PYTHON

Advanced OOP

Advanced techniques of creating and serving exceptions

ADVANCED TECHNIQUES OF CREATING AND SERVING EXCEPTIONS

- ¿Qué es una exception?
  - Python contiene 63 (este número puede variar en función de la versión de Python) excepciones integradas (*built-in exceptions*).
  - Todas heredan de *BaseException*.
  - Las nuevas excepciones creadas deben heredar de *BaseException* o de alguna de las excepciones derivadas.
  - Las excepciones se gestionan a través de los bloques *try-except*.

```python
try:
    print(int('a'))
except ValueError:
    print('Error de conversión')
```

https://docs.python.org/3/library/exceptions.html

ADVANCED TECHNIQUES OF CREATING AND SERVING EXCEPTIONS

- Se puede recoger en una variable la la excepción utilizando la palabra clave **as**.
- El atributo **args** de la excepción contiene una tupla con la información de esta.

```python
try:
    print(int('a'))
except ValueError as ve:
    print('Error de conversión')
    print(ve.args)
```

```
Error de conversión
("invalid literal for int() with base 10: 'a'",)
```

ADVANCED TECHNIQUES OF CREATING AND SERVING EXCEPTIONS

- Algunas excepciones tienen más información en otros atributos además de **args**.
  - *ImportError* incluye los atributos:
    - *name (de la librería que se quería importar)*
    - *path (la ruta del fichero que lanzó la excepción –puede ser None-).*

```
try:
    import abcdefghijk

except ImportError as e:
    print(e.args)
    print(e.name)
    print(e.path)
```

```
("No module named 'abcdefghijk'",)
abcdefghijk
None
```

ADVANCED TECHNIQUES OF CREATING AND SERVING EXCEPTIONS

- Algunas excepciones tienen más información en otros atributos además de **args**.
  - *UnicodeError* incluye los atributos:
    - *encoding – the name of the encoding that raised the error.*
    - *reason – a string describing the specific codec error.*
    - *object – the object the codec was attempting to encode or decode.*
    - *start – the first index of invalid data in the object.*
    - *end – the index after the last invalid data in the object.*

```
try:
    b'\x80'.decode("utf-8")
except UnicodeError as e:
    print(e)
    print(e.encoding)
    print(e.reason)
    print(e.object)
    print(e.start)
    print(e.end)
```

```
'utf-8' codec can't decode byte 0x80 in position 0: invalid start byte
utf-8
invalid start byte
b'\x80'
0
1
```

ADVANCED TECHNIQUES OF CREATING AND SERVING EXCEPTIONS

- Encadenamiento de excepciones (*chained exceptions*).
  - Permite mantener la información de una excepción cuando en su tratamiento se produce otra.
  - Permite transformar una excepción en otra sin perder información.
  - Se basa en el uso de dos atributos:
    - __context__, propio de las excepciones encadenadas implícitamente
    - __cause__, propio de las excepciones encadenadas explícitamente

ADVANCED TECHNIQUES OF CREATING AND SERVING EXCEPTIONS

- Encadenamiento de excepciones **implícito**.

```
a_list = ['First error', 'Second error']

try:
    print(a_list[3])
except Exception as e:
    try:
        # the following line is a developer mistake - they wanted to print progress
as 1/10 but wrote 1/0
        print(1 / 0)
    except ZeroDivisionError as f:
        print('Inner exception (f):', f)
        print('Outer exception (e):', e)
        print('Outer exception referenced:', f.__context__)
        print('Is it the same object:', f.__context__ is e)
```

ADVANCED TECHNIQUES OF CREATING AND SERVING EXCEPTIONS

- Encadenamiento de excepciones **explícito**.
  - *Uso combinado de **raise** y **from***

```
class RocketNotReadyError(Exception):
    pass


def personnel_check():
    try:
        print("\tThe captain's name is", crew[0])
        print("\tThe pilot's name is", crew[1])
        print("\tThe mechanic's name is", crew[2])
        print("\tThe navigator's name is", crew[3])
    except IndexError as e:
        raise RocketNotReadyError('Crew is incomplete') from e

crew = ['John', 'Mary', 'Mike']
print('Final check procedure')

try:
    personnel_check()
except RocketNotReadyError as f:
    print('General exception: "{}", caused by "{}"'.format(f, f.__cause__))
```

ADVANCED TECHNIQUES OF CREATING AND SERVING EXCEPTIONS

- Encadenamiento de excepciones explícito. Transformación de excepciones.

```
class RocketNotReadyError(Exception):
    pass


def personnel_check():
    try:
        print("\tThe captain's name is", crew[0])
        print("\tThe pilot's name is", crew[1])
        print("\tThe mechanic's name is", crew[2])
        print("\tThe navigator's name is", crew[3])
    except IndexError as e:
        raise RocketNotReadyError('Crew is incomplete') from e


def fuel_check():
    try:
        print('Fuel tank is full in {}%'.format(100 / 0))
    except ZeroDivisionError as e:
        raise RocketNotReadyError('Problem with fuel gauge') from e


crew = ['John', 'Mary', 'Mike']
fuel = 100
check_list = [personnel_check, fuel_check]

print('Final check procedure')

for check in check_list:
    try:
        check()
    except RocketNotReadyError as f:
        print('RocketNotReady exception: "{}", caused by "{}"'.format(f, f.__cause__))
```

```
Final check procedure
        The captain's name is John
        The pilot's name is Mary
        The mechanic's name is Mike
RocketNotReady exception: "Crew is incomplete", caused by "list index
out of range"
RocketNotReady exception: "Problem with fuel gauge", caused by
"division by zero"
```

ADVANCED TECHNIQUES OF CREATING AND SERVING EXCEPTIONS

- Atributo __trackback__. Contiene información sobre el origen de la excepción.
  - Se encuentra en el módulo **trackback**

```python
import traceback

class RocketNotReadyError(Exception):
    pass


def personnel_check():
    try:
        print("\tThe captain's name is", crew[0])
        print("\tThe pilot's name is", crew[1])
        print("\tThe mechanic's name is", crew[2])
        print("\tThe navigator's name is", crew[3])
    except IndexError as e:
        raise RocketNotReadyError('Crew is incomplete') from e


crew = ['John', 'Mary', 'Mike']

print('Final check procedure')

try:
    personnel_check()
except RocketNotReadyError as f:
    print(f.__traceback__)
    print(type(f.__traceback__))
    print('\nTraceback details')
    details = traceback.format_tb(f.__traceback__)
    print("\n".join(details))

print('Final check is over')
```

```
Final check procedure
        The captain's name is John
        The pilot's name is Mary
        The mechanic's name is Mike
<traceback object at 0x000002BE50FCBA40>
<class 'traceback'>

Traceback details
  File "h:\Mi unidad\___Transparencias_propias\Python\EDUBE-Python-Professional-Advanced OOP\ejemplo.py", line 22, in <module>
    personnel_check()

  File "h:\Mi unidad\___Transparencias_propias\Python\EDUBE-Python-Professional-Advanced OOP\ejemplo.py", line 14, in personnel_check
    raise RocketNotReadyError('Crew is incomplete') from e

Final check is over
```