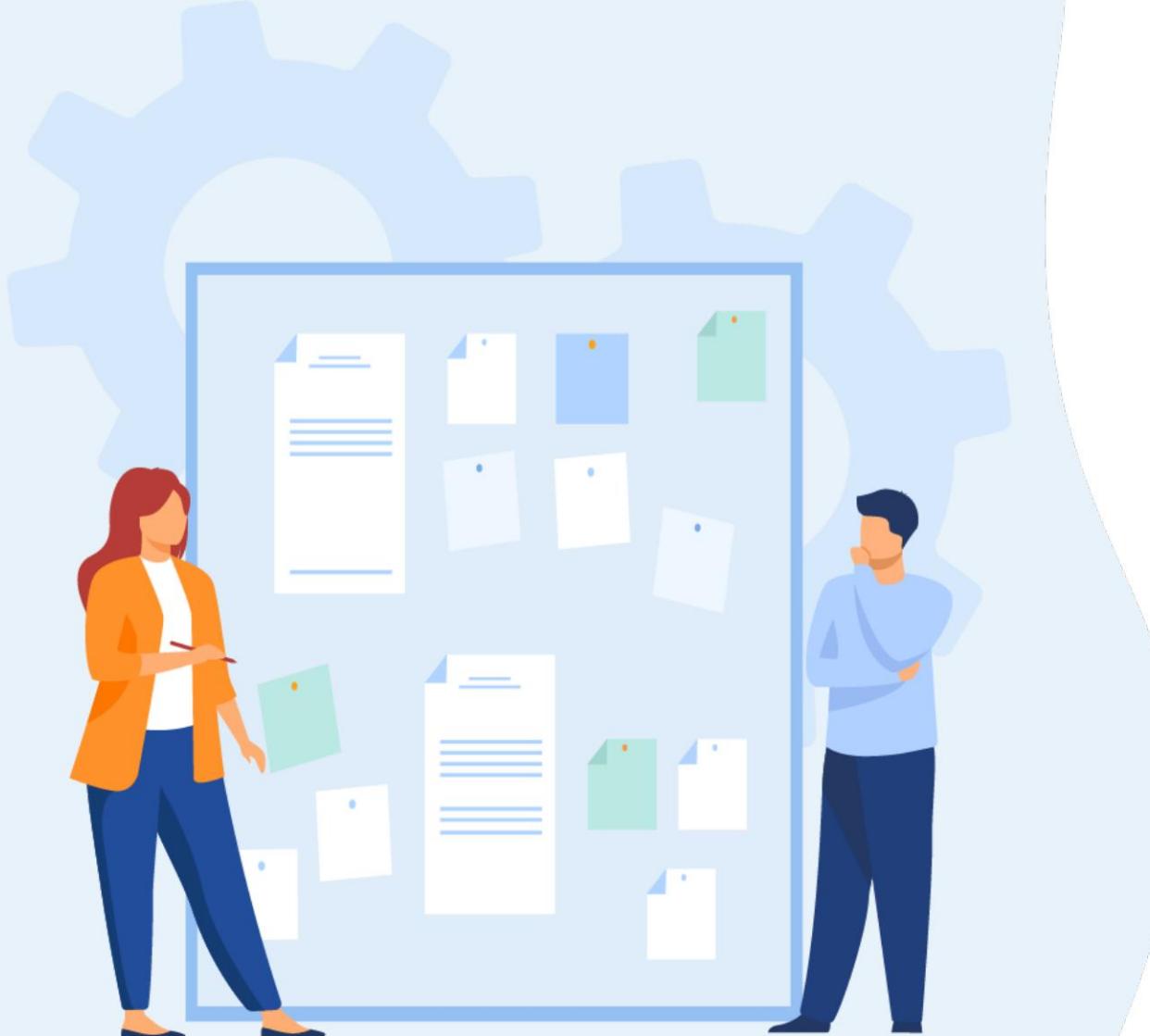


ما سنتعلمه في هذا الفصل:

فهم مبدأ التجريد في OOP • إتقان مفاهيم الطبقات المجردة والأساليب

خلاصة

03 ساعات



الفصل 3

تميز التجريد

1. المبادئ

2. الطبقات المجردة

3. الأسلوب المجردة

03 - تميز التجرييد

مبادئ



مبدأ التجرييد

- التجرييد هو مبدأ يتمثل في تجاهل بعض الجوانب التي لا أهمية لها المشكلة من أجل التركيز على أولئك الذين هم
- يسمح لك OOP بالتركيز على الخصائص المهمة للكائن. هدفها الرئيسي هو إدارة التعقيد عن طريق إخفاء التفاصيل غير الضرورية عن المستخدم
- يتتيح ذلك للمستخدم تنفيذ منطق أكثر تعقيداً دون فهم كل التعقييدات الخفية أو حتى التفكير فيها



الفصل 3

تميز التجريد

1. المبادئ

2. الطبقات المجردة

3. الأسلوب المجردة

-03 تميز التجريد

دروس مجردة



• تطبيق تلك المفاهيم في حل المسائل لها إفادة لتنفيذها

• فاؤبافات الفيصل للجوس هي المهمة الأولى في تحقيق كل المقدمة العثور على جميع الطرق المحددة في كل فئة مجردة في كل فئة فرعية محددة

• وبهذا المعنى، يمكن استخدام فئة مجردة:

• كجذر للميراث المتتالي:

• في هذه الحالة، ستحدد كل فئة فرعية خصائصها وطرقها.

• يعزز تعدد الأشكال

• تشكل الفئات المجردة وبالتالي نوعاً من العقد (مواصفات ملزمة) يضمن توفر طرق معينة في الفئات الفرعية والتي تلزم المبرمجين بتنفيذها في جميع الفئات الفرعية الملموسة

03 - تميز التجريد

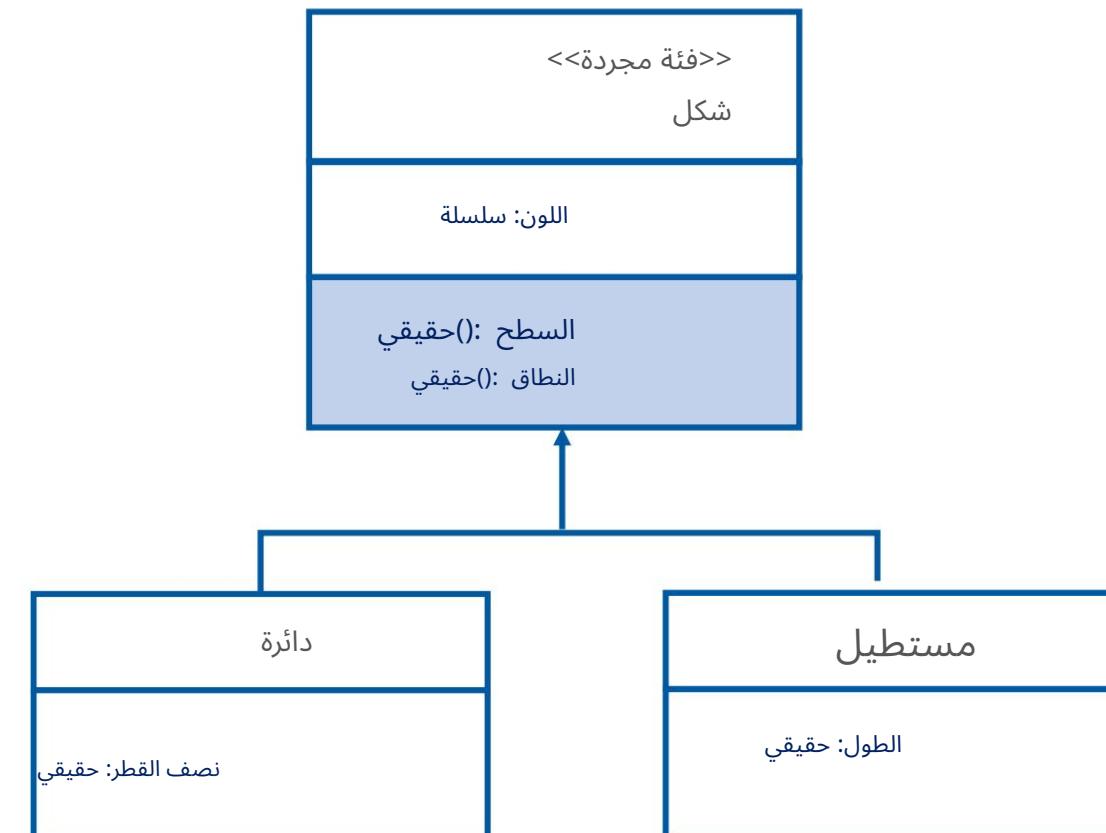
دروس مجردة

مثال :

مع العلم أن جميع الأشكال لها خصائص المحيط والمساحة، فمن الحكمة وضع أساليب محيط () ومنطقة () في الفئة التي هي في جذر الشجرة (الشكل)

لا تقدم طريقتنا (perimeter) و (surface) المحددتان في فئة الشكل أي تعليمات برمجية ويجب تنفيذها في الفئات الفرعية. تعتبر طريقتنا (perimeter) و (surface) طرفيتين

خلاصة





الفصل 3

تميز التجريد

1. المبادئ

2. الطبقات المجردة

3. الأساليب المجردة

03- تميز التجريد

طرق مجردة

- الطريقة المجردة هي الطريقة التي لا تحتوي على جسم. إنه ببساطة يحتوي على توقيع تعريف (بدون كتلة تعليمات)

تنطبق القواعد التالية على الفئات المجردة:

- لا يمكن إنشاء مثيل لفئة فرعية من فئة مجردة إلا إذا أعادت تعريف كل طريقة مجردة لفئتها الأصلية وتتوفر تطبيقاً (نصًا) لكل طريقة من الطرق المجردة

- تم الإعلان عن طريقة مجردة في الفصل الدراسي خلاصة

- إذا لم تقم فئة فرعية من فئة مجردة بتنفيذ جميع الأساليب المجردة التي ورثتها، فإن هذه الفئة الفرعية هي في حد ذاتها مجردة (وبالتالي لا يمكن إنشاء مثيل لها)

- إن الفصل الذي يحتوي على واحد أو أكثر من الأساليب المجردة يصبح بالضرورة فصلاً مجرداً
- ليس من الضروري أن يكون لديك أساليب مجردة في فئة مجردة.



التعامل مع الواجهات



ما ستتعلم في هذا الفصل:

فهم مبدأ الواجهات واستخداماتها • إتقان تنفيذ الواجهة

03 ساعات



التعامل مع الواجهات



- 1.تعريف الواجهات
- 2.فائدة الواجهات
- 3.تنفيذ الواجهات

04- التعامل مع الواجهات

تعريف الواجهات



- مثل الفصل والفصل المجرد، تتيح لك الواجهة تحديد نوع جديد (مرجع).
- الواجهة هي شكل خاص من الفئات حيث تكون كافة الأساليب مجردة.

<>الواجهة<>

للطباعة

طباعة()



التعامل مع الواجهات



- 1.تعريف الواجهات
- 2.فائدة الواجهات
- 3.تنفيذ الواجهات

04- التعامل مع الواجهات

فائدة الواجهات



فائدة الواجهات

تتيح لك الواجهات ما يلي:

- تحديد الخصائص التي يمكن استخدامها بواسطة الفئات التي تقوم بتنفيذ هذه الواجهات.
- إجبار الفئات التي تنفذها على تحديد

الطرق المجردة المعلنة في الواجهات.

- الاستفادة من تعدد الأشكال مع الحالات التي لا تشكل فئاتها جزءاً من نفس التسلسل الهرمي للميراث.



الفصل 4

التعامل مع الواجهات

- 1.تعريف الواجهات
- 2.فائدة الواجهات
- 3.تنفيذ الواجهات

04- التعامل مع الواجهات

تنفيذ واجهات



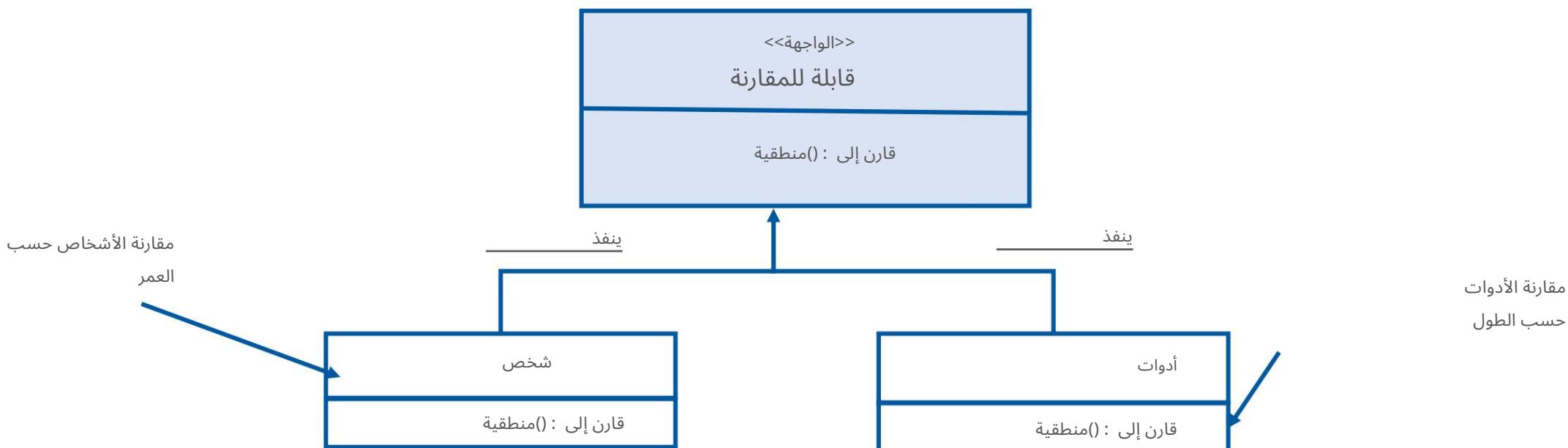
نقول أن الفصل ينفذ واجهة، إذا كان يوفر تطبيقاً (أي نصاً) لكل طريقة من الطرق المستخرجة من هذه الواجهة.

إذا قام الفصل بتنفيذ أكثر من واجهة واحدة، فيجب عليه تنفيذ جميع الأساليب المجردة لكل واجهة من الواجهات.

مثال:

إذا أردنا وصف وظيفة المقارنة المشتركة بين جميع الكائنات التي لها علاقة ترتيب (أصغر، متساوي، أكبر)، يمكننا تحديد الواجهة القابلة للمقارنة.

يجب أن تقدم فئة الشخص والأدوات التي تنفذ الواجهة القابلة للمقارنة تطبيقاً للأسلوب `CompareTo()` وإلا ستكون مجرد.



04- التعامل مع الواجهات

تنفيذ واجهات



مثال: 2

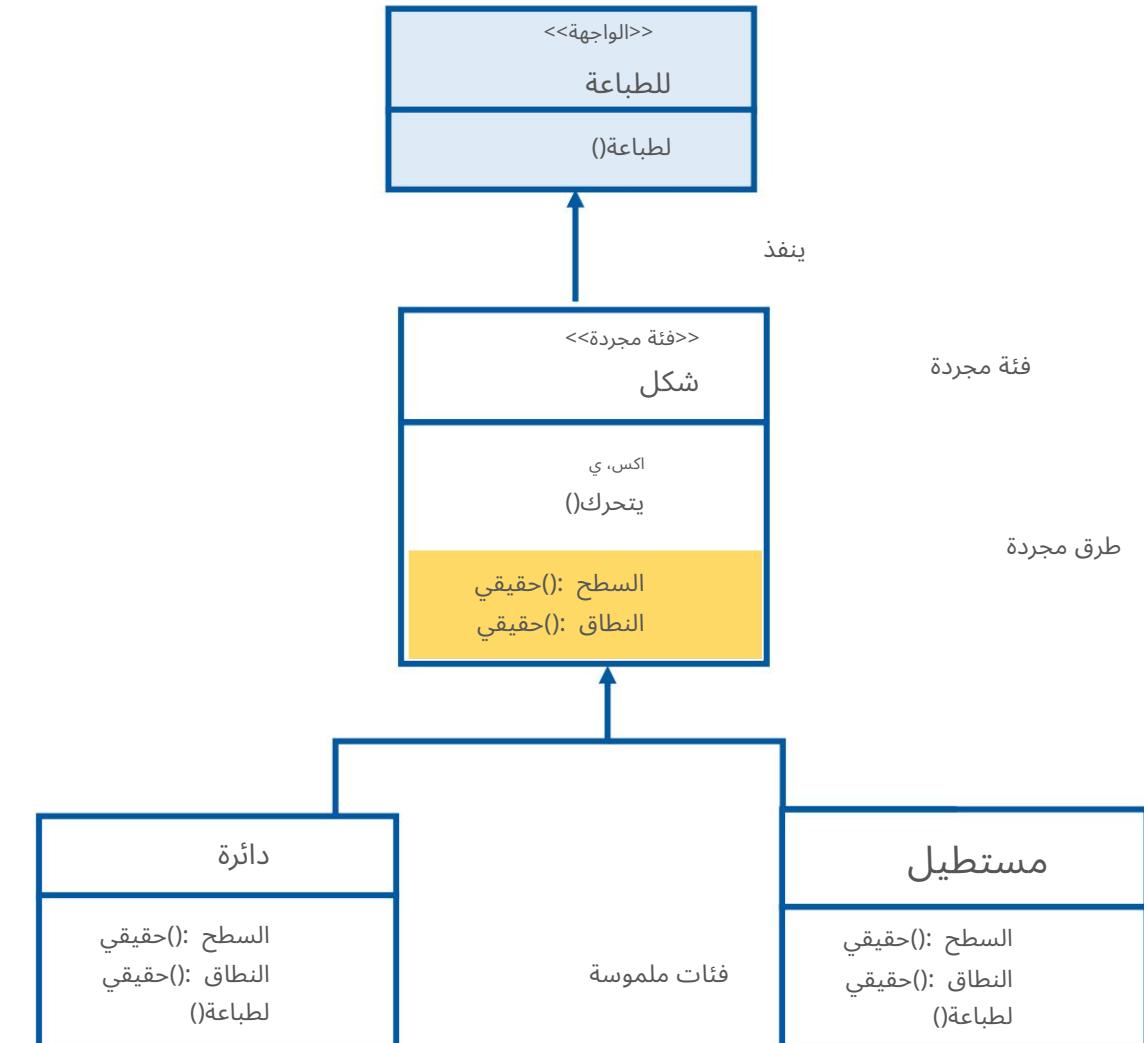
- لنفترض أننا نريد أن يكون للفئات المشتقة من فئة الشكل طريقة print() لطباعة الأشكال الهندسية.

الحل: 1

قم بإضافة طريقة مجردة Print() إلى فئة الشكل وبالتالي يجب على كل فئة فرعية محددة تنفيذ هذه الطريقة. إذا أرادت الفئات الأخرى (التي لا ترث من النموذج) أيضاً أن يكون لديها وظائف طباعة، فسيتعين عليها الإعلان عن طرق الطباعة المجردة مرة أخرى في هيكلها الشجري.

الحل: 2

تطبق فئة النموذج الواجهة القابلة للطباعة • باستخدام طريقة Print()





الجزء 3

تمرين الحلول الموجهة للكائنات



في هذه الوحدة، سوف تقوم بما يلي:

- استكشاف المفاهيم الأساسية الموجهة للكائنات في بايثون
- التعامل مع البيانات بشكل صحيح في بايثون (المجموعات والقوائم والملفات)
- التعامل مع التعبيرات العادية بشكل صحيح في بايثون
- التعامل مع الاستثناءات في بيthon





كود حل موجه للકائنات



ما ستعلمھ في هذا الفصل:

- إتقان ترمیز فئة في بايثون
- إتقان ترمیز المستويات الرئيسية لـ OOP في بايثون هي التغليف والميراث وتعدد الأشكال والتجريد



كود حل موجه للકائنات



1. إنشاء حزمة .2. ترميز الفصل .3. دمج

OOP مفاهيم

- 01 كود الحل الموجه للકائنات

إنشاء حزمة



- دليل يحتوي على ملفات و/أو أدلة = package
- لإنشاء الحزمة الخاصة بك، ابدأ بإنشاء مجلد باسم الحزمة الخاصة بك في نفس المجلد الذي يحتوي على برنامجك الرئيسي (مثلاً، main.py).
- تحتوي حزمة src على ملفات مصدر مصنفة في حزم فرعية
- قبل إصدار Python 3.3، كانت هناك حزمة تحتوي على وحدات __init__.py يجب أن يحتوي على ملف

src	2021-10-15 22:15	Dossier de fichiers
main	2021-10-15 22:08	Python File

حزمة سرك

__init__	2021-10-15 22:08	Python File
ExempleClass	2021-10-15 22:08	Python File

01- كود الحل الموجه للકائنات

إنشاء حزمة



لاستيراد واستخدام الفئات class1 و class2 المحددة في exampleClass.py، يجب عليك إضافة السطر التالي إلى main.py:

```
from class1 import ExampleClass  
from class2 import ExampleClass
```

إذا كان exampleClass.py موجوداً في الحزمة الفرعية المحددة في src، فيجب عليك إضافة السطر التالي إلى main.py:

```
from src.subpackage import ExampleClass
```



كود حل موجه للકائنات



1. إنشاء حزمة .2. ترميز **الفصل** .3. دمج

OOP مفاهيم

- 01 كود الحل الموجه للكائنات

عرض الطبقة

عریف الفئة في بایثون

الطبقات الفائقة التي يرث منها الفصل

سم الفئة (المعرف)

اسم الفئة: (superclass1, superclass2)

ف_tini_(الذات، الحج):

تهيئة السمات

متغير الذاتي = 0

نشئ الفئة (اختياري)

`self.variable` سمة (متغير) للفئة

طريقة_اسم

حسب التقليد فإن اسم الفصل يبدأ بحرف كبير

نصف الكلمة الأساسية pass حقيقة أن الفصل الدراسي فارغ.

فارغة ssalc:

۱۰

01 - كود الحل الموجه للકائنات

ترميز الفصل الدراسي



طرق

فصل.

• تمثل الطرق الوظائف المخصصة مباشرة للفئة. يصلون تلقائياً إلى السمات نفسها

• الأساليب هي وظائف تحتوي على المعلمة الصريحة (الذاتية) التي تمثل مثيل الفئة الحالية. هذه المعلمة يسمح باستخدام البيانات من هذه الفئة.

إعلان طريقة

```
def method_name(self, param_1, ..., param_n):
    فئة: class_name
        #نص الطريقة...
```

self

يعين Self المثيل الحالي للفئة

استدعاء طريقة

```
(value_1, ..., value_n) = امتغير من النوع class_name()
class_name t = cl.method_name
```

- 01 كود الحل الموجه للકائنات

ترميز الفصل الدراسي



صفات

- السمات هي متغيرات يتم تعبيئها مباشرة إلى فئة ما.
- سمات الفئة هي متغيرات عامة يمكن استخدامها في كافة أساليب تلك الفئة.

إعلان السمة

```
فئة 1  
param_n): self.attribute_name = param_1  
class_name: defmethod_name (self, param_1, ....
```

إعلان الصفة بأن يسبق اسمها الذات

- 01 كود الحل الموجه للકائنات

ترميز الفصل الدراسي



الأساليب والأهداف

مثال :

```

فئة: class_example:
"
"نحن نسعى لتوليد رقم عشوائي بين 0 و n1
طريقة الدفع 1 (الذات، ن):
    self.rnd=42 #rnd هي صفة يجب أن تسيقها الذات
    self.rnd=397204094*self.rnd%2147483647
    self.rnd % n #إرجاع n

إنشاء مثيل للفئة nb=class_example() #
استدعاء الطريقة x=nb.methode1(100)
طباعة(x) # ملصق 19

```

01 - كود الحل الموجه للકائنات

ترميز الفصل الدراسي



المنشء والمثيل

• منشئ الفصل هو أسلوب يجب أن يبدأ اسمه بـ `__init__`

• يجب أن يكون لدى المنشئ السمة الذاتية كمعلمة أول. يمكن أن يحتوي أيضًا على مجموعة من المعلمات. الـ `__init__` لا ينبغي للمنشئ أبدًا إرجاع النتيجة.

إعلان الشركة المصنعة

```
def __init__(self, param_1, ..., param_n):
    # كود البناء
```

مقالمة من منشئ

```
x = class_name (value_1,...,value_n)
```

مثال :

```
فئة الدرجة: 1
تعريف __init__(الذات):
    لا توجد معلمات إضافية tnirp("منشئ الفئة"
    # = 1 # إضافة السمة
    n إضافة السمة
        self.n
    )
```

يعرض مُنشئ الفئة `class1 print(xn) #`
الملاصق `1`

إعلان منشئ المعلمة `n`

فئة الدرجة: 2

```
def __init__(self, a, b): #
    # إضافة السمة
    n = (a + b) / 2 # إضافة السمة
        self.n
```

يعرض مُنشئ الفئة `class2 print(xn) #`
يعرض 7 `= class2(5, 9) #`

01 كود الحل الموجه للકائنات - ترميز الفصل الدراسي



المنشئ والمثيل

- بشكل افتراضي، أي فئة في بایثون لديها منشئ افتراضي بدون معلومات
 - لن يعد المنشئ الافتراضي موجوداً إذا كان للفئة مُنشئ معلمة
 - خلافاً لغيرها من اللغات، فئة بایثون لديها منشئ واحد فقط. ومع ذلك، تسمى قيمة افتراضية.
 - ملاحظة: يجب أن تسبق كافة الإعدادات المطلوبة أي إعدادات لها قيم افتراضية.

مثال:

شخص الطبقة: _____

_____ العددات العمر والجنس لها قيمة افتراضية. `def __init__(الذات, الاسم, العمر, الجنس = 1):`

_____ العمر self.age = _____ الاسم self.name =
_____ الجنس self.sex = sex

```
def DisplayInfo(self):  
    طباعة ("الاسم: " , self.name)  
    طباعة ("العمر: " , self.age)  
    طباعة ("الجنس: " , self.sex)
```

إذا: #_name_ == "main":
_____ #display Aimee 21 ("Aimee", "Aimee")
طباعة أنشي محبوب. #displayInfo() طباعة أنشي

#العمر والجنس بشكل افتراضي. أليس = شخص ("أليس") #يظهر أليس 1 ذكر أليس . عرض المعلومات ("طباعة")-----

```
tran = Person("Tran", 37) #poster Tran 37 Male tran.  
()#defaultgender.
```

- 01 كود الحل الموجه للકائنات

ترميز الفصل الدراسي

إنشاء مثيل للفصل الدراسي

عرض المثيل

شخص الطبقة:

```
#إعدادات العمر والجنس لها قيمة افتراضية.
def __init__(الذات, الاسم, العمر, 1=الجنس = "ذكر"):
    self.name = name
    self.age =
        العمر الذات. الجنس = الجنس
```

```
إذا كان __name__=='__main__':
    12,"eemiA")nosreP=Aimee
```

طباعة (أعجبني) كائن الشخص على 0x000002047E549070>#display

* تعلم الدالة () على التأكد من إنشاء مثيل من فئة معينة.

شخص الطبقة:

```
#إعدادات العمر والجنس لها قيمة افتراضية.
def __init__(الذات, الاسم, العمر, 1=الجنس = "ذكر"):
    الذات. اسم = اسم
    self.age =
        العمر self.sex=sex
```

```
إذا كان __name__=='__main__':
    12,"eemiA")nosreP=aimee
```

صحيح لأن الإعجاب من النوع Person print(isinstance(liked,Person)) #prints

01 - كود الحل الموجه للકائنات

ترميز الفصل الدراسي



مدمرة

• يتم استدعاء المدمرات عندما يتم تدمير كائن ما.

إن أدوات التدمير ليست ضرورية لأن Python لديها أداة تجميع البيانات المهملة التي تضمن إدارة الذاكرة تلقائياً.

(هي طريقة تسمى المدمر في بايثون. يتم استدعاؤه عندما تتم إزالة كافة المراجع إلى الكائن .`__del__` أي عند تنظيف الكائن.

بناء الجملة المدمر

```
مواطن __led__(الذات):
#إجراءات
```

- 01 كود الحل الموجه للકائنات

ترميز الفصل الدراسي



مثال :

```
شخص الطبقة:  
  
الذات، الاسم، العمر، "1"، "الجنس = ذكر":  
    الذات. اسم = اسم  
    self.age =  
        self.sex=sex  
  
تعريف معلومات العرض (الذات):  
    طباعة (self.name+ " " + self.age+ " " + self.sex)  
  
#تعريف المدمر:  
def __del__(self):  
    طباعة ("أنا المدمرة")
```

إذا كان :'_name_=='_main_':

شخص("ايمي", "12", "أنثى")

showInfo () #استدعاء وظيفة .showInfo

أحب. #نداء المدمرة delaimee

خطأ لأنه تم تدمير الكائن print(aimee) #show

#عرض:

#ايمي 21 أنثى

#أنا المدمر #Traceback (آخر مكالمة): File "C:/Users/DELL/Desktop/poo/example.py",
19, <module> # print(aimee) في

"aimee". #NameError: لم يتم تعريف الاسم.

01 - كود الحل الموجه للકائنات

ترميز الفصل الدراسي



مساهمة لغة بابيثون

• في التعليمات البرمجية المعدة لإعادة الاستخدام، من الضروري للغاية أن تحدد في الوثائق ما يفعله الفصل وما يفعله المدخل والمخرج

• تعرض السمة `_doc_` الخاصة وثائق الفصل

```

مركبة فئة():  

....  

فئة السيارة مع خيار اللون  

....  

تعريف __init__(الذات):  

    اللون الذاتي ="الأحمر"  

مواطنة get_color(الذات):  

    طباعة ("استعادة اللون")  

    إذا كان '__name__=='__main__':  

        my_car=()
print(my_car._doc_) #عرض: فئة السيارة مع خيار اللون

```

- 01 كود الحل الموجه للકائنات

ترميز الفصل الدراسي



مساهمة لغة بايثون

السمة __dict__ الخاصة

*هذه السمة الخاصة تعطي قيم سمات المثل:

```
إذا كان '__name__ == '__main__':
    my_car=()
    طباعة (my_car.__dict__) #display {'color': 'red'}
```

وظيفة دير

*تعطي الدالة dir نظرة عامة على أساليب الكائن:

```
إذا كان '__name__ == '__main__':
    my_car=()
    print(dir(my_car)) # يعرض أساليب فئة السيارة
```

- 01 كود الحل الموجه للકائنات

ترميز الفصل الدراسي



مساهمة لغة بايثون

عرض كائن

• يمنحك إمكانية إعادة تعريف عرض الكائن `__str__`:

مركبة فئة: ()

فئة السيارة مع خيار اللون

```
self.color = "red" def get_color (self): print
("استرداد اللون")def __init__(self):
```

`__str__`#تعريف الدالة

إرجاع ("لون السيارة هو: "+self.color)

إذا # 9`__name__ == '__main__': my_car= Vehicle() print(my_car) #call of __str__`

يعرض: لون المركبة أحمر

01 - كود الحل الموجه للકائنات

ترميز الفصل الدراسي



مساهمة لغة بايثون

إضافة سمة المثيل

- من الممكن إضافة سمة فقط لمثيل معين عبر بناء الجملة التالي:

```
example.new_attribute = value
```

```

إذا كان '__name__=='__main__':
    my_car=Car()
    طباعة ("سمة سيارتي")
    طباعة (my_car.__dict__)
    his_car=Car()
    طباعة ("سمة سيارته")
    طباعة (_tcid__.rac_ih)
    # ملخص:
    # سمة سيارتي
    # {"لون أحمر"}
    # سمة سياراته
    # {"اللون": "أحمر", "الرقم": 12345}

```

- 01 كود الحل الموجه للકائنات

ترميز الفصل الدراسي



فئة ايربوب

• تختلف سمات الفئة عن سمات المثيل.

• السمة التي تكون قيمتها هي نفسها لجميع مثيلات الفئة تسمى سمة الفئة. ولذلك فإن قيمة `_init_()` تممشاركة سمة الفئة بواسطة جميع الكائنات. • يتم تعريف سمات الفئة على مستوى الفئة وليس داخل أسلوب `. __init__()`.

• على عكس سمات المثيل، يتم الوصول إلى سمات الفئة باستخدام اسم الفئة أو اسم المثيل.

فئة الليمون:

```
Lemon.shape = params
" def GiveShape(self, params):
    Shape="
```

الشكل هو سمة ثابتة

- 01 كود الحل الموجه للકائنات

ترميز الفصل الدراسي



WEBFORCE
BE THE CHANGE

مثال :

```
فئة الفاكهة:  
فاكهة للأكل '#name' هو سمة ثابتة  
تعريف __tini_(الذات، اللون، الوزن_ز):  
طباعة ("أنا أحب الفاكهة.").  
self.color = color  
self.weight_g = weight_g  
  
إذا كان '__name__ == '__main__':  
تفاحة = فاكهة ("أخضر", 100)  
طباعة(فاكهة #ملصق فاكهة للأكل (eman.elppa))  
طباعة (اسم الفاكهة) #ملصق فاكهة للأكل
```

- 01 كود الحل الموجه للકائنات

ترميز الفصل الدراسي



سمة الطبقة

- تغيير سمة الفئة إلى اسم الفئة سوف يؤثر على الجميع حالات من فئة

- لن يؤثر تغيير سمة الفئة باستخدام المثيل على الفئة و حالات أخرى. سيؤثر هذا فقط على المثيل المعدل.

مثال :

فئة الفاكهة:

```
فاكهة لائل' #name=eman هو سمة ثابتة
apple =Fruit("green",100) #
الموز # =Fruit("yellow",100) #الملخص أحب الفاكهة.
print(banana.name) #فاكهة للأكل
فاكهة الصيف =mon.elppa
طباعة(eman.elppa) # ملخص فاكهة الصيف
print(Fruit.name) #فاكهة للأكل
print(banana.name) #فاكهة للأكل
```

إذا كان : __name__=='__main__

```
apple =Fruit("green",100) #
الموز # =Fruit("yellow",100) #الملخص أحب الفاكهة.
print(banana.name) #فاكهة للأكل
فاكهة الصيف =mon.elppa
طباعة(eman.elppa) # ملخص فاكهة الصيف
print(Fruit.name) #فاكهة للأكل
print(banana.name) #فاكهة للأكل
```

01 - كود الحل الموجه للકائنات

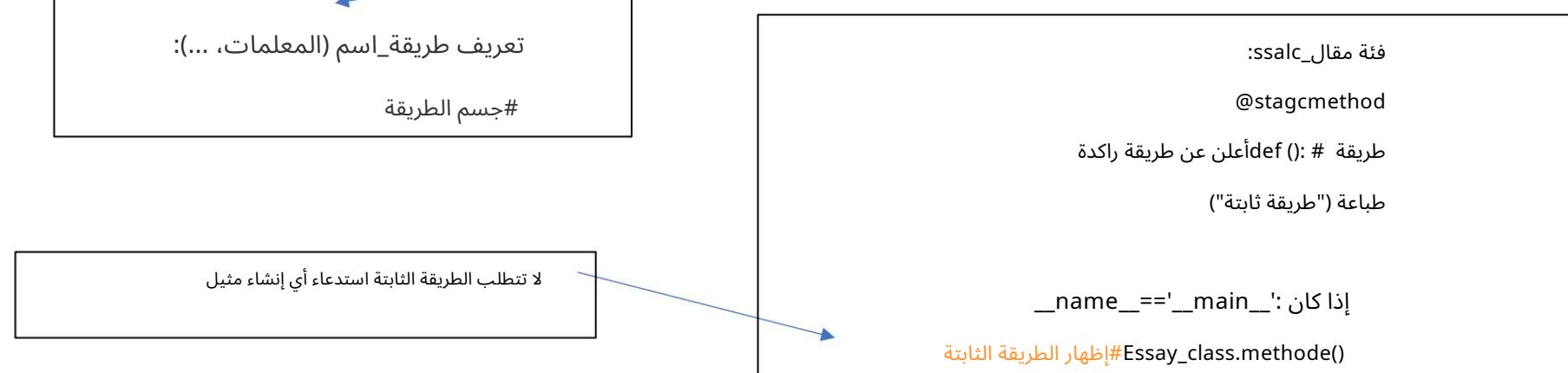
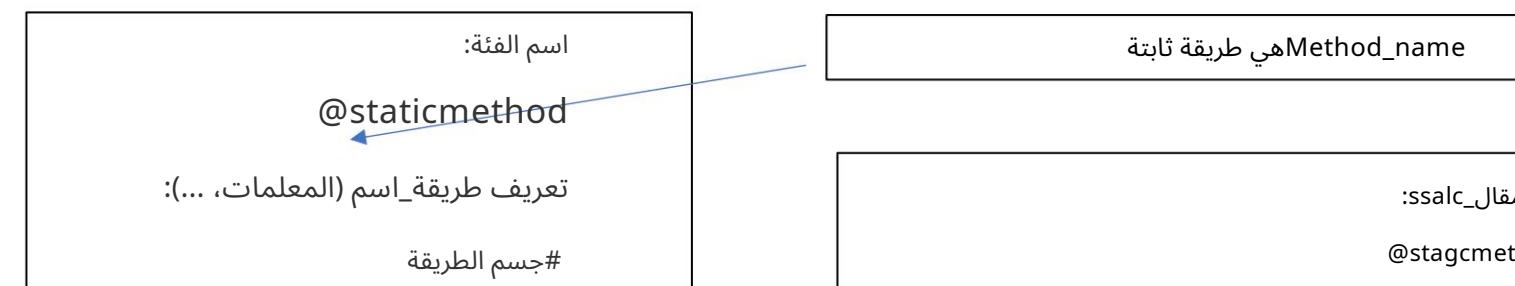
ترميز الفصل الدراسي



الأساليب الثابتة

- تذكير : يمكن استدعاء الأساليب الثابتة دون إنشاء مثيل أولاً.
- يمكن استدعاء أسلوب ثابت بدون مثيل. ولذلك، فإن المعلمة `Self` غير مجدية.
- يتم الإعلان عن الطريقة الثابتة باستخدام مصمم الديكور `@staticmethod`

مثال :



- 01 كود الحل الموجه للકائنات

ترميز الفصل الدراسي



إضافة طرق

لنفترض أنشأنا عن وظيفة ثابتة خارج الفصل الذي نعمل فيه. يمكن استدعاء هذه الوظيفة نفسها كطريقة ثابتة في فصلنا.

في المثال أدناه، نعلن عن دالة أسلوبية. باستخدام `staticmethod` نشير إلى فئة `Essay_class` التي هي كذلك طريقة ثابتة.

مثال :

```
طريقة التعريف: ()  
طباعة ("طريقة الأيل")  
  
فتة مقال_ssalc:  
يمر  
  
إذا كان: __name__ == '__main__'  
stagcmethod #Essay_class.methode = stagcmethod(methode)  
طريقة راكدة Essay_class.methode() #display:
```

- 01 كود الحل الموجه للકائنات

ترميز الفصل الدراسي



معدّل الوصول العام

- يمكن الوصول بسهولة إلى أعضاء الفصل المعلن للعامة من أي جزء من البرنامج.
- كافّة أعضاء البيانات ووظائف الأعضاء في الفصل تكون عامة بشكل افتراضي.

طالب الصف:

```
تعريف __init__(النفس، الاسم، العمر):
    self.Name = name
    self.Age = عمر
طباعة ("العمر: "، self.Age)
طباعة ("الاسم: "، self.Name)
```

الاسم والعمر هما
الصفات العامة

الطريقة العامة

إذا كان '__name__ == 'main':

```
obj = StudentGeek("R2J" ، 20)
طباعة ("الاسم: "، obj.Name)
طباعة ("العمر: "، obj.Age)
```

#ملقب:

```
R2J #الاسم:
20 #العمر:
```

- 01 كود الحل الموجه للકائنات

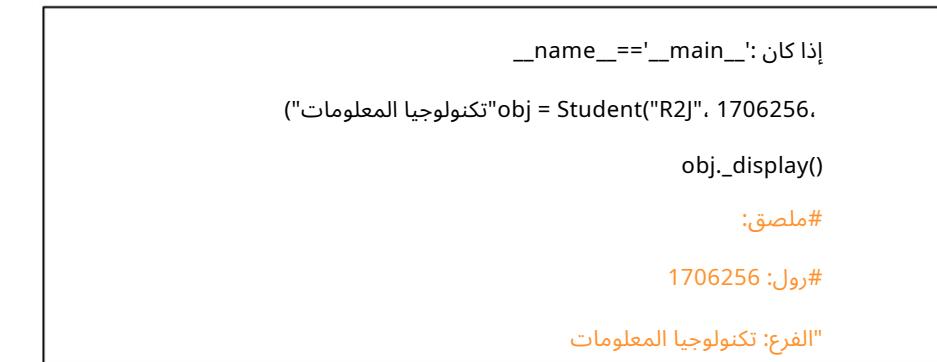
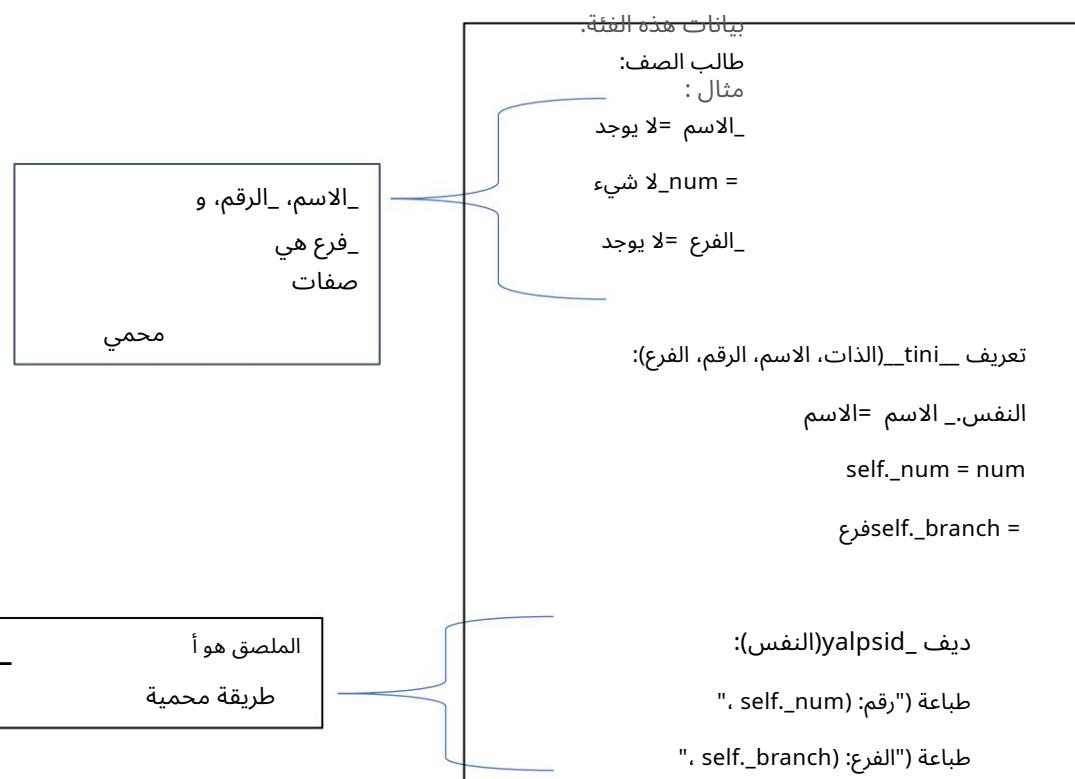
ترميز الفصل الدراسي



معدل الوصول المحمي

لا يمكن الوصول إلى أعضاء الفئة التي تم إعلان أنها محمية إلا للفئة المشتقة منهم.

• يتم الإعلان عن حماية بيانات أعضاء الفئة عن طريق إضافة رمز تسطير سفلي واحد `_` قبل العضو



إذا اعتبرنا أن فئة `Geek` مشتقة من الفئة `Student`:
الطالب إذن:

• `_name` و `_num` هم أعضاء بيانات محميون
طريقة `display()` هي طريقة محمية للفئة الفائقة
طالب.

طريقة `DisplayAge()` هي وظيفة عضو عامة في
فئة `Geek` مشتقة من فئة الطالب، الطريقة
من فئة `Geek` يحصل على بيانات الأعضاء
محمية من قبل فئة الطلاب.

01 كود الحل الموجه للકائنات - ترميز الفصل الدراسي



مُعَدّل الوصول الخاص

٤٠ لا يمكن الوصول إلى أعضاء الفصل الذي تم الإعلان عن أنه خاص إلا داخل الفصل، ومعدل الوصول الخاص هو معدل الوصول الأكثر أماناً.

عن هذه الفئة.

مثال:

• يتم إعلان أن أعضاء البيانات في الفصل خاصون عن طريق إضافة شرطة سفلية مزدوجة "—" قبل عضو البيانات

<code>_name_ == '__main__':</code>	إذا كان: __الاسم = لا شيء
<code>= StudentGeek("R2J", 1706256,</code>	<code>__عدد = لا شيء</code>
<code>obj.accessPrivateFunction()</code>	<code>__فرع = لا شيء</code>
ملخص:	تعريف __tini
(الذات، الاسم، الرقم، الفرع):	تعريف __name_
R2J	self.__name =
1706256	self.__num = num
الفرع: تكنولوجيا المعلومات	self. branch =

طباعة المعرف (النقط) هي دالة `display()` أعضاء خاصون، وهي `branch` عضو خاص به (الإمكان: `الوصول إليه` في الفصل) و `self._name` هي وظيفة عضو عامة في الفئة `AccessPrivateFunction()` طباعة ("الفرع:", `self._branch`)، `self._num`، طباعة ("العنوان:", `self._name`)، `self._branch` ويمكن الوصول إليه من أي مكان في البرنامج.

تعريف الوصول إلى الأعضاء الخاصين (Access Private Function): هي طريقة للوصول إلى الأعضاء الخاصين من فئة.



كود حل موجه للકائنات



1.إنشاء حزمة .2.ترميز الفصل .3.دمج مفاهيم

OOP

- 01 كود الحل الموجه للકائنات

تكامل مفاهيم OOP



في بايثون Setter و Getter

- غالباً ما يتم استخدام الحروف Getters و Setters في لغة Python لتجنب الوصول المباشر إلى حقل الفصل، على سبيل المثال. لا يمكن الوصول إلى المتغيرات الخاصة مباشرةً أو تعديلها بواسطة مستخدم خارجي.
- استخدام الدالة العادية للتعرف على سلوك الحروف والقيم.
- للحصول على خاصية Getters و Setters، إذا قمنا بتعريف أساليب `(get())` و `(set())`، فلن تعكس أي منها تنفيذ خاص.
- *مثال :

طالب الصنف:

```
=تعريف __init__(الذات، العمر: 0)
self._عمر =
def get_عمر(self): #declaration of the getter
    إرجاع الذات._عمر
#إعدان الوضع def set_عمر(self, x):
    self._عمر =
```

إذا كان '__name__ == '__main__':

```
= StudentGeek()
rag()
(12)ega_tes()
طباعة (rag())
ega_teg()
طباعة (rag_.ega_.rag())
#ملخص:
رقم 21
رقم 21
```

- 01 كود الحل الموجه للકائنات

تكامل مفاهيم OOP



مع الخاصية () وGetter وSetter

- في لغة Python، الخاصية () هي وظيفة مضمونة تقوم بإنشاء وإرجاع كائن خاصية.
- كائن الخاصية له ثلاثة طرق، getter()، setter()، delete()، property() في Python على ثلاثة وسیطات للخاصية • fget، fset، fdel، doc) هي دالة لاسترداد قيمة السمة
- هي دالة لتعيين قيمة السمة fset
- هي وظيفة لحذف قيمة سمة doc عبارة عن سلسلة تحتوي على وثائق السمة (ستتم رؤية سلسلة المستندات لاحقاً) fdel

- 01 كود الحل الموجه للકائنات

تكامل مفاهيم OOP



مثال

```
طالب الصف :
self._age = 0
def __init__(self):

    get_age(self): print("getter is call") return self._age
    def
        يتم استدعاء الواقع
        def set_age(self, a):
            self._age

    def del_age(self): del self._age

العمر = الخاصية(get_age, set_age, del_age)
```

يتصل

```
_name__=='__main__':
    علامه () = StudentGeek
    علامه العمر = 10
طباعة (علامه، العمر)
#ملخص:
#سيلر يسمى
#يسمى التجميد
#10
```

- 01 كود الحل الموجه للકائنات

تكامل مفاهيم OOP



استخدام @property

عبارة عن أداة تزيين تتجنب استخدام دالة `getter` الصريحة
هو مصمم ديكور يتتجنب استخدام وظيفة الضبط الصريحة

مثال :

```
طالب الصف :
تعريف __init__(الذات):
    الذات.العمر = 0
    @ملكية
        العمر الافتراضي (النفس)
    طباعة ("تسمى طريقة getter")
        إرجاع الذات.العمر
    @age.setter
        العمر الافتراضي (الذات، أ):
            طباعة ("تسمى طريقة الضبط")
            itself._age =
```

يتصل

```
إذا كان '__name__ == 'main':
    علامة ()
    نداء الواقع mark.age = 19 #
    طباعة (mark.age) # getter call
# ملخص:
# طريقة الضبط تسمى
# طريقة getter تسمى
# 19
```

- 01 كود الحل الموجه للકائنات

تكامل مفاهيم OOP



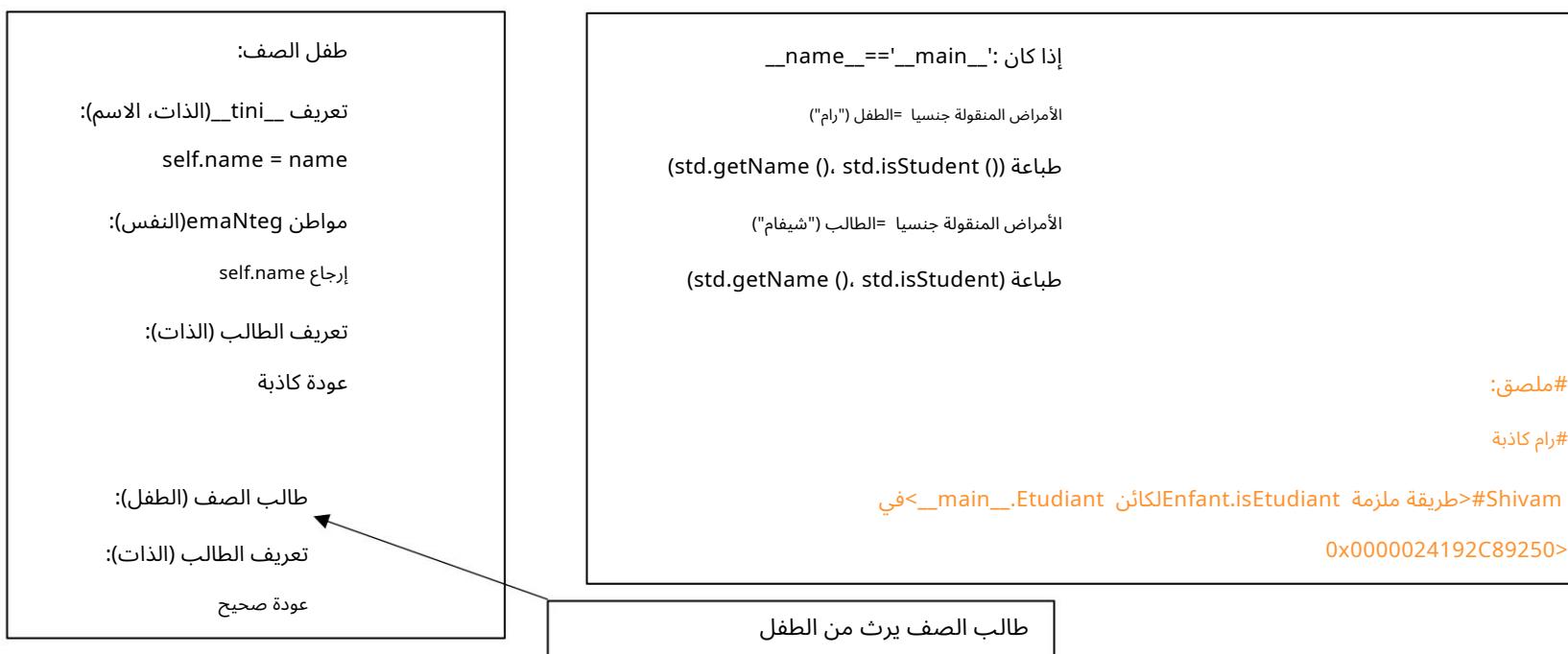
الميراث في بايثون

*تذكير: يتم تعريف الوراثة على أنها قدرة فئة ما على استخلاص أو وراثة خصائص فئة أخرى واستخدامها لكل منها كلما لزم الأمر.

بناء الجملة في بايثون

اسم فئة الفئة (SuperClass_Name)

مثال :



- 01 كود الحل الموجه للકائنات

تكامل مفاهيم OOP



التراث الفريد

• تذكير: الوراثة الفردية تسمح للفئة المشتقة بأن ترث خصائص فئة أصل واحدة، وبالتالي السماح للفئات المشتقة إعادة استخدام التعليمات البرمجية وإضافة وظائف جديدة إلى التعليمات البرمجية الموجودة.

مثال :

والد:

وظيفة ديف 1(الذات):

طباعة ("هذه الوظيفة موجودة في الفئة الأصل."")

فتة الطفل (والد): #يرث من فئة والد

وظيفة التعريف 2(الذات):

طباعة ("هذه الوظيفة موجودة في فئة فرعية.")

إذا كان :_name_=='_main_

الفئة = الطفل ()

كائن. ()

func2 ()

كائن. #

#هذه الوظيفة موجودة في الفئة الأصل.

#هذه الوظيفة في فئة الطفل.

- 01 كود الحل الموجه للકائنات

تكامل مفاهيم OOP



تعدد الميراث

• تذكير: عندما يمكن اشتراق فئة ما من عدة فئات أساسية، فإن هذا النوع من الميراث يسمى الميراث المتعدد. في الميراث المتعدد، يتم توريث جميع وظائف الفئات الأساسية في الفئة المشتقة.

مثال :

أم الطبقة:

```
'''  
    = اسم الأم
```

الأم (نفسها):

طباعة (self.mothername)

أب الصف :

```
'''  
    = اسم الأب
```

دافع عن الأب (نفسه):

طباعة (اسم الأب)

طفل الطبقة (الأم، الأب): # يرث من الصف الأم والأب

الدافع عن الوالدين (نفس):

print("Father:", self.parentname) #
اسم الأب هو سمة موروثة

print("Mother:", self.mothername) #
اسم الأم هو سمة موروثة

إذا كان :'_main_':

```
= s1=ال طفل()
```

اسم الأب = "ذاكرة الوصول العشوائي".

اسم الأم = "سيتا".

s1.parents()

ملخص:

#الأب : رام

#الام : سيتا

- 01 كود الحل الموجه للકائنات

تكامل مفاهيم OOP



الميراث المتتالي

• في الميراث المتتالي، يتم بعد ذلك توريث وظيفة الفئة الأساسية والفئة المشتقة إلى الفئة المشتقة الجديدة

جد الطبقة:

تعريف __tini_(النفس، اسم الجد):

الذات. اسم الجد = اسم الجد

أب الصف (الجد):

تعريف __tini_(النفس، اسم الأب، اسم الجد):

الذات. اسم الأب = اسم الأب

الجد.__tini_(النفس، اسم الجد)

فتة الطفل (الأب):

تعريف __tini_(النفس، اسم الطفل، اسم الأب، اسم الجد):

الذات. اسم الطفل = اسم الطفل

الأب.__tini_(النفس، اسم الأب، اسم الجد)

تعريف اسم_العرض (النفس):

طباعة ("اسم_الجد:", self. Grandfather_name)

طباعة ("اسم_الأب:", ذاتي. اسم_الأب)

طباعة ("اسم_الطفلي:", self. اسم_الطفلي)

إذا كان '__name__ == '__main__

= s1. الطفل ("الأمير", "راميال", "لال ماني")

طباعة .s1(اسم_الجد)

s1.display_name()

#ملخص:

#لال ماني

#اسم_الجد : لال ماني

#اسم_الأب: راميال

#اسم_الطفلي : الأمير

- 01 كود الحل الموجه للકائنات

تكامل مفاهيم OOP



سلسل الشركات المصنعة

• تسلسل المنشئ هو أسلوب لاستدعاء مُنشئ الفئة الأصلية من مُنشئ الفئة التابعة
مثال :

جد الطبقة:

تعريف `_tini`_(النفس، اسم الجد):

الذات. اسم الجد = اسم الجد

أب الصف (الجد):

تعريف `_tini`_(النفس، اسم الأب، اسم الجد):

الذات. اسم الأب = اسم الأب

الجد.`_tini`_(النفس، اسم الجد)

استدعاء مُنشئ الفئة الأصل لتهيئة السمة الموروثة `GrandfatherName`

- 01 كود الحل الموجه للકائنات

تكامل مفاهيم OOP



تسلسل الشركات المصنعة

فتة الطفل (الأب):

تعريف __tini_(النفس، اسم الطفل، اسم الأب، اسم الجد):

الذات. اسم الطفل = اسم الطفل

الأب. __tini_(النفس، اسم الأب، اسم الجد)

تعريف اسم العرض (النفس):

طباعة ("اسم_الجد:",)، self. Grandfather_name

طباعة ("اسم_الأب:", ذاتي، اسم_الأب)

طباعة ("اسم_الطفل:",، اسمself. الطفل)

استدعاء مُنشئ الفتة الأصل لتهيئة سمات اسم الأب واسم الجد الموروثة

nomPere nomGrandPere

إذا كان __name__ == '__main__':

= 1 كالطفل ("الأمير", "راميال", "لال ماني")

طباعة .(s1.اسم_الجد)

s1.display_name()

#ملصق:

#لال ماني

#اسم_الجد : لال ماني

#اسم_الأب: راميال

#اسم_الطفل : الأمير

- 01 كود الحل الموجه للકائنات

تكامل مفاهيم OOP



التحميل الزائد للمشغل في بايثون

- يسمح لك التحميل الزائد للمشغل بإعادة تعريف معنى المشغل بناءً على فئتك.
- هذه الميزة في بايثون، والتي تسمح لنفس العامل أن يكون له معنى مختلف حسب السياق.

العمليات الحسابية

عنصر الطبقة:

```
_init_(self, x, y): self.x = x self.y = y
    def
    = Point(5, 1) p3 = p1+p2 #error
    p1 = Point(2, 4) p2
    __name__=='__main__':
```

يعرض: p3 = p1+p2 TypeError: (آخر مكالمة): ملف "prog.py", السطر 10, في traceback <نوع (أنواع) معاملات غير مدعومة لـ "tnioP" و "Point" Module>

!!!

خطأ!!!

لم تكون بايثون تعرف كيفية إضافة كائنين نقطيين معاً ensemble.

- 01 كود الحل الموجه للકائنات

تكامل مفاهيم OOP



التحميل الزائد للمشغل في بايثون

لزيادة التحميل على عامل التشغيل ، +سيتعين علينا تنفيذ الدالة ()__add__ في الفصل

مثال :

عنصر الطبقة:

```
x, y): self.x = x self.y = y
def __init__(self,
```

{1}").format(self.x, self.y) إرجاع def __str__(self):

"{0},

return Point(a, b) التحميل الزائد للمشغل def __add__(self, p): #
+ a = self.x + px b = self.y + py

1) إذا (4النقطة)2. p2 = p1 (5النقطة)2. __name__ =='__main__':

(7.5) طباعة p3 = p1+p2
(p3) #display

- 01 كود الحل الموجه للકائنات

تكامل مفاهيم OOP



وظائف التحميل الزائد الخاصة بالمشغل في بايثون

	المشغل <small>أعلى الوعاء</small>	تفسير بايثون
الطرح	إضافة صن=ص2	صن1_إضافة_(ص2)
عملية الضرب	صن1*ص2	صن1_.مول_(ص2)
قوة	صن1**ص2	صن1_.الأسرى_(ص2)
قسم	صن1/ص2	صن1_.videurt_(ص2)
تقسيم صحيح	صن1//ص2	صن1_.vidroolf_(ص2)
الباقي (مودلو)	صن1%ص2	صن1_.dom_(ص2)
ثنائي و	صن1&ص2	صن1_.و_(ص2)
ثنائي أو	صن1 ص2	صن1_.أو_(ص2)
XOR	صن1^ص2	صن1_.rox_(ص2)
غير ثانوي	ـ ص1	صن1_.عكس_(ص2)

- 01 كود الحل الموجه للકائنات

تكامل مفاهيم OOP



التحميل الزائد للمشغل في بايثون

عوامل المقارنة

• في بايثون، من الممكن زيادة التحميل على عوامل المقارنة.

استيراد الرياضيات

عنصر الطبقة:

تعريف __init__(الذات، x، y):

self.x = x

self.y = y

مواطن __lt__(الذات، ص):

m_self = math.sqrt((self.x ** 2) + (self.y ** 2))

m2_p = math.sqrt((px ** 2) + (py ** 2))

إرجاع m_self < m2_p

إذا كان: __name__ == '__main__'

4) p1 = النقطة(2, 4)

1) p2 = النقطة(5, 1)

إذا كانت: p1 < p2:

طباعة p2(")" أبعد من p1)

ملخص:

أبعد من p1 # p2

انخفاض التحميل الزائد للمشغل

- 01 كود الحل الموجه للકائنات

تكامل مفاهيم OOP



وظائف التحميل الزائد الخاصة بالمشغل في بايثون

المشغل أو العامل	تعبير	تفسير بايثون
أقل من	ص1<ص2	ص1._lt_(ص2)
أقل أو متساوية	ص1<=ص2	ص1._le_(ص2)
متساوي	ص1==ص2	ص1._eq_(ص2)
مختلف	ص1!=ص2	ص1._ne_(ص2)
أكبر من	ص1>ص2	ص1._gt_(ص2)
أكبر من أو يساوي	ص1>=ص2	ص1._ge_(ص2)

- 01 كود الحل الموجه للકائنات

OOP تکامل مفاهیم



تعدد الأشكال والميراث

- في بايثون، يسمح لك تعدد الأشكال بتعريف الأساليب في الفئة الفرعية التي لها نفس اسم الأساليب الموجودة في الفئة الفرعية.
- في الميراث، يرث الفصل الفرعي أساليب الفصل الأصلي.
- من الممكن تعديل طريقة في فئة فرعية ورثتها من الفئة الأصلية (إعادة تعريف الطريقة).

مثال :

```

صنف الطيور:
مقدمة تعريفية (نفسية):
طباعة ("هناك أنواع مختلفة من الطيور")
رحلة الدفاع (النفس):
طباعة ("هناك طيور تطير والبعض الآخر لا")
فئة العصفور (الطاير):
رحلة الدفاع (النفس):
طباعة ("العصافور يمكنه الطيران.")
طائفة النعامة (الطيير):
رحلة الدفاع (النفس):
طباعة ("النعامة لا تطير.")

```

```

إذا كان: __name__=='__main__':
    obj_bird = obj_bird()
    obj_spr = obj_spr()
    obj_ost = obj_ost()

    obj_bird.intro() #shows: هناك أنواع مختلفة من الطيور
    obj_bird.vol() #display: هناك طيور تطير وأخرى لا تطير.
    obj_spr.intro() #shows: هناك أنواع مختلفة من الطيور
    obj_spr.vol() #shows: العصفور يمكنه الطيران.
    obj_ost.intro() #shows: هناك أنواع مختلفة من الطيور.
    obj_ost.vol() #shows: النعامة لا تطير.

```

- 01 كود الحل الموجه للકائنات

تكامل مفاهيم OOP



فئة مجردة

• تذكر: الفئات المجردة هي فئات لا يمكن إنشاء مثيل لها، فهي تحتوي على واحد أو أكثر من الأساليب المجردة
(طرق بدون كود)

• تتطلب الفئة المجردة فئات فرعية توفر تطبيقات للطرق المجردة، بخلاف هذه الفئات الفرعية
يتم الإعلان عنها مجردة

- فئة مجردة ترث من فئة مجردة قاعدة - ABC

• الغرض الرئيسي من الفئة الأساسية المجردة هو توفير طريقة موحدة لاختبار ما إذا كان الكائن يلتزم بالمواصفات

منح

• لتحديد طريقة مجردة في فئة مجردة، نستخدم الديكور @abstractmethod

- 01 كود الحل الموجه للકائنات

تكامل مفاهيم OOP



مثال

Abstractmethod عبارة عن وحدة بايثون مدمجة، تقوم باستيراد ABC و Abstractmethod # abc من

فئة الحيوان # (ABC) ترث من ABC (الفئة الأساسية المجردة)
 مصمم ديكور لتعريف الطريقة المجردة
 @abstractmethod #
 مواطنه G (النفس): taeMAeviG

يمر

صنف الحصان (الحيوان): # الصنف الذي يرث من الحيوان
 حدد اسم_آخر (النفس):
 طباعة ("مرحبا الحصان!")

إنشاء مثيل لـ Cheval مستحيل لأن طريقة DonneAManger مجردة ويجب تنفيذها في

Cheval() # if __name__ == '__main__': po = Cheval()

ملف

ملحق:
 التتبع آخر مكالمة أخيرة:
 "C:\Users\DELL\AppData\Local\Programs\Python
 <module>, 14, في \السطر ، Python39\ex0.py",
 #إنشاء مثيل لشيفال مستحيل لأن طريقة التغذية مجردة
 po = Cheval()
 ويجب تنفيذها في الباندا

لا يمكن إنشاء مثيل لفئة المجردة Cheval باستخدام الطريقة
 DonneAManger المجردة



الفصل 2

معالجة البيانات

ما سنتعلمه في هذا الفصل:

- التعامل مع هيكل بيانات بايثون، أي المجموعات والقوائم
- التعامل مع ملفات البيانات في بايثون





الفصل 2

معالجة البيانات

1. القائمة
2. المجموعات
3. الملفات

- 02 التعامل مع البيانات

القوائم



WEBFORCE
BE THE CHANGE

القوائم

• القائمة عبارة عن مجموعة مرتبة وقابلة للتعديل ويمكن أن تحتوي على نفس القيمة عدة مرات.

• يتم الإعلان عن القائمة على النحو التالي:

[el1, elt2,...eln] = القائمة

• القائمة[i]: تسمح بالوصول إلى عنصر القائمة الموجود في الموضع i

#بيانات إلسا thisList=[تفاحة، "موز" ، "كرز"]

#طباعة (قائمة thisList)

طباعة (هذه القائمة [i]) [القائمة]

طباعة (هذه القائمة [-1]) [الدورة]

thisList1=[تفاح، "موز" ، "كرز" ، "برتقال" ، "كيوي" ، "بطيخ" ، "مانجو"]

طباعة (قائمة thisList) [5 غلى 12] [5 غلى 5] غير متضمنة

عرض:
["تفاحة" ، "موز" ، "كرز"]
موز
الكرز
["كرز" ، 'برتقالي' ، 'كيوي']

• يسمح لك بتغيير قيمة عنصر القائمة الموجود في الموضع i: List[i]=val:

هذه القائمة =["تفاحة" ، "موز" ، "كرز"]

هذه القائمة [1] الكشمش الأسود

print(thisList) #print ['apple', 'blackcurrant', 'cherry']

عرض:

["تفاحة" ، "الكشمش الأسود" ، "الكرز"]

- 02 التعامل مع البيانات

القوائم



WEBFORCE
BE THE CHANGE

القوائم

*تصفح القائمة:

```
this_eList =
```

```
لـ x في ce_eList: #browse
```

```
طباعة(x) #قيمة x التي تتوافق مع عنصر في القائمة
```

*البحث عن عنصر في القائمة:

```
thisList=
```

```
إذا كانت قائمة "تفاحة": #تحقق مما إذا كانت السلسلة عنصر قائمة
```

```
طباعة ("نعم، "تفاحة" تنتهي إلى القائمة")
```

*الدالة len(List): هي دالة تستخد لـرجاع طول قائمة القائمة

```
هذه القائمة = ["تفاحة", "موز", "كرز"]
```

```
print(len(thisList)) #عرض طول القائمة
```

- 02 التعامل مع البيانات

القوائم



WEBFORCE
BE THE CHANGE

القوائم

• هي وظيفة تسمح لك بإضافة عنصر Foncyon append(elem):

البند في نهاية القائمة

هذه القائمة = ["تفاحة", "موز", "كرز"]

أضاف العنصر "البرتقالي" إلى نهاية القائمة thisList.append("orange") #

طباعة(هذه القائمة) موز، كرز، برتقالي]

• وظيفة إدراج (pos, elem) هي وظيفة تسمح لك بإضافة

عنصر العنصر في الموضع أمن القائمة

هذه القائمة = ["تفاحة", "موز", "كرز"]

أدخل العنصر "orange" في الموضع الثاني thisList.insert(1, "orange") #

#عرض عناصر القائمة [تفاحة، طباعة(هذه القائمة)]

• هي وظيفة تضمن حذف الأخير Foncyon pop()

قائمة الأغراض

هذه القائمة = ["تفاحة", "موز", "كرز"]

إزالة العنصر الأخير من القائمة وهو الكرز thisList.pop() #

عرض عناصر القائمة [print(thisList) #

- 02 التعامل مع البيانات

القوائم



WEBFORCE
BE THE CHANGE

القوائم

- وظيفة `del(elem)` هي وظيفة تسمح لك بحذف عنصر معين عنصر في القائمة

```
thisList = ["تفاحة", "موز", "كرز"]
```

العنصر الموجود في الموضع الأول `Del(thisList[0]) #delete`

طباعة (هذه القائمة) `#طباعة [الموز, "الكرز"]`

- وظيفة تمديد (القائمة): هي وظيفة تسمح لك بدمج القائمة

```
القائمة أ = ["أ", "ب", "ج"]
```

```
القائمة ب = [1, 8, 9]
```

قائمة أ. تمديد (قائمة ب)

#دمج القائمتين List1 و List2

طباعة (قائمة أ)

#عرض [أ, ب, ج, 1, 8, 9]

- وظيفة `:Copy()` هي وظيفة تسمح لك بنسخ محتويات القائمة إلى

```
هذه القائمة = ["تفاحة", "موز", "كرز"]
```

= thisList.copy()

طباعة (ماليست) `#print ["apple", "banana", "cherry"]`

أخرى

- 02 التعامل مع البيانات

القوائم



WEBFORCE
BE THE CHANGE

القوائم

• وظيفة Clear() هي وظيفة تسمح لك بحذف جميع عناصر ملف

قائمة

هذه القائمة = ["تفاحة", "موز", "كرز"]

thisList.clear()

طباعة (هذه القائمة) #عرض قائمة فارغة []

• وظيفة Reverse() هي وظيفة تسمح لك بعكس القائمة

هذه القائمة = ["تفاحة", "موز", "كرز"]

thisList.reverse() #reverse list thislist

#طباعة ([apple, banana, cherry]) عكوسه []



الفصل 2

معالجة البيانات

1. القائمة
2. المجموعات
3. الملفات

- 02 التعامل مع البيانات

المجموعات



*تحتوي وحدة المجموعات على حاويات

البيانات المتخصصة

*قائمة الحاويات هي كما يلي

حاوية

جدوى

nametuple

دالة لفئة فرعية تحتوي على حقول مسممة

في أقرب وقت

حاوية تشبه القائمة ولكن مع سرعة الإضافة والحذف في كل طرف

خريطة السلسلة

يتيح لك ربط عدة تعينيات معاً لإدارتها مثل أي تعينيات أخرى

عداد

يسمح لك بحساب تكرارات الكائنات القابلة للتجزئة

OrderedDict

فئة فرعية من القاموس تتيح لك معرفة ترتيب الإدخالات

com.defaultdict

فئة فرعية من القاموس تسمح لك بتحديد قيمة افتراضية في المنشئ

- 02 التعامل مع البيانات المجموعات



المجموعات - المسمّاة elput

الصف عبارة عن مجموعة غير قابلة للتغيير من البيانات غير المتغيرة في كثير من الأحيان.

```
(22.11)
طباعة(r) # طباعة(11, 22)
طباعة(r[0]) # طباعة11
طباعة(r[1]) # display22
```

تتيح لك فئة "namedtuple" في وحدة المجموعات إضافة أسماء صريحة إلى كل عنصر من عناصر tuple.
لتوسيع هذه المعاني في برنامج بايثون

من المجموعات، قم باستيراد namedtuple #import

النقطة = المسمّاة elput('نقطة', (['x', 'y']))

طباعة(النقطة(11, 22).x) # صن11 صن22 أو باستخدام اسم الحقل

طباعة(النقطة([0], [1])) # يقول من خلال فهارسها المعروضة: 33

طباعة(بكلسل([الجمل]) الوصول إليها عن طريق اسم العرض 33

طباعة(ع) # اعطه بأسلوب عرض الاسم = القيمة Point(x=11,y=22)

- 02 التعامل مع البيانات

المجموعات

المجموعات - المسمّاة elput

- تُقوم الطريقة `asdict()` بتحويل مثيل إلى معجم.
- يؤدي استدعاء `(p._asdict()` إلى إرجاع قاموس يعين أسماء كل حقل من حقل `p` إلى القيم المقابلة لها.

من مجموعات استيراد اسمه `elput`

النقطة = المسمّاة `elput('نقطة')`

ع=نقطة(11, ص=22)

`print(p._asdict()) #` يُرجع قاموساً ويطبع `{'x': 11, 'y': 22}`

- تُقوم الدالة `replace(key=args)` بإرجاع نسخة جديدة من صفنا بقيمة معدلة.

من مجموعات استيراد اسمه `elput`

النقطة = المسمّاة `elput('نقطة')`

ع=نقطة(11, ص=22)

`print(p._replace(x=4)) #` تغيير قيمة `x` في المطبوعات: `Point(x=4, y=22)`

- 02 التعامل مع البيانات

المجموعات



المجموعات - المسمّاة elputat

- تتيح لك وظيفة `mytuple._fields` استرجاع أسماء الحقول الموجودة في صفنا. يكون ذلك مفيداً إذا كنت تريد إنشاء صف جديد باستخدام حقول الصف الموجود.

من مجموعات الاستيراد اسمه Tuple

النقطة = المسمّاة elputat(['x','y'])

print(Point._fields) # يرجع أسماء الحقول المعروضة:

اللون = صف مسمى ("اللون", "أحمر أحضر، أزرق")

نقوم بإنشاء صف جديد باستخدام حقول النقاط والألوان Tuple ('Pixel', Point._fields + Color._fields) # باسمه Pixel=

طباعة (بكسل: (11.22.128.266.0)) #display: ص= 11، بكسل (مس= 22، ص= 128، أحمر= 266، أخضر= 0، أزرق= 0)

- 02 التعامل مع البيانات

المجموعات



المجموعات - ديك

• قوائم بايثون هي تسلسل مرتب من العناصر، قابلة للتغيير أو التعديل.

• يمكن لبايثون إضافة قوائم في وقت ثابت ولكن الإدراج في بداية القائمة يمكن أن يكون أبطأ (يزيد الوقت المطلوب مع نمو القائمة).

مثال :

```
Favorite_list = ["سامي", "جيسي", "ماري"]
#أدخل "Alice" في بداية قائمة الأسماك المفضلة
Favorite_list.insert(0, "Alice")
print(favorite_list) #poster ['Alice', 'Sammy', 'Jamie', 'Mary']
```

• فئة deque لوحدة المجموعات هي كائن من نوع القائمة يسمح لك بإدراج عناصر في بداية التسلسل أو نهايته مع أداء زمني ثابت. ($O(1)$)

• أداء ($O(1)$) يعني أن الوقت اللازم لإضافة عنصر إلى بداية القائمة لن يزيد، حتى لو كانت تلك القائمةآلاف أو ملايين العناصر.

02- التعامل مع البيانات المجموعات

المجموعات - ديك

- الوظائف إلهاق، (x) إلهاق اليسار (x): يضيف إلهاق قيمة واحدة إلى الجانب الأيمن من deque وإلهاق اليسار إلى الجانب الأيسر

مراجع

```
Favorite_deque = deque(["Sammy", "Jamie", "Mary"])
```

```
Favorite_fish_deque Favorite_deque.appendleft("Alice") #appen
```

```
Favorite_fish_deque.append("Bob") #append
```

```
print(favorite_deque) #print deque(['Alice', 'Sammy', 'Jamie', 'Mary', 'Bob'])
```

- وظائف : clear() و pop() و popleft() و popleft() و pop() تسمح لك بـ إزالة كائن من deque و يقوم Clear() بـ إفراغه.

من مجموعات الاستيراد deque

```
Favorite deque = deque(['Alice', 'Sammy', 'Jamie', 'Mary', 'Bob'])
```

Favorite_deque.pop() #display Bob

```
Favorite_deque.popleft() #display: Alice
```

طباعة لا شيء(favorite_deque.clear()) #display

- 02 التعامل مع البيانات

المجموعات

المجموعات - خريطة السلسلة

- * تسمح لك فئة `Collections.ChainMap` بربط العديد من التعيينات بحيث يتم إدارتها كواحد.
- مجموعات الفتنة (`spam*`) تقوم هذه الوظيفة بإرجاع `ChainMap` جديد. إذا لم تكن هناك خرائط محددة في المعلمات، فستكون `ChainMap` فارغة.

مثال :

من المجموعات قم باستيراد `ChainMap`

```
س = {"أ": 1, "ب": 2}
```

```
ص = {"ب": 10, "ج": 11}
```

```
ض = تشين ماب (ص, س)
```

```
for k, v in z.items(): #walk over items in z
```

طباعة (ك، الخامس)

ملخص:

1#

01#

11#

- في المثال السابق لاحظنا أن المفتاح `أ`أخذ القيمة 10 وليس 2 لأنه تم تمرير `z` قبل `ChainMap`.

- 02 التعامل مع البيانات

المجموعات

المجموعات - العداد

فئة `Collections.Counter` هي فئة فرعية من الإيماء.

والذي يسمح لك بحساب الكائنات القابلة للتجزئة.

في الواقع هو قاموس مع العناصر كمفتوح

وكما القيم عدهم.

*مجموعات الطبقة. مكافحة([elbareti]) [or-mapping]

إرجاع عدد. تسمح لك الوسيطة بتحديد هذا

التي نريد أن نضعها والتي يجب أن تحسب.

مثال :

من عدد استيراد المجموعات

`# العداد()`

`طبيعة(ج): العداد()`

`Counter('gallahad') #counter`

`print(c) #print Counter({'a': 3, 'l': 2, 'g': 1, 'h': 1, 'd': 1})`

`Counter({'red': 4, 'blue': 2}) #`

`print(c) #affcihe: Counter({'red': 4, 'blue': 2})`

`# العدد المليون في الكلاب (8)`

`Counter({' الكلاب': 8, 'قطط': 4})`

- 02 التعامل مع البيانات

المجموعات



المجموعات - العداد

- إذا طلبنا قيمة ليست في قائمتنا فإنها ترجع 0 وليس KeyError

من عداد استيراد المجموعات

```
ج = العداد(["البيض", "لحم الخنزير"])
```

```
c['bacon'] # مفتاح غير معروف، يعرض: 0
```

- دالة Elements(): تقوم بإرجاع قائمة بجميع عناصر العداد:

من عداد استيراد المجموعات

```
ج = العداد (أ=4, ب=2, ج=0, د=2)
```

```
sorted(c.elements()) #display ['a', 'a', 'a', 'a', 'b', 'b']
```

- الدالة: Most_common([n]): تُرجع العناصر n الأكثر تواجدًا في العداد

من عداد استيراد المجموعات

```
ج = العداد (أ=4, ب=2, ج=0, د=2)
```

```
print(Counter('abracadabra').most_common(3)) #print [('a', 5), ('b', 2), ('r', 2)]
```

- 02 التعامل مع البيانات

المجموعات



المجموعات - العدد

• وظيفة الطرح([قابلة للتكرار أو التعبيين]) : تتيح لك طرح العناصر من العدد (ولكن ليس حذفها)

من عداد استيراد المجموعات

ج = العدد (أ=4, ب=2, ج=0, د=2-)

د = العدد(أ=1, ب=2, ج=3, د=4)

```
print(c.subtract(d)) #affiche Counter({'a': 3, 'b': 0, 'c': -3, 'd': -6})
```

- 02 التعامل مع البيانات

المجموعات

المجموعات - أمرديكت

• تشبه الإملاءات. لكنهم يتذكرون الترتيب الذي تم به إدخال Collections.OrderedDict القيم. إذا كررنا ذلك، فسيتم إرجاع البيانات بترتيب الإضافة في إملاءنا.

• الدالة: popitem(last=True) تقوم بإخراج زوج من المفاتيح والقيمة من قاموسنا وإذا كانت الوسيطة الأخيرة هي "True" فسيتم إرجاع الأزواج في LIFO وإلا فسيتم وضعها في FIFO.

• الوظيفة move_to_end(key, last=True): تسمح لك بنقل المفتاح إلى نهاية قاموسنا إذا كان last صحيحًا وإلا إلى بداية الإملاء.

مثال :

```
من المجموعات 'd['b']=2' d['c']=3' d['d']=4'
d=OrderedDict() d['a']=1' #fill d
import OrderedDict

d.move_to_end('b')
print(d) #print OrderedDict([('a', '1'), ('c', '3'), ('d', '4'), ('b', '2')])

d.move_to_end('b',last=False)
print(d) #print OrderedDict([('b', '2'), ('a', '1'), ('c', '3'), ('d', '4')])

#print ('d', '4')( صحيح ) (d.popitem()

print(d) #print OrderedDict([('b', '2'), ('a', '1'), ('c', '3')])
```



- 02 التعامل مع البيانات

المجموعات



المجموعات - الإفتراضي

- الإفتراضي من وحدة المجموعات التي تسمح لك بجمع المعلومات في القواميس بسرعة وإيجاز.
 - لا يثير خطأ KeyError لأن defaultdict لا يغير أبداً.
- في حالة عدم وجود مفتاح، يقوم defaultdict بإدراج قيمة بديلة وإرجاعها بدلاً من ذلك.
- يتصرف الإفتراضي بشكل مختلف عن القاموس العادي. بدلاً من رفع خطأ KeyError على مفتاح مفقود، يستدعي التابع defaultdict قيمة الاستبدال بدون وسائل إنشاء كائن جديد.

في المثال أدناه، () والإنشاء قائمة فارغة.

من مجموعات الاستيراد الإفتراضي

```
my defaultdict = defaultdict(list)
```

```
#affcihe [] ([]gnissim")tcidtluafed_ym
```

- 02 التعامل مع البيانات

المجموعات



المجموعات - الإفتراضي

من مجموعات الاستيراد الافتراضي

تعريف: default_message()

إرجاع "المفتاح غير موجود"

```
defaultdict_obj= defaultdict(default_message)
```

"القيمة1"=["1yek"]jbo_ticdtluafed

"القيمة2"=["2yek"]jbo_ticdtluafed

```
print(defaultdict_obj["key1"]) #display: value1
```

```
print(defaultdict_obj["key2"]) #display: value2
```

print(defaultdict_obj["key3"]) #shows:

ملء الإملاء الافتراضي

عرض عناصر الافتراضي

في حالة عدم وجود المفتاح، قم باستدعاء الدالة default_message



استخدام
وظيفة لاما

من مجموعات الاستيراد الافتراضي

defauldcit_obj= defaultdict(lambda:
"المفتاح مفقود") # إعلان دالة لاما لعرض رسالة

"القيمة1"=["1yek"]jbo_ticdtluafed

"القيمة2"=["2yek"]jbo_ticdtluafed

```
print(defauldcit_obj["key1"]) #display: value1
```

```
print(defauldcit_obj["key2"]) #display: value2
```

استدعاء دالة لاما، يطبع: المفتاح مفقود

02- التعامل مع البيانات

المجموعات



المجموعات - الافتراضي

عند توفير فئة `int` كوظيفة افتراضية، تكون القيمة الافتراضية التي يتم إرجاعها هي صفر.

من مجموعات الاستيراد الافتراضي

```
defaultdict_obj = defaultdict(int)
    "1"="القيمة1
    "2"="القيمة2
print(defaultdict_obj["key1"]) #display: value1
print(defaultdict_obj["key2"]) #display: value2
(defaultdict_obj["key3"]) #display: 0
```

*يمكننا أيضًا استخدام تعيين الكائن ككائن افتراضي

من مجموعات الاستيراد الافتراضي

```
defaultdict_obj = defaultdict(set)
    "1"="القيمة1
    "2"="القيمة2
print(defaultdict_obj["key1"]) #display: value1
print(defaultdict_obj["key2"]) #display: value2
(defaultdict_obj["key3"]) #display: set()
```



الفصل 2

معالجة البيانات

1. القائمة
2. المجموعات
3. الملفات

- 02 التعامل مع البيانات

ملفات



WEBFORCE
BE THE CHANGE

العمل مع الملفات

- لدى Python العديد من الوظائف لإنشاء الملفات النصية وقراءتها وتحديثها وحذفها.

- الوظيفة الرئيسية للعمل مع الملفات في بايثون هي `open()`.

- يستغرق معلمتين. اسم الملف ووضعه.

- هناك أربع طرق (أوضاع) مختلفة لفتح ملف:

- "ص" - قراءة - الافتراضي. يفتح ملفاً للقراءة، خطأ إذا كان الملف غير موجود

- "أ" - إضافة - فتح ملف لإضافته وإنشاءه
الملف إذا لم يكن موجوداً

- "w" - Write - يفتح ملفاً للكتابة، ويقوم بإنشاء الملف إذا لم يكن موجوداً

- "x" - إنشاء - إنشاء الملف المحدد، وإرجاع خطأ في حالة وجود الملف

- فتح وقراءة ملف

- لفتح الملف، استخدم وظيفة `open()` المضمنة

- تقوم الدالة `open()` بإرجاع كائن ملف، والذي يحتوي على طريقة `read()` لقراءة محتويات الملف

```
f = open("demofile.txt", "r")
```

طباعة `f.read()`

- قم بإرجاع الأحرف الخمسة الأولى من الملف:

```
f = open("demofile.txt", "r")
```

طباعة `f.read(5)`

- 02 التعامل مع البيانات

ملفات



WEBFORCE
BE THE CHANGE

العمل مع الملفات

- يمكنك إرجاع سطر باستخدام طريقة : readline()

```
f = open("demofile.txt", "r")
print(f.readline())
```

- الكتابة إلى ملف

- للكتابة إلى ملف موجود، يجب عليك إضافة معلمة إلى الدالة: open()

```
f = open("demofile2.txt", "a") #
#يحتوي الملف على نص أكثر) #أضاف العبارة "يحتوي الملف على نص أكثر"
f.write("New text content!") #إغلاق الملف
f.close() #
```

افتح الملف في وضع الكتابة عن طريق حذف محتواه القديمة

"عيي!! تم حذف المحتوى القديم" #اكتب عبارة عيي!! يتم حذف المحتوى القديم

إغلاق الملف
f.close() #

```
f = open("demofile3.txt", "r")
```

افتح الملف واقرأه بعد إضافته
print(f.read()) #

- 02 التعامل مع البيانات

ملفات



WEBFORCE
BE THE CHANGE

العمل مع الملفات

• إغلاق ملف

- يوصى بإغلاق الملف دائمًا عند الانتهاء منه.

```
f = open("demofile.txt", "r")
print(f.readline())
f.close()
```

• حذف من ملف

- لحذف ملف، تحتاج إلى استيراد وحدة نظام التشغيل وتشغيل وظيفة `os.remove()` الخاصة بها:

```
import os
os.remove("demofile.txt")
```

02- التعامل مع البيانات

ملفات



تنسيق CSV

- هناك تنسيقات تخزين البيانات القياسية المختلفة. يوصى بتفضيل هذه التنسيقات نظرًا لوجود وحدات Python النمطية بالفعل لتبسيط استخدامها.

ملف القيم المفصولة بفواصل (CSV) هو تنسيق لتخزين الجداول في ملف نصي.

يتم تمثيل كل سطر بسطر من النص ويتم فصل كل عمود بفاصيل (فاصلة، فاصلة منقوطة، وما إلى ذلك). يمكن أيضًا تحديد الحقول النصية بعلامات اقتباس.

البيانات على شكل جدول

الإسم الأول	الإسم الأول	العمر
دوبوا ماري	دوبوا ماري	29
جوليان "بول"	جوليان "بول"	47
جاكيه برنارد	جاكيه برنارد	51
مارتن لوسي، كلارا	مارتن لوسي، كلارا	14

البيانات في شكل ملف CSV

عندما يحتوي الحقل نفسه على علامات اقتباس، يتم مضاعفة هذه العلامات حتى لا تعتبر بداية الحقل أو نهايته.

- إذا كان الحقل يحتوي على علامة يمكن استخدامها كفاصيل أعمدة (فاصلة، فاصلة منقوطة، وما إلى ذلك) أو كفاصيل أسطر، فإن علامات اقتباس تكون إلزامية حتى لا يتم الخلط بين هذه العلامة والفاصيل.

الإسم الأخير؛ الإسم الأول؛ العمر

"دوبوا"؛"ماري"؛ 29

"دوفال"؛"جوليان"؛"بول"؛ 74؛

سترة؛ برنارد؛ 51

مارتن؛"لوسي"؛"كلارا"؛ 41；

02- التعامل مع البيانات

ملفات



WEBFORCE
BE THE CHANGE

تنسيق CSV

• تعلم وحدة CSV الخاصة بـ Python على تبسيط استخدام ملفات CSV

• قراءة ملف CSV

• لقراءة ملف CSV، يجب عليك فتح دفق قراءة الملف وفتح قارئ CSV من هذا الدفق.

استيراد CSV # مكتبة استيراد CSV

ملف = مفتاح ("exampleCSV.csv", "r")

قارئ CSV = csv.reader(file, delimiter=";") # فتح قارئ CSV عن طريق توفير الحرف الفاصل (هنا ";")

للتخطي فلوك: CSV

CSV # عرض كل سطر من ملف CSV

file. Close() # إغلاق ملف CSV

عرض:

[["الاسم", "الاسم الأول", "العمر"]]

["دوبوا", "ماري", "29"]

["دوفال", "جولييان", "47"]

["سترة", "برنارد", "51"]

["مارتن", "لوسي", "14"]

• من الممكن أيضًا قراءة البيانات والحصول على قاموس سطري يحتوي على البيانات باستخدام DictReader بدلاً من القارئ

استيراد CSV # مكتبة استيراد CSV

الملف = مفتاح ("names.csv", "r")

قارئ CSV = csv.DictReader(file, delimiter =";")

للتخطي في قارئ CSV: طباعة (خط)

ملف. إغلاق()

عرض:

{"الاسم": "دوبوا", "الاسم الأول": "ماري", "العمر": "29"}

{"الاسم": "دوفال", "الاسم الأول": "جولييان", "العمر": "47"}

{"الاسم": "سترة", "الاسم الأول": "برنارد", "العمر": "51"}

{"الاسم": "مارتن", "الاسم الأول": "لوسي", "العمر": "14"}

- 02 التعامل مع البيانات

ملفات



تنسيق CSV

- الكتابة إلى ملف CSV
- مثل القراءة، نفتح تدفق الكتابة (وظيفة الكاتب (ونفتح كاتب CSV عليه writerow() : وظيفة من هذا الدفق)

استيراد CSV # مكتبة استيراد CSV

```
#فتح ملف في وضع الكتابة
file = open("directory.csv", "w")
#يفتح دفق الكتابة
writer = csv.writer(file, delimiter=";")

#اكتب السطر الأول في ملف Directory.csv
writer.writerow(["الاسم الأول", "الاسم العائلة", "الهاتف"])

#اكتب السطرين الثاني والثالث في ملف Directory.csv
writer.writerow(["Dubois", "Marie", "0198546372"])
writer.writerow(["Duval", "Julien", "0399741052"])

#اكتب السطرين الرابع والخامس في ملف Directory.csv
writer.writerow(["Jacquet", "Bernard", "0200749685"])
writer.writerow(["Martin", "Julie", "0399731590"])

#إغلاق الملف
file.close()
```

إنشاء ملف Directory.csv

لهم العائلة الاسم الأول	إسم العائلة
دوبيوا ماري	0198546372
جوفلان	0399741052
جاكيه برنارد	0200749685
مارتن جولي	0399731590

- 02 التعامل مع البيانات

ملفات



WEBFORCE
BE THE CHANGE

تنسيق CSV

- ومن الممكن أيضًا كتابة الملف من خلال توفير قاموس واحد في كل سطر على أن يكون لكل قاموس نفس المفاتيح.

يعتمد على الالة التي تتوفر في المكتبة لفتح الملفات للاخراج من القواميس

```
استيراد CSV # مكتبة استيراد CSV
نماذج الطلب = [{"product": "notebook", "reference": "F452CP", "quantity": 41, "Unitprice": 1.6}
 {"المنتج": "قلم أزرق", "مرجع": 18, "الكمية": "D857BL", "سعر الوحدة": 0.95},
 {"المنتج": "قلم أسود", "مرجع": 18, "الكمية": "D857NO", "سعر الوحدة": 0.95}
 # إعلن القوميس
 # افتح ملفاً واكتبه
 file = open("bon-commande.csv", "w")
 كاتب = csv.DictWriter(file, delimiter = ";",
                        fieldnames = goodOrder[0].keys())
 # إنتاج خطوط الإخراج من القوميس،
 # أسماء الحقول تحتوي على مفاتيح القوميس
 الكاتب.writeheader()
 للسطر في GoodCommand: # تصفح القوميس
 # اكتب سطراً في الملف
 الكاتب.writerow(line)
 ملف إغلاق()
```

إنشاء ملف order-order.csv

دفتر ملاحظات:
 المرجع: الكمية: المنتج: F452CP: 41!!
 قلم أزرق: 18: D857BL: 0.95
 قلم أسود: 18: D857NO: 0.95

- 02 التعامل مع البيانات

ملفات



تنسيق جيسون

- يأتي تنسيق JavaScript Object Notation (JSON) من ملف تدوين الكائنات في لغة جافا سكريبت.
- وهو الآن تنسيق بيانات شائع جدًا السماح بتخزين البيانات في النموذج منظم.
- يتضمن فقط الارتباطات الرئيسية [] القيم (في مثل القواميس)، وكذلك القوائم المرتبة من القيم (مثل القوائم في بايثون).
- يمكن أن تكون القيمة ارتباطاً رئيسياً آخر [] القيم، قائمة القيم، عدد صحيح، عدد حقيقي، سلسلة أحرف أو قيمة منطقية أو فارغة.
- بناء الجملة يشبه قواميس بايثون.

مثال لملف JSON لتعريف القائمة إنه كائن يتكون من أعضاء وهم سمة (ملف) ومصفوفة (أوامر)

الذي يحتوي على كائنات أخرى: خطوط القائمة. يتم تعريف خط القائمة من خلال عنوانه وعمله

مثال لملف JSON

{ "القائمة": "ملف", "الأوامر": [}

 { " جديد": { "title": "CreateDoc", "action": "CreateDoc" },

 { "فتح": { "title": "OpenDoc", "action": "OpenDoc" },
 "إغلاق": "

 "CloseDoc": "الإجراء"

}

]

02- التعامل مع البيانات

ملفات



تنسيق جيسون

قراءة ملف JSON

* تتيح لك وظيفة التحميل (نص JSON) الذي تم ترميره كوسقطة وتحويله إلى قاموس أو قائمة.

استيراد مكتبة json

```
#فتح ملف example.json للقراءة
file = open("example.json", "r")
#فك تشفير المحتوى
x=json.load(file)
#طباعة (خ)
```

عرض:

```
{'قائمة': {'ملف': 'أوامر', '[new]: 'جديد', 'title': 'Open', 'action': 'OpenDoc'}, 'إغلاق': {'title': 'Close', 'action': 'CloseDoc'}}]
```

كتابة ملف JSON

* تقوم وظيفة dumps (variable,sort_keys=False) بتحويل القاموس أو القائمة إلى نص JSON عن طريق توفير المتغير المطلوب تحويله كوسقطة.

استيراد مكتبة json

```
#تعريف قاموس
quantitySupplies= {"notebooks":134, "pens": 14, "eulb":47, "der":14, "محابات":58} #فتح ملف للكتابة
file=open ("quantiteFournitures.json", "w")
#تحويل القاموس إلى نص
file.write(json.dumps(quantitySupplies)) #
#إغلاق الملف
```



استخدم التعبيرات العادية



ما ستعلم في هذا الفصل:

- التعرف على مبدأ التعبيرات العادية
- التعامل مع وظائف التعبيرات العادية، وهي وظائف البحث والقسمة والاستبدال





استخدم التعبيرات العاديّة



1. إنشاء التعبيرات العاديّة . 2. التعامل مع التعبيرات العاديّة

- 103 استخدم التعبيرات العاديّة

إنشاء التعبيرات العاديّة



العبارات العاديّة الأساسية

بهدف التوجّه القولي العادي للأقواء علماً بـ [aeiouy] يطّوّج الأئمّة في التقسيمة حتّى يُؤثّر على جهاده.

مثال: [aeiouy] يمثل حرف علة، [dt] ويشير إلى Durand أو Durant.

• بين القوسين، يمكنك ملاحظة الفاصل الزمني باستخدام الوصلة.

مثال: يمثل [0-9] الأرقام من 0 إلى 9، ويتمثل [a-zA-Z] حرفاً صغيراً أو كبيراً.

• يمكننا أيضًا استخدام اللكتنة المنعطفة في الموضع الأول بين القوسين للإشارة إلى عكس ذلك

مثال: يمثل [az^] شيئاً آخر غير الحرف الصغير، و[^a] ليس فاصلة عليا ولا علامة اقتباس.

3. العلامة النجمية هي محدد كمي، فهي تعني صفر أو أكثر من تكرارات الحرف أو العنصر الذي يسبقه مباشرة.
* مثال: التعبير a^* يعني الحرف a متبعًا بصفر أو أكثر من الأحرف، على سبيل المثال ababab أو a[AZ] و bbabb يتوافق مع صفر أو أكثر من الأحرف الكبيرة.

مثال: يمثل التعبير t^{9c} جميع مجموعات الأحرف الثلاثة التي تبدأ بـ "t" وتنتهي بـ "c" مثل tac أو $tqac$ أو $t9c$.

- ١٠٣ استخدم التعبيرات العادبة

إنشاء التعبيرات العادبة

العبارات العادبة الأساسية

4. اللهجة المنعطفة ^٨ هي مرساة. يشير إلى ذلك التعبير الذي يليه موجود في بداية السطر.

مثال: التعبير Since^٨ يشير إلى أننا نبحث عن السطور التي تبدأ بكلمة منذ.

5. رمز الدولار \$ هو أيضًا مرساة. يشير إلى ذلك التعبير الذي يسبقه يكون في نهاية السطر.

مثال: التعبير التالي: \$ يشير إلى أننا نبحث عنه الأسطر التي تنتهي بـ "التالي": .

6. الشرطة المائلة العكسية \ تسمح لك بالهروب من المعنى الأحرف الأولية. لذا . يعين نقطة حقيقة، *علامة النجمة، ٨ علامة الإقحام، \$ دولار و ١١ شرطة مائلة عكسية

العبارات العادبة الموسعة

يضيفون خمسة رموز لها المعاني التالية:

1. زوج الأقواس: () يستخدم لتشكيل أنماط فرعية ولتحديد التعبيرات الفرعية، مما يسمح باستخراج أجزاء من سلسلة الأحرف.

مثال: التعبير (إلى)* سيشير إلى، توتوا، إلخ.

2. علامة الجمع : + هي محدد كمي مثل ، * ولكنها تشير إلى حدوث واحد أو أكثر للحرف أو العنصر الذي يسبقها مباشرة.

مثال: التعبير + يعني الحرف a متبعًا بحرف أو أكثر

3. علامة الاستفهام ؟ : تعني صفر أو مثيل واحد من التعبير الذي يسبقها.

مثال: الشاشة (الشاشات)؟ تعني الشاشة أو الشاشات؛

- ١٠٣ استخدم التعبيرات العادبة

إنشاء التعبيرات العادبة

العبارات العادبة الموسعة

زوج القوسين : { } يحدد عدد التكرارات المسموح بها للنطاق الذي يسبقه.

مثال: {2,5}[0-9] يتوقع ما بين رقمين وخمسة أرقام عشرية.

• تتوفر المتغيرات التالية: 0-

{2} يعني حدوثين على الأقل للأعداد الصحيحة العشرية و[9-0] {2} حدفين بالضبط

5. الشريط العمودي : | يمثل اختيارات متعددة في النطاق الفرعي.

مثال: يمكن أيضًا كتابة التعبير Duran | Durant. (Durand | Durant). يمكننا استخدام التعبير (dur | ma | me | th | ve | sa | sun) عند كتابة التاريخ.

يتضمن بناء الجملة الموسع أيضًا سلسلة من التسلسلات العادم:

\d: رمز الهروب؛

\e: تسلسل التحكم في الهروب؛

\f: فاصل الصفحات؛

\n: نهاية السطر؛

\r: حرف العودة؛

\t: اعلامة تبوييب أفقيه؛

\v: اعلامة تبوييب عمودية؛

\d: فئة الأعداد الصحيحة؛

\s: فئة الأحرف الفضائية؛

\w: فئة الأحرف الأبجدية الرقمية؛

\b: المحددات في بداية الكلمة أو نهايتها؛

\D: نفي الفئة؛

\S: نفي الفئة؛

\W: نفي الفئة؛

\B: نفي الفئة.



الفصل 3

استخدم التعبيرات العادية



١. إنشاء التعبيرات العادية . ٢. التعامل مع التعبيرات العادية

- 103 استخدم التعبيرات العادي

التعامل مع التعبيرات العادي

خيارات التجميع:

- وحدة إعادة تسمح لك باستخدام التعبيرات العادي في نصوص بايثون.
- ولذلك يجب أن تتضمن البرامج النصية السطر: `import re`
- وظيفة تجميع التعبير العادي في بايثون:
- لتهيئة تعبير عادي مع بايثون، من الممكن تجميعها، خاصة إذا كنت ستستخدمها عدة مرات خلال البرنامج. للقيام بذلك، يجب عليك استخدام وظيفة الترجمة.()

• بفضل مجموعة من خيارات التجميع، من الممكن التحكم في سلوك التعبيرات العادي لهذا نستخدم بناء الجملة (...؟) مع العلامات التالية:

- أ: مطابقة ASCII (Unicode افتراضياً):
- أمطابقة غير حساسة لحالة الأحرف;
- لـ: تستخدم المراسلات المكان، أي خصوصيات البلد;
- م: تطابق في سلاسل متعددة الأسطر;
- س: يعدل سلوك الحرف الأولي النقطي الذي سيمثل أيضا فاصل الأسطر;

```
إعادة الاستيراد
التعبير =re.compile(r"\d{1,3}")
```

• مطابقة Unicode (افتراضي):

س: الوضع المطول.

```
إعادة الاستيراد
#التحكم في سلوك Expression =re.compile(r"[az]+")
التعبير العادي مع حساسية الحالة
#التحكم في السلوكي Expression =re.compile(r" (?i) [az]+")
التعبير العادي دون حساسية لحالة الأحرف
```

103 - استخدم التعبيرات العادية

التعامل مع التعبيرات العادية

*توفر الوحدة 're' مجموعة من الوظائف التي تسمح لنا بالبحث عن سلسلة متطابقة:

وظيفة

إيجاد¹لقائمة تحتوي على كافة التطابقات

يبحث

أرسل كائن مطابقة إذا كان هناك تطابق في أي مكان في السلسلة

ينقسم

إرجاع قائمة حيث تم تقسيم السلسلة لكل مبارزة

الفرعية

يستبديل واحداً أو أكثر من التطابقات بسلسلة مرتبة

*تقوم الدالة ()all بإرجاع قائمة تحتوي على كافة التطابقات. تحتوي القائمة على التطابقات بالترتيب الذي هي عليه وجد. إذا لم يتم العثور على أي تطابق، يتم إرجاع قائمة فارغة.

مثال :

إعادة الاستيراد

Nameage = "جانيس تبلغ من العمر 22 عاماً وثيون يبلغ من العمر 33 عاماً"

غابريل يبلغ من العمر 44 عاماً وجوي يبلغ من العمر 21 عاماً

الأعمار . re.findall(r'\d{1,3}', # ابحث عن واحد أو اثنين أو ثلاثة أعداد صحيحة عشرية

عرض [21, '22', '33', '44']#

طباعة (العصور)

تحتوي على حرف صغيرة وكبيرة name = re.findall(r'[A-Z][az]*', Nameage) #strings

طباعة (أليساندرا، "أليساندرا، "ثيون، "ثيون، "غابريل، "جوي")

- 103 استخدم التعبيرات العادي

التعامل مع التعبيرات العادي

• تقوم وظيفة البحث (بالبحث في السلسلة عن التطابق وإرجاع كائن المطابقة إذا كان هناك تطابق. إذا كان هناك أكثر من تطابق، فسيتم إرجاع التواجد الأول فقط للمطابقة:

مثال: استخراج بسيط

• يستقبل متغير التعبير الصيغة المجمعة للتعبير العادي، ثم يطبق على هذا النمط المجمع طريقة البحث () التي ترجع الموضع الأول للنمط في سلسلة Nameage وتعيينه لمتغير الأعمار. أخيراً نعرض المطابقة الكاملة (من خلال عدم إعطاء وسيلة للمجموعة ())

```

    إعادة الاستيراد
    ""
    اسم الاسم = جانيس تبلغ من العمر 22 عاماً ونيون يبلغ من العمر 33 عاماً
    غابرييل يبلغ من العمر 44 عاماً وجوي يبلغ من العمر 21 عاماً
    ""

    # بحث عن عدد صحيح عشري واحد أو اثنين أو ثلاثة
    Expression = re.compile(r"\d{1,3}") #

    الأعمار= # يرجع الموضع الأول للنمط
    (egaemaN)hcraes.noisserpxe= طباعة(العصور)
    <re.Match object; span=22>
    display: = 22
    النتيجة = 22
    موقف النتيجة

```

- 103 استخدم التعبيرات العادي

التعامل مع التعبيرات العادي

مثال: استخراج المجموعات الفرعية

من الممكن تحسين عرض النتيجة عن طريق تعديل تعبير البحث العادي حتى تتمكن من التقاط عناصر النتيجة نمط

إعادة الاستيراد

```
Pattern_date = re.compile(r"(\d\d ?) (\w+) (\d{4})")
hcraes.etad_fitomcorresp =
    ("corresp.group(1):", "corresp.group(2)", "corresp.group(3))",
     ("corresp.group(1):", corresp.group(1)) #corresp.group(1): 14
     ("corresp.group(2):", corresp.group(2)),
     ("corresp.group(3):", "#corresp.group(3))",
      ("corresp.group(1,3):", corresp.group(1,3)) #corresp.group(1,3): ('14', '1789')
      ("corresp.groups()", "#corresp.groups()))"
```

103 - استخدم التعبيرات العادية

التعامل مع التعبيرات العادية



اسمي:

• بناء الجملة لإنشاء نمط اسمي: (?P=pattern_name); • بناء الجملة للإشارة إليه: (?P<pattern_name>);

لدي بايثون صيغة تسمح لك بتسمية أجزاء من النمط المفصول بين قوسين، وهو ما يسمى بالنمط

مثال: استخراج المجموعات الفرعية المسماة

• تقوم طريقة groupdict() بإرجاع قائمة بأسماء وقيمة المجموعات الفرعية التي تم العثور عليها (والتي تتطلب تسمية المجموعات الفرعية).

إعادة الاستيراد

```
Pattern_date = re.compile(r"(?P<day>\d\d ?) (?P<month>\w+) (\d{4})")
```

```
1789("اليوم 14"الباстиيل 1789)hcraes.etad_fitomcorresp =
```

```
#print(corresp.groupdict())
```

```
طباعة(14) #طباعة('اليوم', 'شهر', 'اليوم')
```

```
طباعة(pserroc("الشهر")) #طباعة('اليوم', 'شهر', 'اليوم')
```

- 103 استخدم التعبيرات العادلة

التعامل مع التعبيرات العادلة

- 103 استخدم التعبيرات العادية

التعامل مع التعبيرات العادية



* تقوم الدالة الفرعية () باستبدال التطابقات بالنص الذي تختاره

مثال: استبدل كل مسافة بواصلة :-'

إعادة الاستيراد

العنوان = "شارع 41 دي لا ريبابليك"

```
list=re.sub("\s","-",address) #affcihe: Rue-41-de-la-République
```

طباعة (قائمة)

من الممكن التحكم في عدد البدائل عن طريق تحديد معلمة العد:

مثال: العدد=2

إعادة الاستيراد

العنوان = "شارع 41 دي لا ريبابليك"

```
list=re.sub("\s","-",address,2) #poster Rue-41-de la République
```

طباعة (قائمة)



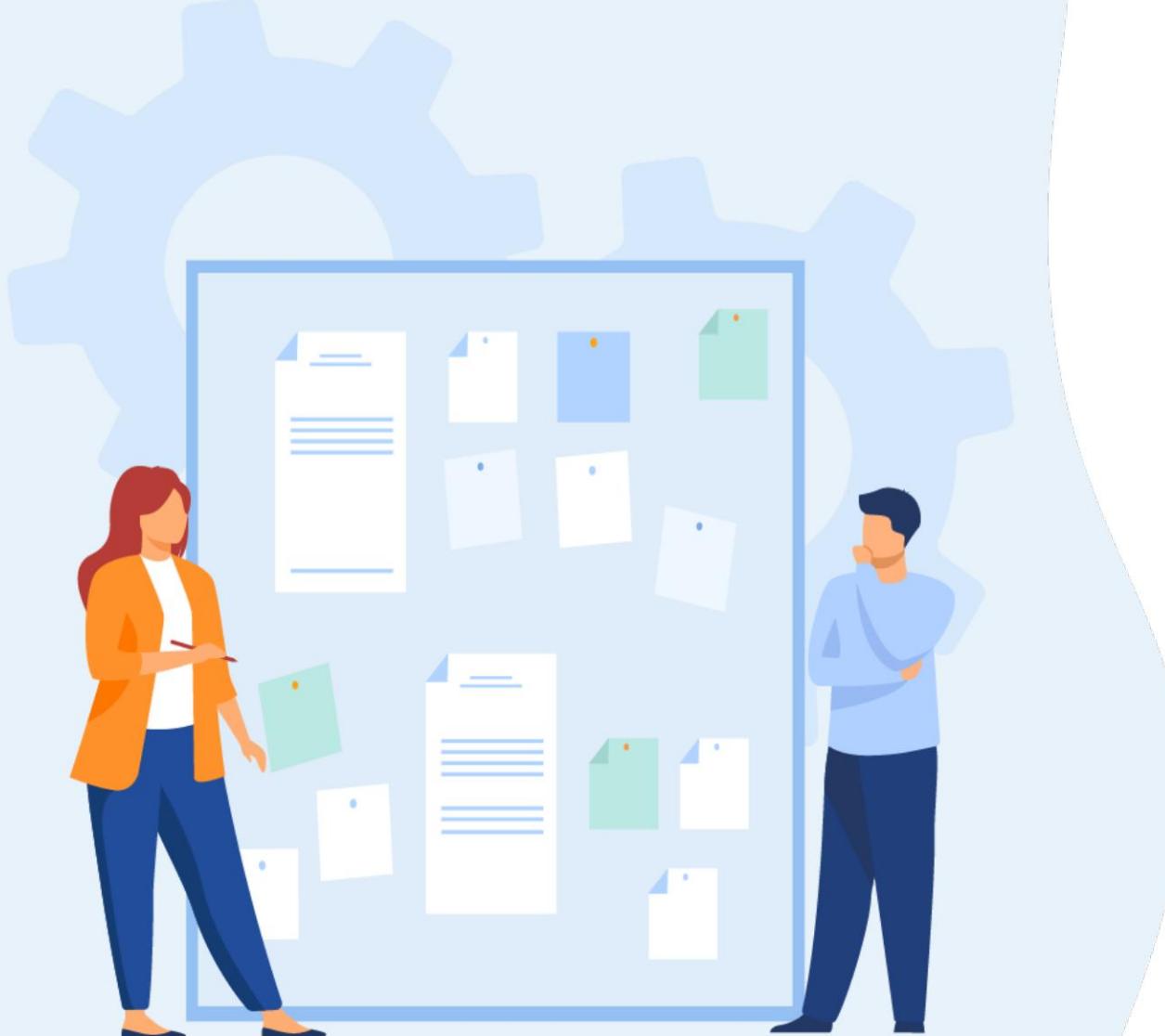
إدارة الاستثناءات



ما ستعلم في هذا الفصل:

- التعرف على أنواع الأخطاء المختلفة • معرفة الأنوع الرئيسية من الاستثناءات في بايثون
- إتقان التعامل مع الاستثناءات في بايثون

05 ساعات



الفصل 4

إدارة الاستثناءات

1. أنواع الأخطاء والتجريب .2. أنواع الاستثناءات .3. معالجة

الاستثناءات

04 - إدارة الاستثناءات

أنواع الأخطاء والتجريب



أخطاء في بناء الجملة

• أخطاء بناء الجملة هي أخطاء في تحليل التعليمات البرمجية

مثال:

```
<><> أثناء الطياعة الحقيقة ("مرحبا بالعالم")
 ملف ، "السطر 1 أثناء الطياعة الحقيقة
 ("Hello World")
 ^
```

خطأ قواعدي: بناء جملة غير صالح

• يشير المحلل إلى الخط المخالف ويعرض "سهمًا" صغيراً يشير إلى المكان الأول على الخط الذي تم اكتشاف الخطأ فيه.

• سبب الخطأ هو الرمز الموجود قبل السهم.

• في هذا المثال، يوجد السهم على الدالة print() لأن النقطتين (:) مفقودة قبله مباشرة. يتم عرض اسم الملف ورقم السطر لتسهيل تحديد موقع الخطأ عندما يأتي الرمز من برنامج نصي.

04 - إدارة الاستثناءات

أنواع الأخطاء والتجربة



الاستثناءات

• حتى لو كانت العبارة أو التعبير صحيحاً من الناحية النحوية، فقد يؤدي ذلك إلى حدوث خطأ عند تنفيذه.

• تُسمى الأخطاء التي يتم اكتشافها أثناء التنفيذ بالاستثناءات وهي ليست دائمًا قاتلة. ومع ذلك ، لا تتم معالجة معظم الاستثناءات بواسطة البرامج، مما يؤدي إلى ظهور رسائل خطأ مثل هذه:

* (1/0) >>> 10

التتابع (آخر مقالمة أخيرة):
خطأ: ZeroDivisionError: القسمة على صفر

• يشير السطر الأخير من رسالة الخطأ إلى ما حصل. يمكن أن تكون الاستثناءات من أنواع مختلفة ويتم الإشارة إلى هذا النوع في الرسالة: الأنواع المشار إليها في المثال هي ZeroDivisionError



الفصل 4

إدارة الاستثناءات

1. أنواع الأخطاء والتجريب .2. أنواع الاستثناءات .3. **معالجة**

الاستثناءات

04 - إدارة الاستثناءات

أنواع الاستثناءات

• في لغة بايثون، تسمى الأخطاء التي يتم اكتشافها أثناء تنفيذ البرنامج النصي بالاستثناءات لأنها تتوافق مع حالة "استثنائية" من البرنامج النصي.

النصي

• تقوم لغة بايثون بتحليل نوع الخطأ الذي تم تشغيله

• لدى بايثون العديد من فئات الاستثناءات الأصلية وأي استثناء هو مثيل (كائن) تم إنشاؤه من فئة استثناء

• فئة الاستثناء الأساسية للاستثناءات الأصلية هي `BaseException`

• أربع فئات استثناء مستمدّة من فئة `BaseException` وهي:

• جميع استثناءات النظام المضمنة لعدم الخروج مشتقة من هذه الفئة

• يجب أيضًا اشتقاق كافة الاستثناءات المعرفة من قبل المستخدم من هذه الفئة

استثناء

`SystemExit`

`GeneratorExit`

`KeyboardInterrupt`

• يتم تشغيله بواسطة الدالة (`sys.exit()`) إذا كانت القيمة المرتبطة عبارة عن عدد صحيح بسيط، فإنه يحدد حالة الخروج من النظام (يتم تمريرها إلى الدالة `exit()`) في لغة C

• يتم رفعه عند استدعاء طريقة إغلاق () الخاصة بالمولد

• يتم رفعه عندما يضغط المستخدم على مفتاح المقاطعة (عادة التحكم-C أو حذف)

04 - إدارة الاستثناءات

أنواع الاستثناءات



هناك أيضاً فئات استثناء أخرى مشتقة
استثناءات مثل:

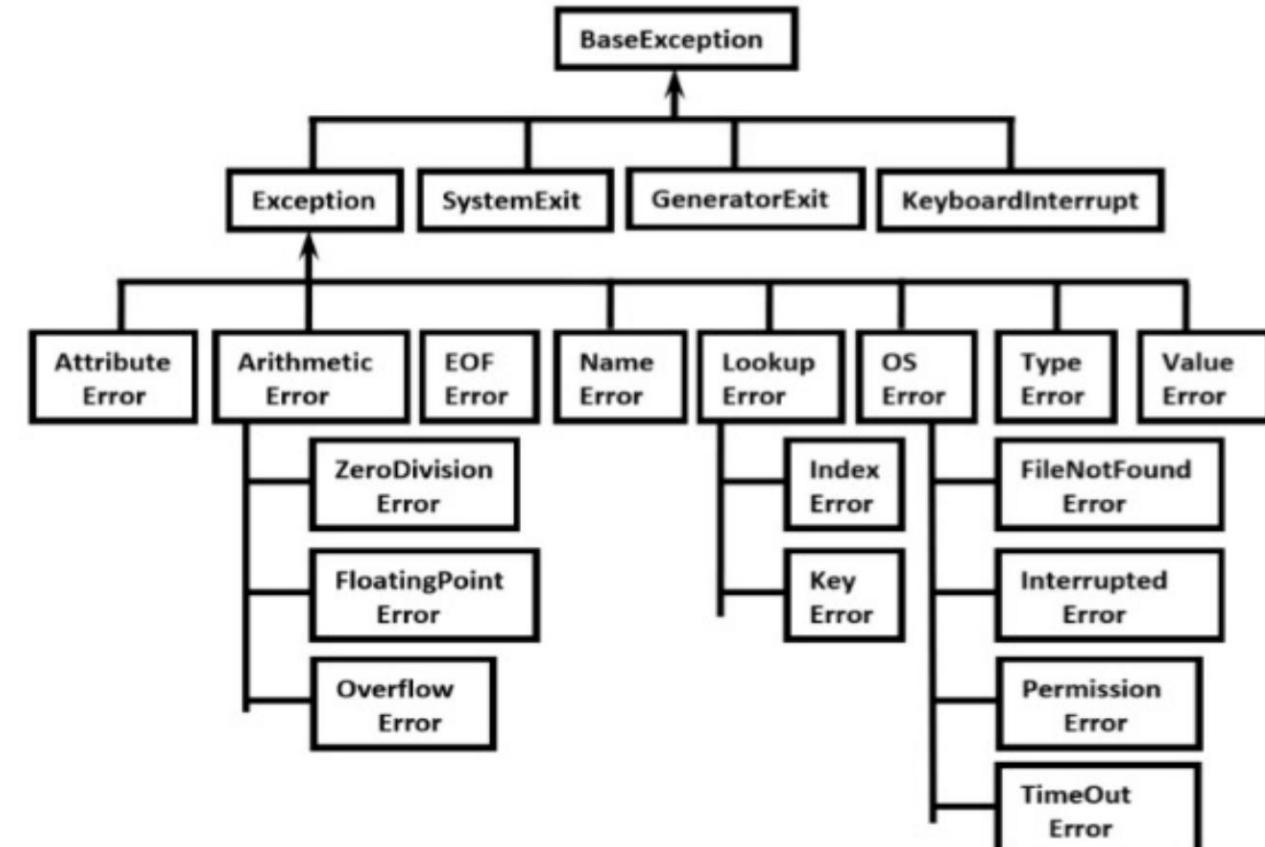
- فئة `ArithmetricError` هي الفئة الأساسية للاستثناءات الأصلية التي يتم طرحها لمختلف الأخطاء الحسابية وخاصةً لفئات `ZeroDivisionError` و `OverflowError` و `FloatingPointError`.

`gPointError` ;

- فئة `LookupError` هي الفئة الأساسية للاستثناءات التي يتم طرحها عندما يكون المفتاح أو الفهرس المستخدم في جدول البحث أو التسلسل غير صالح.

• ثم يتم اشتقاق العديد من الفئات من هذه الفئات.

اعتباراً على الخطأ الذي واجهه محلل بايثون،
كائن استثناء ينتمي إلى هذه الفئة أو تلك
سيتم إنشاء الاستثناء وإعادته. تم اعتراض هذا الكائن
والتلعب بها.





الفصل 4

إدارة الاستثناءات

1. أنواع الأخطاء والتجريب .2. أنواع الاستثناءات .3. معالجة

الاستثناءات

04 - إدارة الاستثناءات

معالجة الاستثناء



WEBFORCE
BE THE CHANGE

اكتشاف الاستثناءات ومعالجتها في بايثون

يمكن اكتشاف الاستثناءات عن طريق وضع تعليمات قد تولد استثناءات في كتلة المحاولة.

مثال :

- هناك شكلان متبادلان من تعبيرات المحاولة (يمكن استخدام واحد فقط في كل مرة): المحاولة باستثناء والمحاولة أخيراً.
- زر حساب العمر الذي أدخله المستخدم عن طريق طرح سنة ميلاده من 2016 وللقيام بذلك، يجب علينا تحويل قيمة سنة الميلاد المتفغرة إلى int.
- قد يفشل هذا التحويل إذا لم تكن سلسلة الأحرف التي أدخلها المستخدم رقمًا.
- من الممكن أن تكون عبارة المحاولة مصحوبة بواحدة أو أكثر من الجمل باستثناء، أو عبارةأخيرة واحدة ، أو مجموعة من المحاولة باستثناء النهاية.

سنة الميلاد = الإدخال ("سنة الميلاد؟")

يحاول:

طباعة ("لديك", int(سنة الميلاد), "سنوات.",) #رمز المحاطرة

يستثنى:

طباعة ("خطأ، الرجاء إدخال رقم.") #رمز للتنفيذ عند حدوث خطأ

طباعة ("نهاية البرنامج.")

04 - إدارة الاستثناءات

معالجة الاستثناء



محاولة باستثناء البيان

```
سنة الميلاد = الإدخال ("سنة الميلاد?")
طباعة ("الديك", "int", 2016(سنة الميلاد), "سنوات.") #رمز المخاطرة
يحاول:
طباعة ("خطأ, الرجاء إدخال رقم.") #رمز للتنفيذ عند حدوث خطأ
يستثنى:
طباعة ("خطأ") #رمز للتنفيذ عند حدوث خطأ
طباعة ("نهاية البرنامج.")
```

في الحالة الأولى، تم التحويل بشكل طبيعي، وبالتالي تمكنت كتلة المحاولة من التنفيذ بالكامل دون أخطاء.

في الحالة الثانية، يحدث خطأ في كتلة المحاولة أثناء التحويل. وبالتالي، يتوقف تنفيذ هذه الكتلة فوراً وينتقل إلى الكتلة باستثناء، قبل المتابعة أيضاً بعد تعليمات المحاولة باستثناء.

الحالة الأولى

- إذا أدخل المستخدم عدداً صحيحاً، فسيتم التنفيذ دون أخطاء ويتم حساب عمره وعرضه

سنة الميلاد ؟ 1994

عمرك 22 سنة.

نهاية البرنامج.

الحالة الثانية

- إذا قام المستخدم بإدخال أي سلسلة من الأحرف، والتي لا تمثل عدداً صحيحاً، فسيتم عرض رسالة خطأ

سنة الميلاد ؟ اثنين

خطأ, الرجاء إدخال رقم.

نهاية البرنامج.

04 - إدارة الاستثناءات

معالجة الاستثناء



نوع الاستثناء

- عند استخدام عبارة "محاولة باستثناء"، تلتقط كتلة الاستثناء جميع الأخطاء المحتملة التي قد تحدث في كتلة المحاولة المقابلة.
- يتم تمثيل الاستثناء فعليًا بواسطة كائن، مثيل للفئة استثناء.
- يمكننا استرجاع هذا الكائن عن طريق تحديد اسم المتغير بعد باستثناء.

يحاول:

```
#رمز غير آمن
i = int(input('i ? '))
```

```
#رمز غير آمن
j = int(input('j ? '))
```

```
print(i, '/', j, '=', i / j) #
```

باستثناء الاستثناء ك `# :المتغير e من النوع Exception`

طباعة (نوع (e)) استثناء

طباعة (e) #showex

- ولذلك فإننا نستعيد كائن نوع الاستثناء في المتغير `e`.

- في خانة الاستثناء نعرض نوعها وقيمتها.

04 - إدارة الاستثناءات

معالجة الاستثناء



نوع الاستثناء

تشغيل الأمثلة التي تكشف عن نوعين مختلفين من الأخطاء:

إذا لم يتم توفير عدد صحيح، فلا يمكن تحويله إلى int ويحدث خطأ في نوع :
`ValueError`

```
أنا؟ ثلاثة
#ملصق:
<الفئة "خطأ في القيمة">
#حرف غير صالح لـ int() ذو الأساس 10: 'three'
```

إذا قدمنا قيمة 0 لـ b ، فسيكون لدينا قسمة على صفر مما ينتج عنه خطأ من نوع:
`ZeroDivisionError`:

```
أنا؟ 5
ي؟ 0
#ملصق:
<الفئة ZeroDivisionError>
#القسمة على صفر
```

04 - إدارة الاستثناءات

معالجة الاستثناء



التقاط خطأ محدد

لذلك يتم تعريف كل نوع من الأخطاء بواسطة فئة معينة.

- من الممكن ربط العديد من الكتل مع نفس كتلة المحاولة، لتنفيذ تعليمات برمجية مختلفة اعتماداً على الخطأ الذي تم التقاطه.
- عند حدوث خطأ، يتم فحص الكتل الاستثناء واحدة تلو الأخرى، من الأولى إلى الأخيرة، حتى يتم العثور على الكتلة المطابقة إلى الخطأ الذي تم التقاطه.

يحاول:

`أنا = كتافة العمليات (الإدخال ("أنا"))`

`ي = tni=(الإدخال(j)?")`

`طباعة (ط، '/ي، '= ط / ي)`

باستثناء خطأ القيمة:

`طباعة ("خطأ في التحويل.")`

`ZeroDivisionError:`

`طباعة ("القسمة على صفر.")`

يستثني:

`طباعة ("خطأ آخر.")`

مثال :

عند حدوث خطأ في كتلة المحاولة، سيتم تنفيذ كتلة واحدة فقط من الكتل باستثناء، اعتماداً على نوع الخطأ الذي حدث.

آخر كتلة باستثناء موجودة للقبض على كافة الأخطاء الأخرى.

يعد ترتيب الكتل باستثناء أمراً مهماً جدًا ويجب ترتيبها من الأكثر تحديداً إلى الأكثر عمومية، بحيث تأتي الكتلة الافتراضية في النهاية.

04 - إدارة الاستثناءات

معالجة الاستثناء



معالج الأخطاء المشتركة

- من الممكن تشغيل نفس الكود لأنواع مختلفة من الأخطاء، وذلك بإدراجها في صنف بعد الكلمة المحجوزة باستثناء.
- إذا أردنا تشغيل نفس الكود للتحويل والقسمة على صفر خطأ، فعليينا أن نكتب:

يحاول:

```
أنا = (الإدخال(ا؟ا))
```

```
ي = (الإدخال(j؟j))
```

طباعة (ط، ،'ي، ،'=ط / ي)

باستثناء (ValueError, ZeroDivisionError) كـ e:

```
# (خطأ في الحساب:، # قم بتشغيل نفس الكود لأنواع مختلفة من الاستثناءات
print("خطأ في الحساب:",
```

يستثنى:

طباعة ("خطأ آخر.")

04 - إدارة الاستثناءات

معالجة الاستثناء



كتلة أخيرا

• تتيح لك الكلمة الممحوzaة أخيراً إدخال كتلة سيتم تنفيذها إما بعد تنفيذ كتلة المحاولة بالكامل دون خطأ، أو بعد تنفيذ كتلة الاستثناء المقابلة للخطأ الذي حدث أثناء تنفيذ كتلة المحاولة.

إذا قدم المستخدم القيم الصحيحة ل `bo` فسيكون العرض كما يلي:

```
طباعة("بدء الحساب.")  
يحاول:  
أنا = كافة العمليات (الإدخال ("أنا"))  
ي = tni=(الإدخال(j?" ))  
طباعة ("النتيجة:", ط / د)  
يستثنى:  
طباعة ("خطأ.")  
أخيراً:  
طباعة ("تنظيف الذاكرة.")  
طباعة ("نهاية الحساب.")
```

```
بداية الحساب.  
أنا؟ 2 أيام؟ 8 النتيجة:  
0.25  
  
تنظيف الذاكرة.  
نهاية الحساب.
```

في حالة حدوث خطأ، يكون العرض كما يلي:

```
أنا؟ 2 أيام؟ 0 خطأ.  
  
تنظيف الذاكرة.  
نهاية الحساب.
```

في كلتا الحالتين تم تنفيذ الكتلة النهائية

04 - إدارة الاستثناءات

معالجة الاستثناء



توليد الخطأ

- في الواقع يكفي ببساطة استخدام الكلمة الممحوزة `raise` متبوعة بإشارة إلى كائن يمثل أستثناء.

مثال :

```
تعريف العامل (أ):  
إذا كان: 0 < n:  
    رفع # ArithmeticError إذا كان n < 0 #أبلغ عن خطأ ArithmeticError  
    إذا كان: 0 == 1:  
        العودة 1  
    العودة قررت (أ) - 1
```

مثلاً بالفاموجن التواليد يعطيه في الكيرنالتقاط الاستثناء على وجه التحديد لـ `ArithmeticalException` عند استدعاء الدالة الحقيقة.

يحاول:

```
طباعة # factorial (a) #أدخل رقمًا: #رمز غير آمن  
طباعة # factorial (a) #رمز غير آمن  
باستثناء ArithmeticError: #catch من استثناء ArithmeticError  
طباعة ("الرجاء إدخال رقم موجب."). #إظهار الرسالة في حالة الاستثناء  
تم اكتشاف #ArithmeticalException  
باستثناء: #catch #أنواع الاستثناءات الأخرى  
طباعة ("الرجاء إدخال رقم."). #show message #إذا كانت هناك أنواع أخرى  
تم اكتشاف #الاستثناءات
```

- إذا كانت `n` سالبة تماماً، فهذا استثناء من النوع يتم إنشاء الخطأ الحسابي .

04 - إدارة الاستثناءات

معالجة الاستثناء



إنشاء نوع الاستثناء

- في بعض الأحيان يكون تحديد أنواع الاستثناءات الخاصة بنا أكثر عملية وأكثر قابلية للقراءة للقيام بذلك، ما عليك سوى تحديد فئة جديدة ترث من فئة الاستثناء

مثال :

دعونا نحدد دالة ثلاثية الحدود تقوم بحساب وإرجاع جذور ثلاثة الحدود من الدرجة الثانية بالصيغة $c + bx + ax^2$ والتي تولد خطأ عندما لا يكون هناك جذر حقيقي:

مثال :

```
فئة NoRootException(استثناء):
يمر
```

- هذه الفئة ببساطة فارغة حيث أن جسمها يتكون فقط من تعليمات النجاح

من استيراد الرياضيات `sqrt`

تعريف ثلاثي الحدود `(أ, ب, ج)`:

`دلتا = ب * 2 - 4 * أ`

لا يوجد جذر حقيقي

إذا كانت دلتا < 0:

رفع ()#NoRootException رفع استثناء من النوع

#جذر حقيقي مزدوج

`== 0: إذا دلتا`

العودة -`ب / (2 * أ)`

#جذرين حقيقيين بسيطين

`(أ * دلتا) / (-2 * ب) = trqsx1`

`(أ * دلتا) / (-2 * ب) = x2`

العودة `(x1, x2)`

04 - إدارة الاستثناءات

معالجة الاستثناء



استثناء المعلمة

- عندما نستدعي الدالة ثلاثية الحدود ، سنكون قادرین على استخدام تعليمة المحاولة باستثناء اكتشاف هذا الخطأ عند حدوثه
- لنجاول، على سبيل المثال، حساب وعرض الجذور الحقيقية لثلاثية الحدود. للقيام بذلك، نستدعي دالة ثلاثية الحدود عن طريق تمرير المعلمات $a=1$ و $b=2$ حيث أن $c=0$ يتواافق مع

محاولة: #Catch استثناءً باستخدام كتلة المحاولة باستثناء

طباعة (ثلاثي الحدود (1, 0, 2))

NoRootException: باستثناء

طباعة ("لا يوجد جذر حقيقي.")

- في المثال السابق، قد يكون من المفيد معرفة قيمة المميز. في حالة عدم وجود جذر حقيقي. للقيام بذلك تحتاج إلى إضافة متغير مثل `delta`

أداة الوصول إلى فئة `NoRootException`

فئة `NoRootException` (استثناء):

تعريف المُنشئ باستخدام المعلمة `def __init__(self, delta): #`

`self.__delta = delta`

@ملكية

دلتا الدفاع (الذات):

العودة الذاتية.`__atled`

٤٠ إدارة الاستثناءات

استثناء المعلمة

- من الممكن استرداد قيمة المميز في
متلة الاستثناء من الكائن الذي يمثل الاستثناء
لذى حدث

استيراد الرياضيات sqrt

تعريف ثلاثي الحدود (أ، ب، ج):

** 2 - 4 ب = تا

لَا يوجد جذر حقيقى

ا كانت دلتا: <0

#NoRootException(delta) و استثناء المعلمة

حذف حقيقة، مزدوج

دلتا == 0;

152 * 6

$$f(x) = f(x+1) - 1 \in \mathbb{N}$$

112-77000-15

$$x_2 = (-b - \sqrt{\Delta}) / (2)$$

کوڈھ

حاول:

طباعة (ثلاثي الحدود ((1، 0، 2))

e: NoRootException باستثناء

طباعة ("لا يوجد حذر حقيقه ..")

```
#print('Delta =', e.delta)
```



الجزء 4

التعامل مع الوحدات و المكتبات

في هذه الوحدة، سوف تقوم بما يلي:

• تعلم كيفية إنشاء وحدات في بايثون. • إتقان استيراد الوحدات

- تثبيت المكتبات في بايثون
- إنشاء مكتبات في بايثون • استيراد المكتبات في بايثون



ساعة 30



التعامل مع الوحدات



ما ستتعلم في هذا الفصل:

•إنشاء وحدات في بيئون •تصنيف أنواع الوحدات •إتقان

الاستيراد المطلق

•الاستيراد النسبي الرئيسي



05 ساعات



الفصل 1

التعامل مع الوحدات

1. إنشاء الوحدات.

2. استيراد الوحدات.

- 01 التعامل مع الوحدات

إنشاء وحدات



إنشاء وحدات

• الوحدة هي ملف ".py". يحتوي على مجموعة من المتغيرات والوظائف والفنانات التي يمكن استيرادها واستخدامها في البرنامج الرئيسي (أو في وحدات أخرى).

• لإنشاء وحدة، ما عليك سوى برمجة المتغيرات/الوظائف والفنانات التي تتكون منها في ملف يحمل اسم الوحدة، متبوعاً باللحقة ".py".

من برنامج بايثون آخر (آخر)، ما عليك سوى استخدام عنصر الاستيراد الأساسي لتمكن من استخدام هذه المتغيرات/الوظائف/الفنانات.

• الوحدات :

• السماح بفصل التعليمات البرمجية وبالتالي تنظيم التعليمات البرمجية بشكل أفضل

• تعطيم إعادة الاستخدام

• تسهيل مشاركة التعليمات البرمجية

مثال :

وحدة المثال، المساعدة المرتبطة بها

مثال_متغير 3 =

تعريف: function_example()
'''مثال للوظيفة''' إرجاع 0

فئة: class_example:
'''فئة نموذجية'''

إرجاع def __str__(self):
"example_class"

• يوضح المثال أعلاه تعريف الوحدة النمطية التي تحتوي على متغير ودالة وفئة. يمكن استخدام هذه الثلاثة الأخيرة بواسطة أي ملف مهم لهذه الوحدة.

• يمثل اسم الوحدة اسم الملف بدون امتداده.



الفصل 1

التعامل مع الوحدات

1. إنشاء الوحدات

2. استيراد الوحدات

- 01 التعامل مع الوحدات

استيراد الوحدات



• **تعليمات import Module_name** تسمح باستيراد وحدة. يجب أن يتم الاستيراد قبل استخدام مكونات الوحدة. عادة، تتم كافة عمليات الاستيراد في بداية البرنامج. ومع ذلك، يمكن القيام بها في أي جزء من البرنامج.

مثال :

• تسمح التعليمات المتبوعة بالاسم المستعار بتعيين معرف لوحدة نمطية. قد يكون هذا المعرف مختلفاً عن اسم الملف الذي تم تعريف الوحدة فيه.

• تمثل هذه الطريقة طريقة أخرى للقيام بالأشياء المراد استيرادها وحدات.

```
elpmaxe import Module_example #import
example_class = Module_example.example_class() #
طباعة (ج)
example_function()# استدعاء الوظيفة
print(module_example.example_function()) # استدعاء الفئة
```

```
استيراد Module_example كاسم مستعار
ج = الاسم المستعار
()ssalc_elpmaxe.elpmaxe.#
طباعة (ج)
(alias.example_function()) طباعة (ج)
```

لا يُنصح باستخدام الصيغة التالية لأنها تخفي الوحدة التي تأتي منها الوظيفة بالإضافة إلى استيراد كل شيء.

01- التعامل مع الوحدات

استيراد الوحدات



* يستورد

يقدم لك إمكانية استيراد كافة مكونات الوحدة النمطية (الفئات والسمات والمهام). ومع ذلك، فمن الممكن استيراد واحد فقط فئة هذه الوحدة عن طريق الكتابة من `Module_example` استيراد `example_class`

عندما تقوم باستيراد وحدة نمطية، يبحث عنها مترجم بايثون أدلة مختلفة بالترتيب التالي:

الدليل الحالي؛

إذا لم يتم العثور على الوحدة، تقوم Python بالبحث في كل دليل مدرج في متغير `shell PYTHONPATH`:

إذا فشل كل شيء آخر، تتحقق بايثون من المسار الافتراضي (على سبيل المثال `(biL\93nohtyP\بايثون\يندوز)`)

^{*} من الوحدة النمطية `import # elpmaxe` استيراد كافة **الفئات والسمات والوظائف..**
من الوحدة النمطية `elpmaxe`، قم باستيراد `example_class`، و `example_function`.

```
= example_class()
    ج
        طباعة(ج)
    طباعة(function_example())
```

- 01 التعامل مع الوحدات

استيراد الوحدات



شجرة الوحدة

- إذا كان لديك عدة وحدات، فمن الأفضل توزيع ملفاتها في الدليل. يجب أن تظهر هذه في قائمة `sys.path`. • Python يتسمح لغة `import` بتعريف الحزمة. يعتبر الدليل بمثابة حزمة تحتوي على كافة ملفات بايثون.

- قبل إصدار Python 3.3، يجب أن تحتوي الحزمة التي تحتوي على وحدات `__init__.py` النمطية على ملف `__init__.py`

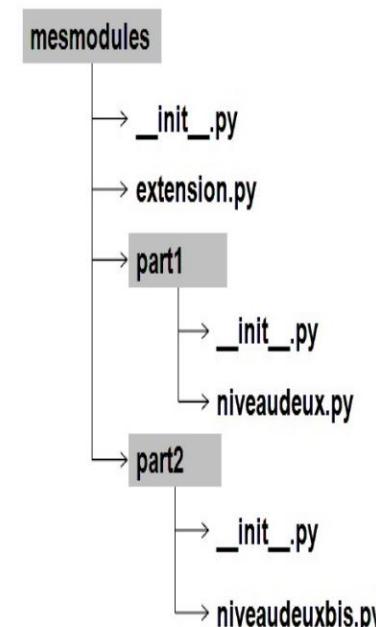
استيراد مطلق

- يحدد الاستيراد المطلق المورد المطلوب استيراده باستخدام مساره الكامل من الدليل الجذر.

مثال :

```
mymodules.extension
استيراد mymodules.part1.leveltwo
استيراد import mesmodules.part2.leveldeuxbis
```

- في المؤدي `mymodules.extension` تقوم `Extension.py` بتنفيذ الملف `__init__.py` وأيضاً جميع محتويات الملف `__init__.py` Python



- 01 التعامل مع الوحدات

استيراد الوحدات



الاستيراد النسبي

• يحدد الاستيراد النسبي المورد المطلوب استيراده بالنسبة إلى الموقع الحالي، أي الموقع الذي يوجد به بيان الاستيراد. • الرمز . يسمح لك باستيراد وحدة نمطية في نفسه

الدليل.

• الرمز .. يسمح لك باستيراد وحدة إلى الدليل الأبوين.

مثال :

يمكن للوظيفة "أ" استخدام الوظيفة "ب" أو "ج" عن طريق استيرادها كما يلي:

التالي :

B.subpackage1. نما استيراد

C.subpackage1.moduleX. نما استيراد

يمكن للوظيفة E استخدام الوظيفة F أو A أو C عن طريق استيرادها بالطريقة التالية:

A..subpackage1.moduleX.. استيراد F من .. نما استيراد

```
package/
    __init__.py      # fonction A
    subpackage1/
        __init__.py  # fonction B
        moduleX.py   # fonction C
    subpackage2/
        __init__.py  # fonction D
        moduleY.py   # fonction E
        moduleA.py   # fonction F
```

01- التعامل مع الوحدات

استيراد الوحدات



بايثونبات

- إضافة مجلد إلى PYTHONPATH في بايثون، يجب عليك الإشارة إلى الأسطر التالية مباشرة في الكود:

استيراد sys.path.insert(0, "E:/exampleImport")

مسار الوحدة النمطية لأخذها إلى المنزل

- تتيح لك وظيفة المسار الخاصة بوحدة sys التحقق من متغير PYTHONPATH

نظام الاستيراد

(sys.path)

- ملاحظة: لا يؤدي الإدخال إلى إضافة المجلد المعنى إلى PYTHONPATH بشكل دائم

```
['', 'C:\\\\Users\\\\DELL\\\\AppData\\\\Local\\\\Programs\\\\Python\\\\Python39\\\\Lib\\\\idlelib', 'E:\\\\exampleImport', 'C:\\\\Users\\\\DELL\\\\AppData\\\\Local\\\\Programs\\\\Python\\\\Python39\\\\python39.zip', 'C:\\\\Users\\\\DELL\\\\AppData\\\\Local\\\\Programs\\\\Python\\\\Python39\\\\DLLs', 'C:\\\\Users\\\\DELL\\\\AppData\\\\Local\\\\Programs\\\\Python\\\\Python39\\\\lib', 'C:\\\\Users\\\\DELL\\\\AppData\\\\Local\\\\Programs\\\\Python\\\\Python39', 'C:\\\\Users\\\\DELL\\\\AppData\\\\Local\\\\Programs\\\\Python\\\\Python39\\\\lib\\\\site-packages']
```



التعامل مع المكتبات



ما سنتعلمه في هذا الفصل:

- تثبيت المكتبات القياسية في بايثون
- التعامل مع مكتبة الرسومات Tinker
- إنشاء مكتبات في بايثون
- استيراد المكتبات في بايثون



الفصل 2

التعامل مع المكتبات

1. تثبيت المكتبات الخارجية (نقطة)
2. إنشاء المكتبات.
3. استيراد المكتبات

02- التعامل مع المكتبات

تثبيت المكتبات الخارجية (نقطة)

مكتبة في بايثون

- في لغة بايثون، المكتبة هي مجموعة من البرامج الوحدات (الفئات (أنواع الكائنات)، والوظائف، الثوابت...) إضافةً إمكانيات موسعة إلى بايثون: الحساب العددي، الرسومات، برمجة الإنترنت، أو الشبكة، وتنسيق النص، وإنشاء المستندات، الخ.

- هناك عدد كبير جدًا منهم، وهو أيضًا من نقاط القوة العظيمة في بايثون.

- يتم تجميع معظمها في Python (حزمة PyPI) المستودع الرسمي لجهة خارجية لغة برمجة بايثون.

المكتبة القياسية

- يحتوي توزيع Python القياسي على مكتبة غنية جدًا من الوحدات التي توسيع قدرات اللغة في العديد من المناطق.

- تغطي المكتبة القياسية مجموعة واسعة من الوظائف، على وجه الخصوص:

- وحدات التاريخ والوقت
- وحدات الواجهة الرسومية
- الوحدات العددية والرياضية
- وحدات نظام الملفات
- وحدات نظام التشغيل
- وحدات لقراءة وكتابة تنسيقات البيانات
- محددة مثل XML وHTML وJSON

- وحدات لاستخدام بروتوكولات الإنترنت مثل HTTP، SMTP، بروتوكول نقل الملفات، الخ.

- وحدات لاستخدام بيانات الوسائط المتعددة مثل بيانات الصوت والفيديو

02- التعامل مع المكتبات

تثبيت المكتبات الخارجية (نقطة)

النقطة (نقطة تثبيت بايثون)

- حزمة تثبيت Python هو مدير الحزم لـ Python
- هي طريقة لتثبيت وإدارة الحزم والتطبيقات الإضافية التي لم يتم توزيعها بعد في إطار العمل من الإصدار القياسي للحزمة.
- تم دمج حزمة Pip في تثبيت Python منذ الإصدار 3.4 لـ Python3 والإصدارات 2.7.9 لـ Python2، ويتم استخدامها في العديد من مشاريع بايثون.
- عن طريق تشغيل الأمر أعلاه من الممكن التحقق مما إذا كانت النقطة متاحة أم لا

النقطة - الإصدار

• نتيجة التنفيذ:

```
pip 21.2.4 from C:\Users\DELL\AppData\Local\ Packages\PythonSoftwareFoundation.Python.3.9_qbz5n2kfra8p0\LocalCache\local-packages\Python39\site-packages\pip (python 3.9)
```

- يعرض الإخراج إصدار النقطة الموجودة في جهازك، بالإضافة إلى موقع إصدار Python الخاص بك
- إذا كنت تستخدم إصداراً أقدم من لغة Python لا يتضمن النقطة pip، فستحتاج إلى تثبيته

02- التعامل مع المكتبات تثبيت المكتبات الخارجية (نقطة)

تثبيت بيب

الطريقة الأولى:

• تحميل [مجلد get-pip.py](https://bootstrap.pypa.io/get-pip.py) على جهاز الكمبيوتر الخاص بك.

• افتح موجه الأوامر وانتقل إلى المجلد الذي يحتوي على برنامج التثبيت get-pip.py.

• قم بتشغيل الأمر التالي:

`py get-pip.py`

الطريقة الثانية:

إليك أمر تحميله باستخدام أداة wget لـ Windows
`wget https://bootstrap.pypa.io/get-pip.py`

• للتعرف على الأوامر التي يدعمها pip واستخدم الأمر التالي:

مساعدة النقطة

نتيجة التنفيذ:

```
Usage:
  C:\Users\DELL\AppData\Local\Microsoft\WindowsApps\PythonSoftwareFoundation.Python.3.9_qbz5n2kfra8p0\python.exe -m pip <command> [options]

Commands:
  install           Install packages.
  download          Download packages.
  uninstall         Uninstall packages.
  freeze            Output installed packages in requirements format.
  list               List installed packages.
  show               Show information about installed packages.
  check              Verify installed packages have compatible dependencies.
  config             Manage local and global configuration.
  search             Search for packages.
  cache              Inspect and manage pip's wheel cache.
  index              Inspect information available from package indexes.
  wheel              Build wheels from source distributions.
  hash               Compute hashes of package archives.
  completion         A helper command used for command completion.
  debug              Print additional output for debugging.
  help               Show help for commands.

General Options:
  -h, --help          Show help.
  --isolated         Run pip in an isolated mode, ignoring environment variables and user configuration.
  -v, --verbose       Give more output. Option is additive, and can be used up to 3 times.
  -q, --quiet         Give less output. Option is additive, and can be used up to 3 times (corresponding to
                     silent, quiet, info, warning, and error logging levels).
  --log <path>        Path to a verbose appending log.
  --no-input         Disable prompting for input.
  --proxy <proxy>     Proxy URL or IP:port or proxy server:port.
  --retries <retries> Maximum number of retries each connection should attempt (default 5 times).
  --timeout <sec>    Set the socket timeout (default 15 seconds).
  --exists-action <action> Define action when a path already exists: (s)witch, (i)gnore, (w)ipe, (b)ackup,
                    (a)bort.
  --trusted-host <hostname> Mark this host or host:port pair as trusted, even though it does not have valid or any
                    SSL certificate.
  --cert <path>       Path to PEM-encoded CA certificate bundle. If provided, overrides the default. See 'SSL
                    Certificate Verification' in pip documentation for more information.
```

02- التعامل مع المكتبات

تثبيت المكتبات الخارجية (نقطة)

تثبيت المكتبات

- توفر النقطة أمر تثبيت لتنصيب الحزم/المكتبات

```
Sampleproject -m pip
```

```
Uninstalling sampleproject:
[...]
Proceed (y/n)? y
Successfully uninstalled sampleproject
```

- من الممكن أيضًا تحديد الحد الأدنى للإصدار مباشرةً من سطر الأوامر
 - استخدم أحرف المقارنة مثل <أو > أو أحرف خاصة أخرى يتم تفسيرها بواسطة الصدفة، ويجب وضع اسم الحزمة وإصدارها بين علامتي اقتباس

```
pip install "SomePackage>=1.0.4" # إصدار محدد
                           # -m pip install SomePackage==1.0.4 # الإصدار الأدنى
```

- عادةً، إذا تم بالفعل تثبيت مكتبة مناسبة، فلن يكون لتنسيقها مرة أخرى أي تأثير

02- التعامل مع المكتبات تثبيت المكتبات الخارجية (نقطة)



تثبيت المكتبات

• يجب طلب تحديث المكتبات الموجودة بشكل صريح:

```
SomePackage ترقية -m pip install -
```

• إلغاء تثبيت مكتبة، يجب عليك استخدام الأمر التالي

```
Sampleproject إلغاء تثبيت -m pip
```

02- التعامل مع المكتبات تثبيت المكتبات الخارجية (نقطة)

مكتبات الرسومات

• تُستخدم مكتبة matplotlib ومكتبها الفرعية pyplot بشكل أساسى لعرض الرسومات. استخدامه مشابه جداً لاستخدام Matlab.

للتثبيت، يجب عليك كتابة الأمر:

matplotlibpy -m pip تثبيت

```
PS C:\Users\DELL> pip install matplotlib
Collecting matplotlib
  Downloading matplotlib-3.4.3-cp39-cp39-win_amd64.whl (7.1 MB)
    |██████████| 7.1 MB 2.2 MB/s
Collecting numpy>=1.16
  Using cached numpy-1.21.3-cp39-cp39-win_amd64.whl (14.0 MB)
Collecting cycler>=0.10
  Downloading cycler-0.10.0-py2.py3-none-any.whl (6.5 kB)
Collecting kiwisolver>=1.0.1
  Downloading kiwisolver-1.3.2-cp39-cp39-win_amd64.whl (52 kB)
    |██████████| 52 kB 84 kB/s
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\dell\appdata\local\packages\pythonsoftwarefoundation.python.3.9_qbz5n2kfra8p0\localcache\local-packages\python39\site-packages (from matplotlib) (2.4.7)
Collecting pillow>=6.2.0
  Downloading Pillow-8.4.0-cp39-cp39-win_amd64.whl (3.2 MB)
    |██████████| 3.2 MB 1.7 MB/s
Collecting python-dateutil>=2.7
  Downloading python_dateutil-2.8.2-py2.py3-none-any.whl (247 kB)
    |██████████| 247 kB 2.2 MB/s
Requirement already satisfied: six in c:\users\dell\appdata\local\packages\pythonsoftwarefoundation.python.3.9_qbz5n2kfra8p0\localcache\local-packages\python39\site-packages (from cycler>=0.10->matplotlib) (1.16.0)
```

02- التعامل مع المكتبات

تثبيت المكتبات الخارجية (نقطة)



مكتبات الرسومات

- هي وحدة مجانية تسمح لك بإنشاء واجهات رسومية في Python.
- لتنصيب PyQt، يجب عليك كتابة الأمر:

```
pyqt5 تثبيت -m pip
```

```
PS C:\Users\DELL> pip install pyqt5
Collecting pyqt5
  Downloading PyQt5-5.15.5-cp36-abi3-win_amd64.whl (6.7 MB)
    |██████████| 6.7 MB 2.2 MB/s
Collecting PyQt5-sip<13,>=12.8
  Downloading PyQt5_sip-12.9.0-cp39-cp39-win_amd64.whl (63 kB)
    |██████████| 63 kB 280 kB/s
Collecting PyQt5-Qt5>=5.15.2
  Downloading PyQt5_Qt5-5.15.2-py3-none-win_amd64.whl (50.1 MB)
    |██████████| 50.1 MB 1.1 MB/s
Installing collected packages: PyQt5-sip, PyQt5-Qt5, pyqt5
Successfully installed PyQt5-Qt5-5.15.2 PyQt5-sip-12.9.0 pyqt5-5.15.5
WARNING: You are using pip version 21.2.4; however, version 21.3.1 is available.
You should consider upgrading via the 'C:\Users\DELL\AppData\Local\Microsoft\WindowsApps\PythonSoftwareFoundation.Python.3.9_qbz5n2kfra8p0\python.exe -m pip install --upgrade pip' command.
```

02- التعامل مع المكتبات تثبيت المكتبات الخارجية (نقطة)

مكتبات الرسومات

- عبارة عن وحدة مدمجة في Python لتطوير التطبيقات الرسومية.
- تعتمد هذه الوحدة على مكتبة رسومات Tk/Tk.
- لتنصيب Tk، يجب عليك كتابة الأمر:

```
py -m pip install tk
```

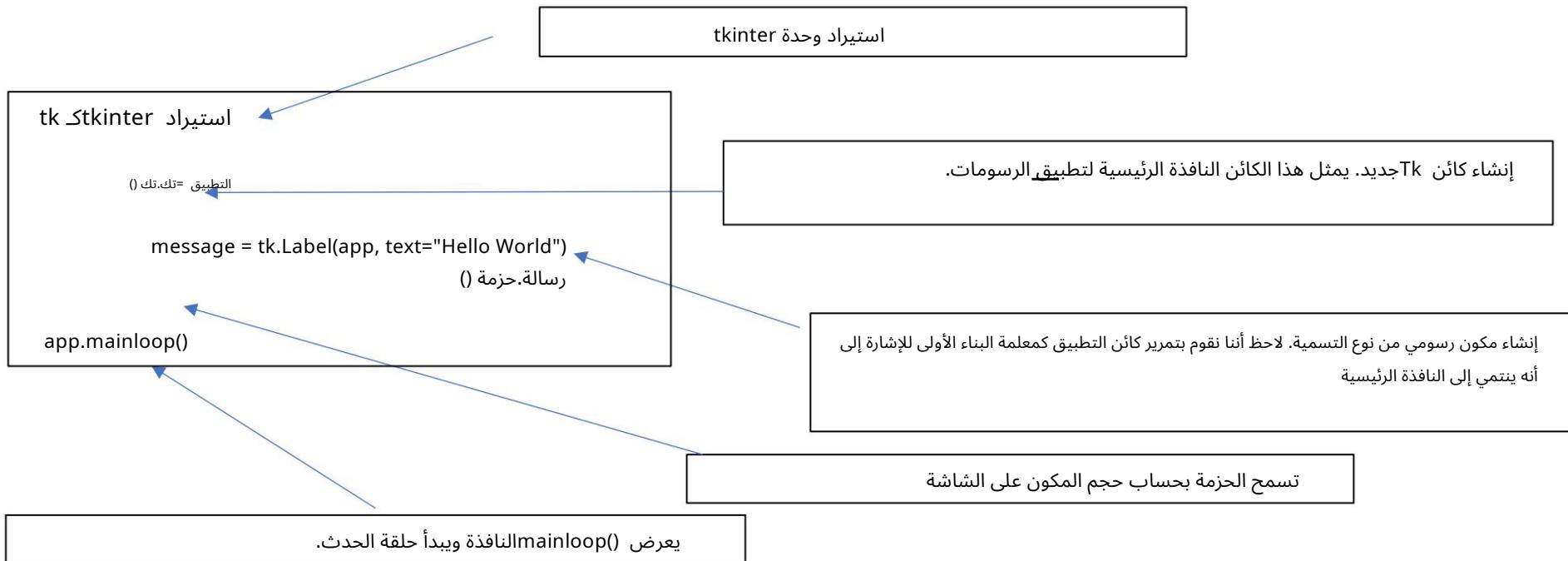
```
PS C:\Users\DELL> pip install tk
Collecting tk
  Downloading tk-0.1.0-py3-none-any.whl (3.9 kB)
Installing collected packages: tk
Successfully installed tk-0.1.0
WARNING: You are using pip version 21.2.4; however, version 21.3.1 is available.
You should consider upgrading via the 'C:\Users\DELL\AppData\Local\Microsoft\WindowsApps\PythonSoftwareFoundation.Python.3.9_qbz5n2kfra8p0\python.exe -m pip install --upgrade pip' command.
```

02- التعامل مع المكتبات

ثبيت المكتبات الخارجية (نقطة)

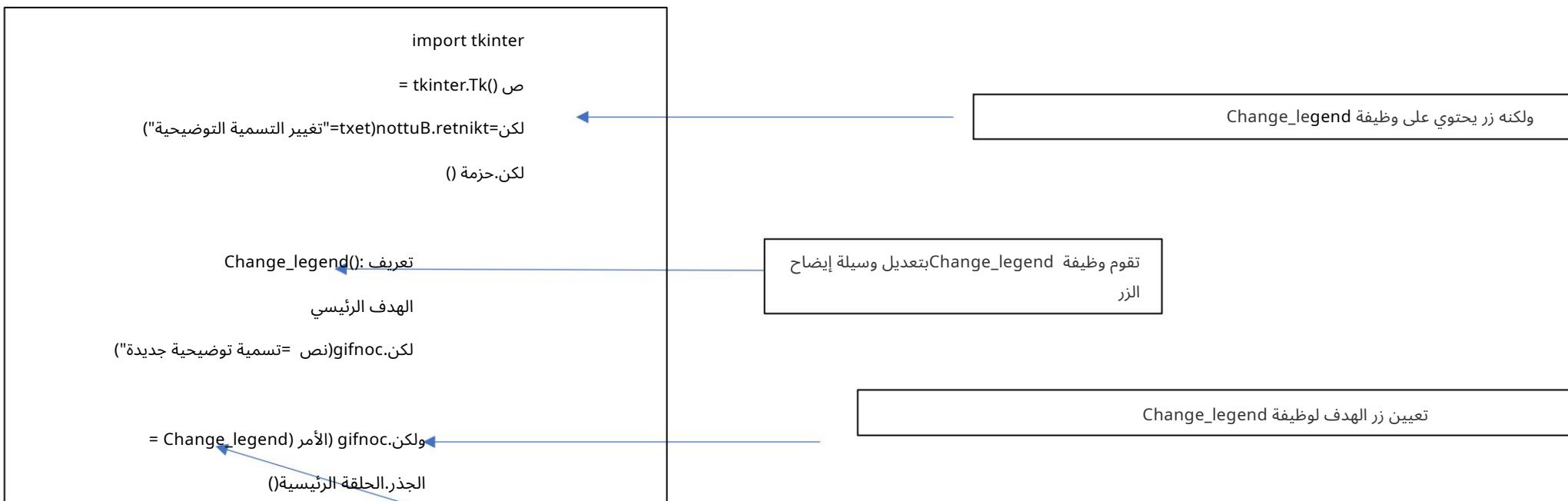
تكتنر

• يوضح البرنامج أدناه المبدأ الأساسي لـ `tkinter`



02- التعامل مع المكتبات تثبيت المكتبات الخارجية (نقطة)

Tkinter - تشغيل وظيفة عند الضغط على زر



يقوم الأمر بتشغيل البرنامج ويعرض النافذة

02- التعامل مع المكتبات

تثبيت المكتبات الخارجية (نقطة)

ربط حدث بـ Tkinter -

يمكن تنفيذ وظيفة عند تحريك الماوس والكتابة

على مفتاح لوحة المفاتيح أو أي حدث تم التقاطه بواسطة الواجهة الرسومية.

فئة الحدث تحدد الأحداث. ويوضح الجدول التالي

الأحداث الرئيسية المتعلقة بحركات الماوس وضغط زر.

خزان

يحتوي Char على رمز المفتاح الذي تم الضغط عليه. لا يأخذ في الاعتبار مفاتيح كتم الصوت (الرجوع والحذف والتحول والبديل LRTC) .

com.keysym

يحتوي على رمز المفتاح الذي تم الضغط عليه مهما كان المفتاح.

رقم

تحتوي هذه السمة على معرف الكائن الذي استقبل الحدث.

س. ص

الإحداثيات النسبية للماوس بالنسبة إلى الزاوية اليسرى العليا للكائن الذي استقبل الحدث.

x_root, y_root

إحداثيات الماوس المطلقة بالنسبة إلى الزاوية اليسرى العليا من الشاشة.

القطعة

معرف يسمح بالوصول إلى الكائن الذي استقبل الحدث.

02- التعامل مع المكتبات تثبيت المكتبات الخارجية (نقطة)

Tkinter - حدث بـ Tkinter

- يبدأ أسلوب الربط في تنفيذ وظيفة عندما يُعترض حدث من قبل كائن.
- الدالة المنفذة لها كمعلمات واحدة كائن من النوع حدث. بناء الجملة الخاص به هو كما يلي:

```
w.bind(EV, وظيفة)
```

يمثل `w` معرف الكائن الذي سيتم اعتراضه الحدث إيف. ويوضح الجدول التالي القيم التي يمكن أن تأخذ `ev`.

• تمثل الدالة الدالة التي سيتم استدعاؤها عند وصول الحدث. لديها واحدة فقط معلومة نوع الحدث.

<المفتاح>

يلتقط أي مفتاح لوحة المفاتيح عند الضغط عليه.

<زر->i

التقط النقر بالماوس. يجب أن يتم استبدالي بـ 1,2,3.

<ButtonRelease-i>

يلتقط الإفراج عن النقر بالماوس. يجب أن يتم استبدالي بـ 1,2,3.

<زر مزدوج->i

يلتقط النقر المزدوج للماوس. يجب أن يتم استبدالي بـ 1,2,3.

<الحركة>

يلتقط حركة مؤشر الماوس.

<أدخل>

عندما يدخل مؤشر الماوس إلى المنطقة الرسمية للكائن، يلتقط <Enter> الحدث المرتبط.

<مغادرة>

يلتقط الحدث المرتبط عندما يغادر مؤشر الماوس منطقته الجغرافية.

02- التعامل مع المكتبات

تثبيت المكتبات الخارجية (نقطة)



• المثال أدناه يوضح استخدام الأسلوب bind. يجب أن يلتقط زر النافذة أي حركة للمؤشر عندما يكون فوق منطقته الجغرافية (الضغط على مفتاح على لوحة المفاتيح أو النقر على زر بالماوس).

```

استيراد جذر (=tkinter.Button = tkinter.Tk() = صن)
اضغط
على أي مفتاح() pack()

def Display_pressed_key (evt): print("-----")
    = evt.x_root, evt.y_root) print("evt.widget = ", evt.widget)
= evt.num) print("evt.x,evt.y = ", evt.x, "", evt.y) print("evt.x_root,evt.y_root
print("evt.char = ", evt .char) print("evt.keysym = ", evt.keysym) print("evt.num

b.bind ("<Motion>", Display_key_press) b.focus_set ()
Display_key_press) b.bind ("<Button-1>", Display_key_press)
b.bind ("<Key>",

الجذر.الحلقة الرئيسية()

```



focus_set هو الكائن الذي يلتقط الأحداث المتعلقة بلوحة المفاتيح

02- التعامل مع المكتبات تثبيت المكتبات الخارجية (نقطة)



المكونات الرسومية (الحاجيات)

النافذة الرئيسية

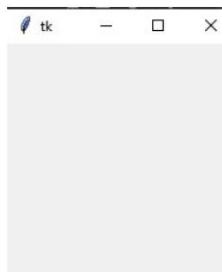
صندوق الرسائل

* يكون النافذة الرسومية مفتوحة على كل جهاز لـ Tkinter، وهي تظهر هنا كنافذة فرعية وتحافظ على الأسلوب الشمالي من الكود.

التي سيتم عرضها كمعلومة ثانية.

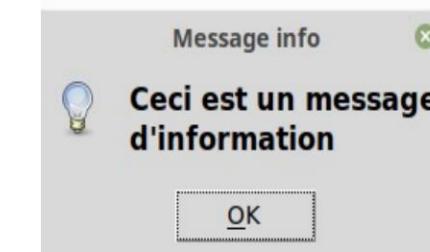
رسالة معلومات

```
root = tkinter.Tk () #  
root.mainloop
```



من صندوق رسائل الاستيراد

messagebox.showinfo("معلومات الرسالة", "هذه رسالة معلومات")



02- التعامل مع المكتبات تثبيت المكتبات الخارجية (نقطة)



المكونات الرسومية (الحاجيات)

صندوق الرسائل

رسالة تحذير

من صندوق رسائل الاستيراد tkinter

رسالة تحذير، "هذه رسالة تحذير") messagebox.showwarning("

رسالة خطأ

من صندوق رسائل الاستيراد tkinter

رسالة خطأ، "هذه رسالة خطأ") messagebox.showerror ("

سؤال مع الإجابة موافق / إلغاء

من صندوق رسائل الاستيراد tkinter

استجابة ، "هل تريد المتابعة؟" = messagebox.askokcancel ("

نعم / لا سؤال الإجابة

من صندوق رسائل الاستيراد tkinter

استجابة ، "هل تريد المتابعة؟" = messagebox.askyesnocancel ("

Message d'avertissement

Ceci est un message d'avertissement

OK

Message d'erreur

Ceci est un message d'erreur

OK

Question

Voulez-vous continuer ?

OK

Annuler

Question

Voulez-vous continuer ?

Oui

Non

02- التعامل مع المكتبات تثبيت المكتبات الخارجية (نقطة)

المكونات الرسومية (ال حاجيات)

منطقة النص

• مربع النص عبارة عن تسمية أو وسيلة لإيضاح توضع عادةً بجوار مربع الإدخال وتحذر المستخدم بما يجب كتابته، لإنشاء مربع نص، يجب عليك كتابة هذا السطر من التعليمات البرمجية:

```
(النص = "مربع النص")text_area = tkinter.Label
```

• من أجل تغيير النص الموجود في مربع النص، يجب عليك استدعاء طريقة التكوين على النحو التالي:

```
= text_zone = tkinter.Label (text =
    "النص الأول") # #للتغيير النص
...
    "النص الثاني")text_zone.config (text
```

يمكن أن يكون مربع النص في حالة "معطل".

• للقيام بذلك، يجب عليك إضافة السطر التالي من التعليمات البرمجية:

```
= tkinter.DISABLED)text_zone.config
```

• للعودة إلى الحالة الطبيعية، يجب عليك كتابة هذا السطر من التعليمات البرمجية:

```
= tkinter.NORMAL)text_zone.config
```

يمكن استخدام هذين الخيارين على أي نوع من كائنات واجهة المستخدم الرسومية

02- التعامل مع المكتبات تثبيت المكتبات الخارجية (نقطة)

المكونات الرسومية (الحاجيات)

زر

- الزر هو مكون رسومي يربط وظيفة ما بنقرة بالماوس. لكي تتمكن من إنشاء زر، عليك كتابة الكود التالي:

```
زر = tkinter.Button(text = "مربع النص")
```

- لكي تتمكن من تعديل النص الذي يظهر أعلى الزر، يجب عليك استدعاء طريقة التكوين كما يلي:

```
= Button = tkinter.Button(text = "النص الأول") # للتغيير النص
...                                             "النص الثاني"
Button.config (text
```

02- التعامل مع المكتبات

تثبيت المكتبات الخارجية (نقطة)



المكونات الرسمية (الحاجيات)

حقل الإدخال

• منطقة الإدخال عبارة عن مكون رسومي يوفر للمستخدم إمكانية إدخال البيانات.

• من أجل إنشاء منطقة دخول، يجب عليك الاتصال بفئة الإدخال عن طريق كتابة السطر التالي من التعليمات البرمجية:

```
= tkinter.Entry()
```

• لتعديل محتوى منطقة الإدخال، يجب عليك استخدام طريقة الإدراج.

```
#المعلمة الأولى هي الموضع # حيث سيتم إدراج النص
input.insert(pos, "content")
```

• من أجل استرداد محتوى منطقة الإدخال، يجب عليك استخدام طريقة :
get

```
المحتوى = الإدخال.get()
```

• لحذف محتوى منطقة الإدخال، يجب عليك استخدام طريقة الحذف.

```
#حذف النص بين المواقع pos1, pos2
Grab.delete (pos1, pos2)
```

02- التعامل مع المكتبات تثبيت المكتبات الخارجية (نقطة)

المكونات الرسومية (الحاجيات)

خانة الاختيار

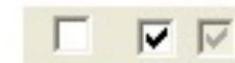
- من أجل إنشاء خانة اختيار، يجب عليك استدعاء فئة Checkbutton.

```
#ينشئ كائناً عدداً صحيحاً لاسترداد قيمة مربع الاختيار،
```

```
0 #لغير المحدد، 1 للمحدد
```

```
v = tkinter.IntVar()
```

```
= tkinter.Checkbutton(متغير v)
```



- من أجل التعرف على المربع الذي تم تحديده، يجب عليك تنفيذ التعليمات التالية:

```
#v.get() يساوي 1 إذا تم تحديد المربع، و0 إذا كان خلاف ذلك
```

- نتيجة لك الإرشادات الواردة أدناه تحديد المربع وإلغاء تحديده:

```
case.deselect() #تحقق من case.select() #  
إلغاء التحديد
```

- لربط نص بمربع، يجب عليك استدعاء طريقة التكوين:

```
حالـة.gifnoc(نص = "مربع الاختيار")
```



02- التعامل مع المكتبات تثبيت المكتبات الخارجية (نقطة)

زر الراديو

* من أجل إنشاء زر اختيار، يجب عليك الاتصال بفئة `Radiobutton`.

* تعمل بشكل مشابه لمربعات الاختيار فيما عدا أنها تعمل في مجموعات، لإنشاء مجموعة من ثلاثة صناديق جولات، فقط اكتب البرنامج التالي:

```
# يقوم بإنشاء كائن عدد صحيح مشترك لاسترداد رقم زر الاختيار المنشط
v = tkinter.IntVar()

الحالة 1 = tkinter.Radiobutton(v, value=10)
case2 = tkinter.Radiobutton(v, value=20)
case3 = tkinter.Radiobutton(v, value=30)
```

v هو متغير تستخدمه أزرار الاختيار الثلاثة. تسمح لك القيمة بربط قيمة بكل راديو.

- إذا كانت v == 10، فسيتم تحديد زر الاختيار الأول.

- إذا كانت v == 20، فسيكون زر الاختيار الثاني.

* تقوم طريقة `Get` بإرجاع رقم زر الاختيار الذي تم تحديده:

```
v.get() # يُرجع رقم زر الاختيار المحدد (هنا، 10 أو 20 أو 30)
```

02- التعامل مع المكتبات

تثبيت المكتبات الخارجية (نقطة)

زر الراديو

• يتيح البرنامج التالي اختيار أحد أزرار الاختيار:

```
v = tkinter.IntVar()
الحالة = 10 #متغير ، v = القيمة
case1 = tkinter.Radiobutton(v, text="الحال", value=10)
case2 = tkinter.Radiobutton(v, text="الحال", value=20)
case3 = tkinter.Radiobutton(v, text="الحال", value=30)

#قم زر الاختيار للتحقق #لهذا المثال، 10 أو 20 أو 30
v.set(number)
```

- premier bouton
- second bouton
- troisième bouton

• تسمح طريقة التكوين بإدراج نص في زر الاختيار:

```
case1.config(text="الحال")
case2.config(text="الحال")
case3.config(text="الحال")
```

02- التعامل مع المكتبات

تثبيت المكتبات الخارجية (نقطة)

قائمة

- القائمة عبارة عن مكون رسومي يحتوي على مجموعة من العناصر القابلة للتحديد. من أجل إنشاء قائمة، يجب عليك الاتصال على

فترة: ListBox

```
= tkinter.ListBox()
```

• تتيح لك التعليمات التالية تعديل أبعاد القائمة:

```
#تعديل أبعاد القائمة
```

```
#العرض <-->العرض
```

```
#الارتفاع <-->الارتفاع بالسطور
```

```
gifnoc.il(العرض=01, الارتفاع=5)
```



• تتيح لك طريقة الإدراج إدراج عنصر في القائمة المنسدلة:

```
"pos عدد صحيح، "end" أو "li.insert(pos, "السطر الأول") لـtkinter.insert هذه الكلمة في النهاية.
```

02- التعامل مع المكتبات

تثبيت المكتبات الخارجية (نقطة)

قائمة

*تيح لك طريقة `Select_set` إمكانية تحديد عنصر من القائمة.

```
= 0 1
      pos1, pos2 =
      ali.select_set(pos1, pos2 =
# يختار جميع العناصر بين المؤشرات pos1 و
# متضمن أو فقط مؤشر pos1 إذا == pos2 لا شيء
```

*تقوم طريقة `cusection` بإرجاع معرفات العناصر المحددة.

```
= li.curselection()
```

*تيح لك أسلوب `get` استرداد عنصر من القائمة. تقوم طريقة `get` بإرجاع عدد العناصر.

```
(0, li.size()): print(li.get
        بالنسبة لـ i في النطاق ((i))
```

02- التعامل مع المكتبات تثبيت المكتبات الخارجية (نقطة)

جدول (عرض الشجرة)

* تتيح لك أداة Treeview عرض البيانات في صفوف وأعمدة. يجب أن يتبع إنشاء TreeView الخطوات التالية

الخطوة 1: إنشاء TreeView • Columns=ac تحدد أسماء الأعمدة، show='headings' يعني أن الصف الأول هو رأس الجدول، والارتفاع=7 يعني أن

الجدول 7 أسطر

```
tv=ttk.Treeview(root,columns=ac,show='headings',height=7)
```

الخطوة 2: تحديد خصائص العمود

[i] هو اسم العمود، والعرض 70 = هو حجم العمود، والمرساة 'e' = هو نوع المحاذاة

```
tv.column(ac[i],width=70,anchor='e')
```

الخطوة 3: تعريف الصف الأول من الجدول • النص = المنطقة [i] يعرض محتوى كل عمود في الصف الأول

```
tv.heading(ac[i],text=area[i])
```

الخطوة 4: ملء الجدول Values=sales_data[i] • تحدد محتوى كل صفات في الجدول

```
tv.insert("",'end',values=sales_data[i])
```

02- التعامل مع المكتبات تثبيت المكتبات الخارجية (نقطة)

جدول (عرض الشجرة)
مثال :

استيراد Tkinter
ttk استيراد Tkinter

```
root = tk.Tk()
root.geometry('320x240')
tk.Label(root, text='Tkinter Treeview widget').pack()
Area = ('#', 'أوروبا الشمالية', 'أوروبا الشرقية', 'أوروبا الجنوبية', 'أوروبا الغربية', 'الولايات المتحدة الأمريكية', 'كندا', 'أستراليا')
```

```
(78348, 41341, 314, 41341, 4114, 34414, 41874)] = atad_selasac = ('all', 'n', 'e', 's', 'ne', 'nw', 'sw', 'c')
```

```
"48349", "94734", "4743", "43434", "43844", "43843", "88844"), ("مستحضرات التجميل", "ملابس", "متفرقات", '39933', '3903', '4344')] , '4986', '39934', '43934', '9394')
```

```
for i in range(8):
    tv.column(ac[i], width=70, anchor='e')
    tv.heading(ac[i], text=area[i])
    tv.pack()
```

```
tv = ttk.Treeview(root, columns=ac, show='headings', height=7)
```

```
root.mainloop()
tv.insert("", 'end', values=sales_data[i])
```



	Northe	Eastern	Southern	Western	USA	Canada	Australia
Electronics	47814	41443	4114	14314	413	14314	84387
Cosmetics	48349	94734	4743	43434	43844	43843	88844
Clothing	14841	49761	147471	49094	57844	48499	49494
Misc	4939	43934	43993	6894	39933	3903	4344

02- التعامل مع المكتبات

تثبيت المكتبات الخارجية (نقطة)



تحديد موضع الكائنات في النافذة

- تسمح لك أساليب الحزمة والشبكة والمكان بوضع الكائنات الرسومية في النافذة.
- الحزمة والشبكة: توفر إمكانية تحديد موضع الأشياء دون القلق بشأن أبعادها ومواقعها.

طريقة الحزمة

- حزم الكائنات الرسومية بطريقة متتالية مثل يظهر المثال التالي:

تنكيف النافذة مع هذين الآخرين تلقائيا. • المكان: وضع الأشياء في النافذة وفقاً لـ إحداثيات محددة.

```
ل = صن=tkinter.Label("السطر الأول")
    ل.باك()
صورة()= tkinter.Entry()
    s.pack()
نص=e=tkinter.Label("السطر الثاني")
    حزمة إلكترونية()
```



• يتيح لك الخيار الجانبي تغيير ترتيب الكائنات عن طريق وضعها جنباً إلى جنب على اليمين.

```
ل = صن=tkinter.Label("السطر الأول")
    = tkinter.RIGHT()l.kcap.= tkinter.Entry()
صورة()= tkinter.RIGHT()l.kcap.s
نص=e=tkinter.Label("السطر الثاني")
    = tkinter.RIGHT()e.pack()
```

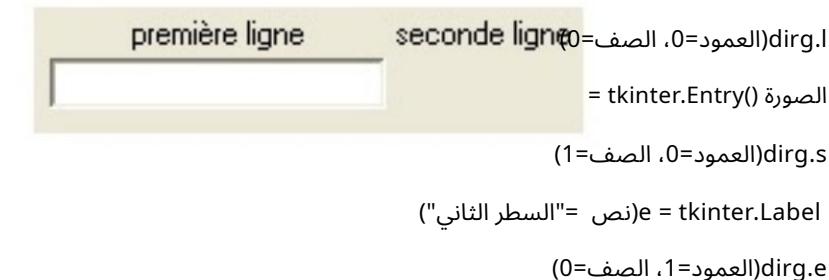
02- التعامل مع المكتبات

تثبيت المكتبات الخارجية (نقطة)



طريقة الشبكة

*توصي طريقة الشبكة باستخدام شبكة يمكن أن تحتوي فيها كل مربع على كائن رسومي. المثال أدناه يضع ثلاثة الكائنات الموجودة في مربعات الإحداثيات (0,0) و (0,1) و (1,0).



*طريقة الشبكة لديها عدة خيارات مثل:

*العمود: العمود الذي سيتم وضع الكائن فيه. `columnspan`: عدد الأعمدة التي يجب أن يشغلها

الكائن.

*الصف: الخط الذي سيتم وضع الكائن فيه.

*عدد الأسطر التي يجب أن يشغلها الكائن. `rowspan`:

02- التعامل مع المكتبات تثبيت المكتبات الخارجية (نقطة)



طريقة المكان

* تتيح لك طريقة المكان وضع كائن رسومي وفقاً لإحداثيات محددة جداً:
`y=50)=txet)lebaL.retnfktl =`

`I.place (x=10,`



الفصل 2

التعامل مع المكتبات



1. تثبيت المكتبات الخارجية (نقطة)
2. إنشاء المكتبات.
3. استيراد المكتبات

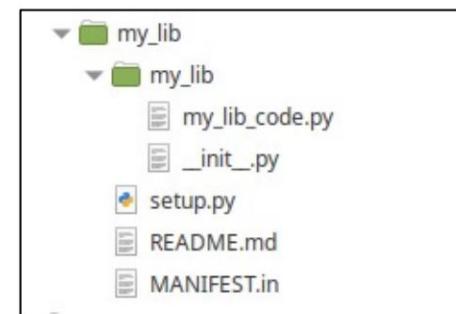
02- التعامل مع المكتبات

إنشاء المكتبات



- في بعض الأحيان لا تكون المكتبات القياسية كافية ونأمل في إنشاء مكتبة تحتوي على وظائف محددة.

- لإنشاء مكتبة، يجب أن يحترم المشروع البنية التالية:



- المجلد `my_lib` الذي يحتوي على جميع التعليمات البرمجية الخاصة بمكتبتنا، بالإضافة إلى ملف `__init__.py` الذي يسمح لباليثون بمراعاة مجلد يحتوي على حزم.

- ملف `README.md` الذي يحتوي على الوصف الكامل للمكتبة • ملف `MANIFEST.in` الذي يسرد جميع الملفات غير

الموجودة في المكتبة Python

- ملف `setup.py` الذي يحتوي على وظيفة الإعداد التي تسمح بتثبيت المكتبة على النظام

02- التعامل مع المكتبات

إنشاء المكتبات



setup.py

• يحتوي هذا الملف على طريقة الإعداد التي تسمح بتنصيب المكتبة على النظام.

• يمكن أن تستغرق وظيفة الإعداد حوالي ثلاثة وسبعين، ولكن هنا في المثال تلك التي سيتم استخدامها في أغلب الأحيان.

• المعلمات الأولى لهذه الطريقة هي بشكل أساسى معلمات وصفية للمكتبة، ومؤلفها، وترخيصها، ونسختها، وما إلى ذلك.

```
from setuptools import setup

setup(
    name='my_lib',
    version='1.0.0',
    author='Author Name',
    author_email='author.name@mail.com',
    description='Courté description de la librairie',
    license='Other/Proprietary License',
    keywords='lib',
    url='Lien vers la page officielle de la librairie',
    packages=[
        'my_lib'
    ],
    long_description=open('README.md').read(),
    classifiers=[
        'Development Status :: 4 - Beta',
        'Intended Audience :: Developers',
        'License :: Other/Proprietary License',
        'Natural Language :: French',
        'Operating System :: OS Independent',
        'Programming Language :: Python :: 3.7',
        'Topic :: Software Development',
        'Topic :: Software Development :: Libraries :: Python Modules',
        "Topic :: Utilities"
    ],
    install_requires=[
        'PyMongo==3.7.2'
    ],
    include_package_data=True
)
```

02- التعامل مع المكتبات

إنشاء المكتبات

- فيما يتعلق بالمعلمات الأخرى:

- الحزمة: تحتوي على قائمة بجميع حزم المكتبة التي سيتم إدراجها في التوزيعة

- التصنيف: البيانات الوصفية التي تسمح للروبوتات بتصنيف المكتبة بشكل صحيح

- هذا الجزء يحتوي على كافة التبعيات الازمة لحسن سير العمل في مكتبتك

- يتبع الدعم لملف MANIFEST.in

- بمجرد تحديد كل هذه العناصر، المكتبة يمكن تثبيتها على النظام باستخدام الأمر التالي (نفذ الأمر في ملف دليل المشروع):

```
py setup.py install
```

02- التعامل مع المكتبات

إنشاء المكتبات

مثال :

إنشاء المكتبة

Nom	Modifié le	Type	Taille
Nouvelle_Bib	2021-10-25 12:42	Dossier de fichiers	
setup	2021-10-25 12:43	Python File	1 Ko

Nom	Modifié le	Type	Taille
__init__	2021-10-23 7:59	Python File	1 Ko
bonjour	2021-10-25 11:44	Python File	1 Ko

```
فئة مرحبا: self.name
def Displays(self):
    self.name=name
def __init__(self,name):
```

من setuptools استيراد الإعداد

```
"الإعداد (اسم "Nouvelle_Bib", "المؤلف "0.0.1", "الإصدار " = "الإصدار ", " المؤلف =
Author_email = "authorX@yahoo.fr", description =
وصفت قصيرة للمكتبة", url = "url" رابط إلى الصفحة
الرسمية "للمكتبة", License, packages=[ 'Nouvelle_Bib', ], "لغة البرمجة :: ]=sreifissalcLicense='Other/Proprietary
MIT", "الترخيص :: معتمد من OSI :: Python 3", نظام التشغيل: نظام تشغيل مستقل", ],
install_requires[], include_package_data=True
```

)

02- التعامل مع المكتبات

إنشاء المكتبات

تثبيت المكتبة

*تنفيذ الأمر:

`py setup.py install`

```
PS E:\nouvelle_Bib> py setup.py install
C:\Users\DELL\AppData\Local\Programs\Python\Python39\lib\distutils\dist.py:259: UserWarning: 'licence' distribution option is deprecated; use 'license'
  warnings.warn(msg)
running install
running bdist_egg
running egg_info
creating Nouvelle_Bib.egg-info
writing Nouvelle_Bib.egg-info\PKG-INFO
writing dependency_links to Nouvelle_Bib.egg-info\dependency_links.txt
writing top-level names to Nouvelle_Bib.egg-info\top_level.txt
writing manifest file 'Nouvelle_Bib.egg-info\SOURCES.txt'
reading manifest file 'Nouvelle_Bib.egg-info\SOURCES.txt'
writing manifest file 'Nouvelle_Bib.egg-info\SOURCES.txt'
installing library code to build\bdist.win-amd64\egg
running install_lib
running build_py
creating build
creating build\lib
creating build\lib\Nouvelle_Bib
copying Nouvelle_Bib\bonjour.py -> build\lib\Nouvelle_Bib
copying Nouvelle_Bib\__init__.py -> build\lib\Nouvelle_Bib
creating build\bdist.win-amd64
creating build\bdist.win-amd64\egg
creating build\bdist.win-amd64\egg\Nouvelle_Bib
copying build\lib\Nouvelle_Bib\bonjour.py -> build\bdist.win-amd64\egg\Nouvelle_Bib
copying build\lib\Nouvelle_Bib\__init__.py -> build\bdist.win-amd64\egg\Nouvelle_Bib
byte-compiling build\bdist.win-amd64\egg\Nouvelle_Bib\bonjour.py to bonjour.cpython-39.pyc
byte-compiling build\bdist.win-amd64\egg\Nouvelle_Bib\__init__.py to __init__.cpython-39.pyc
creating build\bdist.win-amd64\egg\EGG-INFO
copying Nouvelle_Bib.egg-info\PKG-INFO -> build\bdist.win-amd64\egg\EGG-INFO
copying Nouvelle_Bib.egg-info\SOURCES.txt -> build\bdist.win-amd64\egg\EGG-INFO
copying Nouvelle_Bib.egg-info\dependency_links.txt -> build\bdist.win-amd64\egg\EGG-INFO
copying Nouvelle_Bib.egg-info\top_level.txt -> build\bdist.win-amd64\egg\EGG-INFO
zip_safe flag not set; analyzing archive contents...
creating dist
creating 'dist\Nouvelle_Bib-0.0.1-py3.9.egg' and adding 'build\bdist.win-amd64\egg' to it
removing 'build\bdist.win-amd64\egg' (and everything under it)
Processing Nouvelle_Bib-0.0.1-py3.9.egg
Copying Nouvelle_Bib-0.0.1-py3.9.egg to c:\users\dell\appdata\local\programs\python\python39\lib\site-packages
Adding Nouvelle-Bib 0.0.1 to easy-install.pth file

Installed c:\users\dell\appdata\local\programs\python\python39\lib\site-packages\nouvelle_bib-0.0.1-py3.9.egg
```

مكتبة مثبتة بشكل جيد!

02- التعامل مع المكتبات

إنشاء المكتبات

- يؤدي تثبيت المكتبة إلى إنشاء ملف: Nouvelle_Bib-0.0.1-py3.9.egg في مجلد python\Lib\site-packages

```
Installed c:\users\dell\appdata\local\programs\python\python39\lib\site-packages\nouvelle_bib-0.0.1-py3.9.egg
```

- استخدام المكتبة التي تم إنشاؤها

```
Nouvelle_Bib import bonjour #استيراد وحدة Nouvelle_Bib من مكتبة bonjour
```

```
e=hello.Hello("Meriam") #Call من فئة Hello من وحدة الترحيب
```

```
عرض ()
```



التعامل مع المكتبات



1. تثبيت المكتبات الخارجية (نقطة)
2. إنشاء المكتبات.
3. استيراد **المكتبات**

02- التعامل مع المكتبات

استيراد المكتبات

استيراد وحدة نمطية

استخدام وحدة بايثون في المكتبة القياسية
أو وحدة نمطية لمكتبة تم إنشاؤها، يجب عليك استخدام بناء الجملة التالي:

استيراد <وحدة نمطية>

على سبيل المثال، الوحدة العشوائية، نستخدم الصيغة التالية:

استيراد عشوائي

سوف نقوم بتحديد الوظائف المستوردة عن طريق إضافة اسمها إلى اسم الوحدة.

مثلا :

استيراد طباعة عشوائية ((random.choice('aaioer')))

وظيفة الاختيار: اختيار عنصر عشوائياً من القائمة

02- التعامل مع المكتبات

استيراد المكتبات

استيراد وظيفة معينة من الوحدة النمطية

- لاستيراد وظيفة من وحدة نمطية، على سبيل المثال وظيفة الاختيار (.choice) للوحدة العشوائية، نستخدم بناء الجملة التالي:

من اختيار الاستيراد العشوائي

- سنقوم ببساطة بتحديد الوظيفة المستوردة باسمها.

مثال:

من اختيار الاستيراد العشوائي

`print(choice('aaioer')) #print r.`

- الوظائف الأخرى للوحدة غير متوفرة:

```
من اختيار الاستيراد العشوائي
#error
randint(12,42)
```

#ملخص:

```
#randint(12,42)#File "C:/Users/DELL/Desktop/poo/e3.py".
آخر مكالمة): في السطر #Traceback
<module>
```

"randint". #NameError: لم يتم تعريف الاسم.

02- التعامل مع المكتبات

استيراد المكتبات

استيراد كافة وظائف الوحدة النمطية

- لاستيراد كافة وظائف الوحدة النمطية.
- ثم نستخدم بناء الجملة:

من الاستيراد العشوائي*

* سنقوم ببساطة بتحديد الوظائف المستوردة بأسمائها.

مثلا :

```
من الاستيراد العشوائي*
print(choice('ui')) #display i
طباعة (راندنت ((12.42))#عرض 27
```