# Python
# L'instruction `with` et les gestionnaires de contexte

Nicolas Delestre

INSA
ROUEN NORMANDIE

# Un motif de développement courant

## Exemple de motif courant

```
1 # instructions I1 d'initialisation d'un objet o
2 try :
3     # instructions I2 utilisant l'objet o
4     # pouvant lever une exception
5 except UneException as e:
6     # instructions I3 s'il y a eu une exception
7  finally :
8     # instructions I4 à réaliser qu'il y ait eu exception ou pas
```

## Dans quel contexte ?

- l'utilisation d'un fichier (ouverture, lecture/écriture, fermeture)
- communication réseau
- etc.

# Un exemple avec la lecture d'un fichier

## Exemple : affichage du contenu d'un fichier

```
1 f = open("fichier.txt", "rt")
2 try:
3     for ligne in f:
4         print(ligne, end="")
5 except Exception:
6     pass
7 finally :
8     f.close()
```

# L'instruction `with`

## with [...as]

- utilisation d'un gestionnaire de contexte (*context manager*) qui :
  - possède et exécute *I1*
  - retourne l'objet *o* afin de pouvoir exécuter la liste des instruction *I2*
  - possède et exécute automatiquement *I3* et *I4* une fois les instructions *I2* entièrement exécutées ou arrêtées par une exception
- Deux syntaxes :

```
gc = gestionnaire_de_contexte()
with gc:
    I2(gc)
```

```
with gestionnaire_de_contexte() [as gc]:
    I2(gc)
```

## Inspire de https://www.afpy.org/doc/python/3.5/tutorial/errors.html

```
with open("fichier.txt", "rt") as f:
    for ligne in f:
        print(ligne, end="")
```

## À l'aide d'une classe

- Toute classe qui possède les méthodes :
  - `__enter__(self)` qui exécute *I1* retourne l'objet o
  - `__exit__(self, except_type, except_value, traceback)` qui exécute optionnellement *I3* et qui exécute *I4*
    - `except_type` : `None` ou le type de l'exception qui a été levée
    - `except_value` : `None` ou la chaîne de caractères qui décrit l'exception
    - `traceback` : `None` ou un objet permettant d'afficher la trace d'exécution

```
3  class UneException(Exception):
4      pass
5
6  class UnGestionnaireDeContexteJouet:
7      def __enter__(self):
8          print("I1")
9          return #on peut spécifier un objet qui sera récupéré dans le as
11     def __exit__(self, except_type, except_value, traceback):
12         if except_type:
13             print(f"I3 : Type = {except_type}, valeur = {except_value}")
14         print("I4")
```

## Exemples d'utilisation

```
>>> with UnGestionnaireDeContexteJouet() as cm:
    print("I2")
I1
I2
I4
>>> with UnGestionnaireDeContexteJouet() as cm:
    print("I2")
    raise Exception("une erreur")
I1
I2
I3 : Type = <class '__main__.UneException'>, valeur = une erreur
I4
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File ".../context_manager.py", line 15, in <module>
    raise UneException("une erreur")
UneException: une erreur
```

## À l'aide d'une fonction

- Toute fonction décorée par le décorateur `contextmanager` (du module `contextlib`) et qui a le corps suivant :

```
try :
    # instructions à exécuter au début
    yield # optionnellement avec un objet qui sera récupéré dans le as
    # instructions exécutées s'il n'y a pas eu d'exception de levée
except Exception as err:
    # instruction exécutées s'il y a eu une exception de levée
finally :
    # instructions à exécuter après
```

```
16 from contextlib import contextmanager
17
18 @contextmanager
19 def unGestionnaireDeContexteJouet():
20     try :
21         print ("I1")
22         yield #on peut spécifier un objet qui sera récupéré dans le as
23     except Exception as err:
24         print (f"I3 : Type = {type(err)}, valeur = {err}")
25         raise err
26     finally :
27         print ("I4")
```

## Exemples d'utilisation

```
>>> with unGestionnaireDeContexteJouet() as cm:
    print("I2")
I1
I2
I4
>>> with unGestionnaireDeContexteJouet() as cm:
    print("I2")
    raise UneException("une erreur")
I1
I2
I3 : Type = <class '__main__.UneException'>, valeur = une erreur
I4
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File ".../context_manager.py", line 15, in <module>
    raise UneException("une erreur")
UneException: une erreur
```

# Gestionnaires de contexte particuliers

## Gestionnaires de contexte réentrants

```
gc = GestionnaireDeContexteReentrant()
with gc:
    # instructions
    with gc:
        # instructions
```

## Gestionnaires de contexte réutilisables

```
gc = GestionnaireDeContexteReutilisiable()
with gc:
    # instructions
# instructions
with gc:
    # instructions
```

# Conclusion

- Un motif de développement courant
- Utilisation de l'instruction `with`
- Conception de gestionnaire de contexte

INSA ROUEN NORMANDIE