Python
Instructions de base

Nicolas Delestre

INSA
ROUEN NORMANDIE

Rappels

Objets, attributs et méthodes

- Tout est objet en Python
- Un objet a un état défini par « les valeurs » de ces attributs
- On accède à un attribut `a` d'un objet à l'aide de la notation pointée (`o.a`)
- On peut agir (interroger, modifier, obtenir d'autres objets) sur l'objet en envoyant un message à un objet
- On envoie un message (`m`) à un objet `o` (ou on invoque, ou appelle, une méthode `m` d'un objet `o`) à l'aide de la notation pointée avec entre parenthèses (obligatoires) des paramètres effectifs (optionnels) : `o.m()`
- Python cherche alors la méthode (le code python) à interpréter

INSA ROUEN NORMANDIE

Affectation

=

- Instruction qui permet de référencer un objet à l'aide d'une variable

- Affectation des références

```
>>> a = 12.5
>>> b = 12.5
>>> c = a
>>> a is b
False
>>> a is c
True
```

- Possibilité de faire plusieurs affectations en une seule fois (utilisation des tuples)

```
>>> a,b = 1,2
>>> a,b = b,a
>>> a
2
>>> a = 1,2
>>> a
(1,2)
```

- L'affectation est aussi une opération, les références retournées sont celles affectées

```
>>> a = b = 1
>>> a
1
>>> b
1
```

Attention

Par abus de langage on dit pour décrire l'instruction `a = 1` que « a vaut 1 » mais on devrait dire que « a référence l'objet `int 1` »

Conditionnelle

if [elif] [else]

- Syntaxe :

```
if condition:
    ...
[elif condition:
    ...
]
[else:
    ...
]
```

```
if anciennete < 6:
    nb_jours = anciennete
elif anciennete < 12:
    nb_jours = 2 * anciennete
else:
    nb_jours = 28
if cadre:
    if age >= 35 and anciennete >= 36:
        nb_jours = nb_jours + 2
    if age >= 45 and anciennete >= 60:
        nb_jours = nb_jours + 4
```

INSA ROUEN NORMANDIE

Itération déterministe 1 / 3

Itérable

- Objet dont on peut parcourir les valeurs
- Les séquences, les ensembles et les dictionnaires sont des itérables (pour les dictionnaires cela permet de parcourir les clés)

`for in [else]`

- Syntaxe :

```
for e in iterable:
    ...
[else:
    ..
]
```

- La partie `else` est exécutée lorsque l'itérable a été parcouru entièrement (pas exécutée si on sort de la boucle à cause d'un `break`)

Itération déterministe 2 / 3

```
>>> for jour in ("lundi", "mardi", "mercredi", "jeudi", \
...              "vendredi", "samedi", "dimanche"):
...     print(jour)
...
lundi
mardi
mercredi
jeudi
vendredi
samedi
dimanche
```

ROUEN NORMANDIE

Itération déterministe 3 / 3

Fonction enumerate

- Syntaxe :

```
for i,e in enumerate(iterable):
    ...
```

```
>>> for numero, jour in enumerate(("lundi", "mardi", "mercredi",\
...                                "jeudi", "vendredi", "samedi",\
...                                "dimanche")):
...     print(f"{numero} {jour}")
...
0 lundi
1 mardi
2 mercredi
3 jeudi
4 vendredi
5 samedi
6 dimanche
```

Itération indéterministe

while [else]

- Syntaxe :

```
while condition:
    ...
[else:
    ...
]
```

- La partie `else` est exécutée lorsque la condition devient fausse (pas exécutée si on sort de la boucle à cause d'un `break`)

INSA ROUEN NORMANDIE

Opérateur de morse

- La version 3.8 de python a introduit l'opérateur de morse qui permet de réaliser des affectations dans les conditions des conditionnelles ou des itérations indéterministes
- Syntaxe : `:=`

Exemple inspiré de `https://www.dad3zero.net/202002/python-walrus-operator/`

```
long = len(ma_sequence)
if long > 10:
    print(f"Vous avez {long} éléments")
```

```
if (long := len(ma_sequence)) > 10:
    print(f"Vous avez {long} éléments")
```

INSA ROUEN NORMANDIE

Autres instructions 1 / 2

Instructions qui peuvent être utiles

- `pass` instruction qui ne fait rien : utile lorsque l'on reporte le développement d'un corps (d'un `if`, `for`, `while` ou une fonction) à plus tard
- `break` instruction qui permet de sortir prématurément d'une itération
- `continue` instruction qui permet de passer à l'itération suivante sans interpréter les instructions qui suivent
- `del` instruction qui permet de supprimer la référence d'une variable

```
>>> a=(1, 2)
>>> del(a)
>>> a
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'a' is not defined
```

Autres instructions 2 / 2

Instructions qui seront présentées dans de futures cours

- `import` et `from...import` instructions qui permettent d'importer un module, une fonction une classe, etc.
- `try...catch` et `raise` instructions qui permettent de gérer les erreurs (exceptions)
- `return` instruction qui permet de sortir d'une fonction en retournant (optionnellement) la référence vers un objet
- `yield` instruction qui permet de définir des générateurs
- `assert` instruction qui permet d'ajouter une assertion (vérification)
- `global` et `nonlocal` instructions qui permettent de modifier la portée d'une ou de plusieurs variables qui seraient locales sans ces instructions

INSA ROUEN NORMANDIE

Conclusion

Dans ce cours nous avons...

- rappelé que tout est objet en Python et qu'une variable Python référence un objet
- défini le rôle de l'instruction d'affectation
- listé les instructions permettant de coder les schémas conditionnels et itératifs
- listé d'autres instructions dont la plupart seront présentés dans de futurs cours

INSA ROUEN NORMANDIE