Python

Cours 3 - Conditions

Table des matières

Conditions Logiques

Il est possible grâce aux **conditions logiques** de faire des opérations conditionnelles reflétant la vie réelle.

Si un paiement est refusé, si vous vous levez tôt le matin, si vous allez manger au restaurant ou à la maison...

Tout peux se rejoindre au principe de logique conditionnelle.

Tests Logiques - Opérandes Mathématiques

Voici des opérandes traditionnels mathématiques à utiliser **pour créer** les déclarations conditionnelles ("if", etc.) :

- Equals: a == b
- Not Equals: a != b
- Less than: $a < b$
- Less than or equal to: $a \leq b$
- Greater than: $a > b$
- Greater than or equal to: $a \geq b$

Tests Logiques - Opérandes Logique

Voici des opérandes traditionnels logiques à utiliser **avec** les déclarations conditionnelles ("if", etc.) :

- AND: Les deux sont vrais
- OR: Un ou l'autre est vrai
- NOT: N'est pas (l'opposé de)

Déclaration “If”

Cette déclaration est la plus basique, elle permet de considérer le résultat de la condition logique et de passer les instructions à l’intérieur de celle-ci lorsque **true**.

Si le résultat de la condition logique est **false**, le code indenté à l’intérieur du “if” n’est pas exécuté.

Figure: Si la condition est vraie, le code est exécuté.

Déclaration “If”

```
1  # Steve Levesque, All rights reserved
2
3  if 3 > 2:
4      print("This is true.")
5
6  """
7  This is true.
8  """
```

Déclaration “elif”

Figure: Si la condition est fausse pour l’évaluation précédente, l’évaluation du “elif” est exécuté.

Si la condition précédente n’est pas vraie, exécuter **l’évaluation** cette condition présente.

Attention, il y a une différence entre des “if” un par-dessus l’autre, puisque le “elif” s’assure que la déclaration précédente soit fausse avant d’être exécuté. En d’autres mots, elle **dépends** de la déclaration précédente.

Déclaration “elif”

```
1  # Steve Levesque, All rights reserved
2
3  if 1 > 2:
4      print("This is true, 1 is bigger than 2.")
5  elif 3 == 3:
6      print("This is true, 3 equals 3.")
7
8  """
9  This is true, 3 equals 3.
10 """
```

Déclaration “else”

La déclaration “else” exécute simplement le code à l’intérieur de celle-ci si la condition précédente est fausse.

Simplement écrire le code après le “if” exécute le même principe, mais il y a des situations où ceci n’est pas souhaitable (scope, clarté du code, etc.).

Figure: Si la condition est vraie, le code est exécuté.

Déclaration “else”

```
1  # Steve Levesque, All rights reserved
2
3  if 1 > 2:
4      print("This is true, 1 is bigger than 2.")
5  elif 3 != 3:
6      print("This is true, 3 is different than 3.")
7  else:
8      print("Executed.")
9
10 """
11 Executed.
12 """
```

Déclaration “and”

Figure: Il faut que toutes les déclarations conditionnelles soient vraies pour que le résultat soit vrai.

L’opération logique “and” permet de tester si les deux déclarations conditionnelles sont vraies.

Exemple : Pour avoir accès à la plateforme étudiante de mon campus, je dois avoir mon mot de passe exact ET l’authentification double de mon téléphone.

Déclaration “and”

```
1  # Steve Levesque, All rights reserved
2
3  exact_password = True
4  phone_double_factor = True
5
6  if exact_password and phone_double_factor:
7      print("Portal accessed.")
8
9  """
10 Executed.
11 """
```

Déclaration “or”

L’opération logique “or” permet de tester si au moins une déclaration conditionnelle est vraie parmi les deux choix.

Exemple : Pour avoir accès au campus, il faut que j’aie ma carte étudiante OU un accès à mon compte étudiant.

Figure: Tant qu’une des déclarations conditionnelles est vraie, le résultat est vrai.

Déclaration “or”

```
1  # Steve Levesque, All rights reserved
2
3  student_card = True
4  student_account = False
5
6  if student_card and student_account:
7      print("Campus entry - strong security.")
8  if student_card or student_account:
9      print("Campus entry - low security 1.")
10
11 student_card = False
12
13 if student_card or student_account:
14     print("Campus entry - low security 2.")
15
16 """
17 Campus entry - low security 1.
18 """
```

Déclaration “not”

Figure: Il faut que la déclaration conditionnelle soit fausse, pour que la négation de celle-ci (évaluation du “not”) retourne vrai.

L’opération logique “not” permet de tester si la valeur est la négation d’un état.

Dans certains cas, avoir la négation permet de respecter une condition, mais il faut se rappeler que le respect d’une condition doit être vrai. Il faut alors annuler cette négation.

Exemple : Le système doit détecter si le cours sélectionné par l’étudiant est indisponible et retourner un message d’erreur.

Déclaration “not”

```
1  # Steve Levesque, All rights reserved
2
3  student_wanted_course = "ReactJS"
4
5  courses = ["Python", "NoSQL", "Artificial Intelligence"]
6
7  if student_wanted_course not in courses:
8      print("Course not available...")
9
10 """
11 Course not available...
12 """
```

Bibliographie

- https://www.w3schools.com/python/