Institut National Universitaire Champollion

Chapitre 2 : Alternative

Plan du chapitre

Introduction : Pourquoi l’alternative ?

Introduction : Pourquoi l'alternative ?

- Vérifier que des valeurs sont dans un Domaine
- Éviter des erreurs à l'exécution
- Aide à la résolution d'un problème

Introduction : Pourquoi l'alternative ?

Vérifier que des valeurs sont dans un Domaine

par exemple :

à la saisie, les notes sont des réels compris entre 0 et 20

la somme retirée d'un compte est un réel positif

etc ….

Introduction : Pourquoi l'alternative ?

Éviter des erreurs à l'exécution

par exemple :

éviter une division par zéro :

moyenne = sommeNotes / nbNotes

fichier non disponible à l'endroit spécifié dans le programme

Introduction : Pourquoi l'alternative ?

Aide à la résolution d'un problème

par exemple :

Une année est bissextile ssi :

si l'année est divisible par 4 et non divisible par 100 ;

OU

si l'année est divisible par 400 .

Les booléens

Les booléens

La notion d'assertion logique et d'opérateur logique sont des notions venant des mathématiques :

Une **assertion** est une affirmation ou un énoncé qui est soit vrai, soit faux.

connecteurs logiques élémentaires :

la négation

le « ET »

le « OU »

Les booléens

Table de vérité de la négation

p	non(p)
V	F
F	V

Table de vérité du « ET »

p	q	p ET q
V	V	V
V	F	F
F	V	F
F	F	F

Table de vérité du « OU »

p	q	p OU q
V	V	V
V	F	V
F	V	V
F	F	F

Les booléens

Négation du ET :

non (p ET q) = non(p) OU non(q)

Négation du OU :

non (p OU q) = non(p) ET non(q)

il est nécessaire de maîtriser ces opérations simples pour manipuler correctement les alternatives (et les boucles!!)

Les booléens

En python :

un booléen prend donc soit la valeur :

True pour vrai

False pour faux

True et **False** sont des mots réservés à cet effet en python.

ET : **and**

OU : **or**

négation : **not**

Les booléens

Les opérateurs de comparaison : ==,!=, <=, <, >=, > renvoient un booléen.

Exemple 1 :

Entrée [10]:

```
1 print("les opérateurs de comparaison renvoient un booléen :")
2 print("5 > 6 ? ",5>6)
3 print("5 <= 5 ? ",5<=5)
4 print("coucou == coucou ? ","coucou" == "coucou")
5
```

```
les opérateurs de comparaison renvoient un booléen :
5 > 6 ?  False
5 <= 5 ?  True
coucou == coucou ?  True
```

Les booléens

Exemple 2 :

Entrée [11]:

```
1 print("avec des opérateurs logiques :")
2 p = 6 == 4+2
3 q = 6 > 3 + 1
4 r = 1 > 3
5 print("(p ou q) et r ? ",(p or q) and r)
6 print("(p ou q) ou r ? ",(p or q) or r)
7 print("(p et r) ou q ? ",(p and r) or q)
8 print("((non p) et r) ou r ? ",((not p) and r) or r)
```

```
avec des opérateurs logiques :
(p ou q) et r ?  False
(p ou q) ou r ?  True
(p et r) ou q ?  True
((non p) et r) ou r ?  False
```

Instructions conditionnelles

Instructions conditionnelles

Instruction :

si (condition) alors
    instruction(s)

si (condition) alors
    instruction(s)

sinon
    instruction(s)

Entrée [13]:

```
1 x = 3
2 if x >= 0:
3     print("x = ",x," est positif")
4
```

```
x =  3  est positif
```

Entrée [16]:

```
1 x = -3
2 if x >= 0:
3     print("x = ",x," est positif")
4 else:
5     print("x = ",x," est négatif")
```

```
x =  -3  est négatif
```

vous noterez le : , après la condition et l'indentation qui permet d'identifier l'instruction ou le groupe d'instructions suivant le if ou le else.

l'instruction else est optionnelle.

Instructions conditionnelles

Attention :

- Le corps d’instructions (le groupe d’instruction après le if (ou le else) doit être indenté par rapport au if (ou le else)

- Un corps d’instruction (ou groupe d’instruction) doit être indenté de la même façon.
(on utilise généralement une tabulation)

Instructions conditionnelles

Exemple 1 : indentation obligatoire

Entrée [17]:

```
1 x = 3
2 if x >= 0:
3 print("x = ",x," est positif")
4
```

```
File "<ipython-input-17-8764ceee9c7d>", line 3
  print("x = ",x," est positif")
  ^
IndentationError: expected an indented block
```

Instructions conditionnelles

Exemple 2 : même indentation pour un bloc

Entrée [20]:

```
1 x = -3
2 if x >= 0:
3     valAbs = x
4     print("x ",x," est positif")
5     print("|x| = ", valAbs)
6 else:
7     valAbs = -x
8     print("x ",x," est négatif")
9   print("|x| = ", valAbs)
```

```
File "<tokenize>", line 9
  print("|x| = ", valAbs)
  ^
IndentationError: unindent does not match any outer indentation level
```

Instructions conditionnelles

Conditionnelles imbriquées

On peut bien sûr imbriquer les conditionnelles :

Entrée [22]:

```
1 temp_Eau = 50
2 if temp_Eau < 0:
3     print("c'est de la glace")
4 else:
5     if temp_Eau < 100:
6         print("c'est du liquide")
7     else:
8         print("c'est de la vapeur")
```

```
c'est du liquide
```

Vous remarquerez bien les indentations ...

Instructions conditionnelles

Conditionnelles imbriquées : elif

si (condition) alors
    instruction(s)

sinon si (condition) alors
    instruction(s)

Entrée [26]:

```
1 temp_Eau = 150
2 if temp_Eau < 0:
3     print("c'est de la glace")
4 elif temp_Eau < 100:
5         print("c'est du liquide")
6 else:
7         print("c'est de la vapeur")
```

```
c'est de la vapeur
```

code similaire au précédent.

Instructions conditionnelles

Conditionnelles imbriquées : avantage ?

quelle différence faites entre ces deux codes :

Entrée [33]:

```
1 c = 2
2 if c == 1:
3     couleur = "bleu"
4 elif c == 2:
5     couleur = "rouge"
6 elif c == 3:
7     couleur = "vert"
8 print(couleur)
```

```
rouge
```

Entrée [34]:

```
1 c = 2
2 if c == 1:
3     couleur = "bleu"
4 if c == 2:
5     couleur = "rouge"
6 if c == 3:
7     couleur = "vert"
8 print(couleur)
```

```
rouge
```

Exercices

Exercice 1 : reprenez la diapo 7, et construisez un booléen indiquant si une année est bissextile ou pas.

Exercice 2 : On donne deux entiers a et b. Écrire un code qui calcule le plus petit et le plus grand des deux entiers a et b. Les valeurs calculées seront placées dans des variables mini et maxi.

Exercice 3 : Écrire un programme permettant de savoir si un mois est un mois à 31 jours ou pas. Le mois de janvier sera identifié par 1, février par 2, ...