# Fondements informatiques I

## Cours 2: Structures conditionnelles

Sorina Ionica sorina.ionica@uvsq.fr

Sandrine Vial sandrine.vial@uvsq.fr

# Les structures conditionnelles

Utilisées pour exécuter certaines instructions dans un cas spécifique, et d'autres instructions dans d'autres cas.

## Exemple

Si le solde bancaire est insuffisant alors rejeter la transaction
Autrement continuer.

Les instructions conditionnelles contrôlent quelle partie du code est exécutée en fonction de certaines conditions.

## Trois structures

- `if`
- `if...else`
- `if...elif...else`

# L'instruction `if`

- Jusqu'à présent, les différentes instructions saisies sont toutes exécutées d'une manière linéaire (consécutive).
- Avec les structures conditionnelles, on peut sauter des instructions dans le cas où certaines conditions ne seraient pas satisfaites.

## Syntaxe

```
if <expression> :
    instruction1
    instruction2
```

## Bloc d'instructions

Une série d'instructions qui s'exécute dans un cas précis.
L'indentation : un décalage vers la droite permettant d'identifier où se trouvent le début et la fin d'un bloc d'instructions.

# L'instruction `if`

## Example

```
meteo = "pluie"
if meteo == "pluie" :
    # si égalité alors afficher le message "parapluie"
    print("parapluie")
    # si égalité alors afficher le message "impermeable"
    print("impermeable")
print("Bonne journée !") # afficher le message "Bonne journée"
```

- L'opérateur == permet de vérifier si la variable `meteo` est égale à pluie. Que renvoie-t-il ?
- Le message "Bonne journée !" s'affiche car non-indenté !

# L'instruction `if-else`

## Syntaxe

```
if <expression> :
     instruction(s)
else :
    instructions(s)
```

- Le mot clé `else` doit être au même niveau d'indentation que l'instruction if qu'elle complète.
- Le code de bloc `else` s'exécute si le code de bloc if ne s'exécute pas.
- else doit également être suivi de deux points (`:`).

# L'instruction if - else

```
meteo = "froid"
if meteo == "pluie":
    print("parapluie")
    print("impermeable")
else :  # si meteo n'est pas "pluie" alors affiche ce qui suit
    print("t-shirt")
    print("shorts")
print("Bonne journée !")
```

Le message "Bonne journée !" s'affiche car non-indenté !

# L'instruction `elif`

## Syntaxe

```
if <expression>  :
   instructions(s)
elif <expression> :
   instruction(s)
else :
   instructions(s)
```

- Le mot clé `elif` doit être au même niveau d'indentation que l'instruction if qu'elle complète.
- `elif` doit également être suivi d'une condition et de deux points (`:`).
- L'instruction else ne peut figurer qu'une seule fois en clôture du bloc de la condition if.

# L'instruction elif

```
meteo = "froid"
if meteo == "pluie" : #  vérifie si meteo a "pluie" comme valeur
    print("parapluie")
    print("impermeable")
elif meteo == "froid" : #  sinon vérifie si meteo est "froid"
    print("pull")
    print("echarpe")
else : # si meteo n'est ni "pluie" ni "froid" afficher ce qui suit
    print("t-shirt")
    print("shorts")
print("Bonne journée !")
```

- Il est possible de mettre autant de `elif` que l'on souhaite après une condition `if`.
- Les instructions `elif` et `else` sont facultatives : lorsqu'une instruction en `if` ou `elif` est définie, il n'est pas obligatoire de prévoir un `else` après.

Trouver l'erreur dans le code suivant :

```
meteo = "froid"
if meteo == "froid" :
    print("echarpe")
print("Bonne journée !")
elif meteo == "froid" or meteo == "venteux" :
    print("manteau")
    print("Bonne journée !")
else : # sinon affiche ce qui suit
    print("t-shirt")
print("Bonne journée !")
```

# Les structures conditionnelles et les opérateurs de comparaison

Les opérateurs relationnels retournent `True` ou `False`, nous pouvons les utiliser dans des instructions conditionnelles.

| Opérateur | Signification |
|---|---|
| < | strictement inférieur à |
| > | strictement supérieur à |
| <= | inférieur ou égal à |
| >= | supérieur ou égal à |
| == | égal à |
| != | différent de |

Exemple : `a>= b` renvoie `True` si `a` est supérieur ou égal à `b`, `False` sinon.

# Exemple

```
solde = 20.0
prix_achat_hors_taxe = 19.0
taxe = 1.08
# comparer le solde bancaire du client au prix d'achat TTC
if solde >= prix_achat_hors_taxe * taxe :
    print("Transaction possible")
else:
    print("Transaction impossible")
print("Fin")
```

Remarque : afin de comparer, on exécute en premier la multiplication

```
prix_achat_hors_taxe * taxe
```

# Priorité des opérateurs

| Opérateurs |
| --- |
| () |
| ** |
| *, /, //, % |
| ==, !=, <, >, <=, => |
| not |
| and |
| or |

Que valent les expressions suivantes ?

```
a,b=3, 2
not a > 0 and 5*b <= 3
not (a>0 and 5*b <= 3)
```

# Évaluation paresseuse des expressions avec `and` et `or`

Les expressions qui ne doivent pas être évaluées pour déterminer le résultat ne sont pas évaluées.

## `x and y`

On évalue d'abord `x`.
Si `x` vaut `False`, sa valeur est renvoyée et `y` n'est pas évaluée.
Sinon, la variable `y` est évaluée et la valeur résultante est renvoyée.

## `x or y`

On évalue d'abord `x`.
Si `x` vaut `True`, sa valeur est renvoyée et `y` n'est pas évaluée.
Sinon, la variable `y` est évaluée et la valeur résultante est renvoyée.

# Exemples avec and

```
x = 0
if x != 0 and 2 // x == 2 :
    print("Test de l'évaluation paresseuse")
print("Fin")
```

```
x = 0
if 2 // x == 2 and x != 0:
    print("Test de l'évaluation paresseuse")
print("Fin")
```

# Exemple avec or

```
x = 0
if x == 0 or 2 // x == 2:
    print("Test de l'évaluation paresseuse.")
print("Fin")
```

```
x = 0
if 2 // x == 2 or x == 0:
    print("Test de l'évaluation paresseuse.")
print("Fin")
```

Les opérateurs and et or ne sont pas symétriques.

# La portée des variables

On appelle **portée d'une variable** (ou **scope**) l'ensemble des endroits du programme où elle existe.

## Exemple

```
num1 = 1
num2 = 2
resultat = "result non connu"
if num1 < num2 :
    resultat = "num2 est plus grand que num1!"
print(resultat)
print("Fin")
```

- La portée de `resultat` est de la ligne 3 jusqu'à ce que le programme se termine.
- La ligne 5 est dans la portée de la variable `resultat`.

# La portée des variables

```
num1 = 2
num2 = 1
if num1 < num2 :
    max = num2
print(max)
print("Fin")
```

En Python, la portée d'une variable commence lorsqu'elle est créée et se termine lorsque l'une des nombreuses terminaisons se produit (par exemple la fin du programme ou l'utilisation des fonctions).

# Instructions `if` imbriquées

On peut avoir des instructions `if` à l'intérieur d'un bloc d'instructions `if`.

## Syntaxe

```
if <expression> :
      instruction(s)
      if <expression> :
           instruction(s)
```

Quelle est la portée de la variable `num3` ? Que affiche-t-on ?

## Example

```
num1, num2 = 2,5
max= num1
if num1 < num2 :
    max= num2
    num3=3
    if num2 <  num3 :
     max=num3
print(max)
```

# Exemple

Qu'affiche-t-on ?

```
solde = 21.0
taxe = 1.08
nom_detenteur_carte = "Daniel Dupont"
vendeurs_de_confiance = ["Maria", "Yoann", "Emilie", "Tia"]
prix_achat_hors_taxes = 19.0
nom_client = "Daniel Dupont"
vendeur = "Fred"

decouvert_autorise = True
if (solde <= prix_achat_hors_taxes * taxe and not decouvert_autorise) :
    print('Transaction refusée solde insuff. ou pas de découvert autorisé')
else:
    if not nom_detenteur_carte == nom_client :
        print("Transaction non approuvée : client non valide")
    else:
        if not vendeur in vendeurs_de_confiance :
            print("Transaction refusée: vendeur non autorisé")
        else:
            print("Transaction approuvée")
```