Exceptions

الإستثناءات و التعامل مع الأخطاء Exceptions & Error Handling

خلال رحلتك في تطوير مشروعك على البايثون ستواجه بطبيعة الحال بعض الأخطاء و الإستثناءات التي ستوقف برنامجك .
تنقسم الاستثناءات في البايثون الى قسمين :

- أخطاء التركيبة اللغوية (Syntax Errors)
- الأخطاء أثناء تشغيل البرنامج (Runtime errors)
- الأخطاء المنطقية (Logical / Symantec errors)

اخطاء التركيبة اللغوية

هذه الأخطاء متعلقة بتركيبة اللغة Syntax ، حيث تحصل عندما لا يتم اتباع قواعد التركيبة الصحيحة . مثل عدم كتابة اسماء الوظائف بشكل صحيح ، نسيان قوس ، و ما شاكل . و تعتبر هذه الاخطاء من السهل العثور عليها و بالعادة سيساعدك المحرر في اكتشافها و اصلاحها حتى .

الأخطاء أثناء التشغيل

هي الأخطاء التي تحصل اثناء تشغيل البرنامج ، و بالعادة قد لا تكون سهلة الإكتشاف . مثل فتح ملف غير موجود ، القسمة على صفر ، و ما شاكل . او الحاجات التي تعتمد على الارتباط بمصادر أخرى ، أو استقبال معطيات من المستخدم .

عندما لا يتم التعامل بشكل صحيح مع هذه الإستثناءات ، سيتم ايقاف عمل برنامجك بشكل فجائي . لذا يفضل دائما التعامل معها من اجل الحفاظ على سير البرنامج بشكل صحيح و سلس و حفظ سلامة البيانات .

عند حصول استثناء ، يقوم مترجم البايثون بطباعة معلومات كاملة عن هذ الإستثناء ، سبب حصوله ، نوعه ، الملف الذي حصل فيه ، و غيرها من المعلومات المهمة لتتبع الخطأ .

مثال :

```
Traceback (most recent call last):
  File "/path/to/project/python code along/exceptions.py", lin
e 4, in <module>
    1/0
ZeroDivisionError: division by zero
```

الأخطاء المنطقية

هي تلك الأخطاء التي تكون بالمنطق . فالكود صحيح ، البرنامج يشتغل و يعطي النتيجة . لكن النتيجة تكون غير المرجوة ، أو المطلوبة . مثلا عملية حسابية المفترض أن تنتج لنا 25 لكنها نتاج لنا 30 . الكود شغال لكن النتجية خاطئة .

التعامل مع الإستثناءات

تتيح لنا لغة البايثون التعامل مع الإستثناءات بشكل فعال ، و ذلك من خلال استخدام Try...Except ... Finally . ففي حال كانت هنالك عملية قد ترفع استثناء ، نقوم بإحاطتها (تغليفها) في Try...Except

```
#basic catching erros with try , except
try:
    print(10/number)
except:
    print("You are dividing by zero")
```

الأعلى تعلمنا كيف نتعامل مع استثناء بشكل عام . و لكن بإمكاننا ايضا التعامل مع اخطاء محددة ، من اجل ذلك نستخدم التركيبة التالية .

```
try:
    print(10/number)
except DivisionByZero:
    print("You are dividing by zero")
```

بالإمكان ايضا التعامل مع عدة انواع من الأخطاء بشكل مستقل ، باستخدام التركيبة التالية.

```
try:
    print(divideBy10(number))
except DivisionByZero:
    print("You are dividing by zero")
except (ValueError, ValueTooBig) as ve:
    print(f"something went wrong : {ve}")
except Exception as e:
    print(e.__class__)
```

استخدام العبارة **else** في التعامل مع الإستثناءات

في بعض الأحيان ، قد يحتاج المبرمج ان ينفذ عملية معينة في حال أن العمليات داخل الـ try تم تنفيذها بنجاح من دون أخطاء . في هذه الحالة ، نستخدم else

```

try:
    print(divideBy10(number))
except DivisionByZero:
    print("You are dividing by zero")
else:
    print("operation is successful")
```

استخدام العبارة **finally** في التعامل مع الإستثناءات

في بعض الحالات ، قد يحتاج المبرمج أن ينفذ عملية معينة بغض النظر اذا كانت العملية داخل الـ try تم تنفيذها بنجاح أو مع وجود اخطاء . بحيث يتم تنفيذها بغض النظر عن النتيجة في الـ try . تُستخدم عادة لتنظيف الموارد أو اغلاق بعض الإرتباطات .

```
try:
    print(divideBy10(number))
except ZeroDivisionError:
    print("You are dividing by zero")
except (ValueError, ValueTooBig) as ve:
    print(f"something went wrong : {ve}")
except Exception as e:
    print(e.__class__)
else:
    print("operation is successful")
finally:
    print("do some operation/s wether an exception is raised o
r not")
```