Boucles et conditions

Faites une copie avec la commande ci dessus "File>Make a Copy...", et renommez la en cliquant sur le titre (à côté de jupyter TD2...), par exemple en rajoutant votre nom.

Le type booléen

Un booléen est une variable qui prend uniquement deux valeurs, vrai ou faux. En ce sens c'est ce qui est le plus proche de la machine. En effet, comme on le verra dans le cours, un ordinateur ne comprend que deux états : le courant passe ou le courant ne passe pas. Les variables booléennes sont utilisées pour les tests ou les boucles. Essayer les expressions suivantes ; pour mieux comprendre et assimiler voir le cours sur les opérateurs de comparaison, ainsi que sur les opérateurs logiques (cours IV 4).

In [ ]:
```
3 > 5
```

In [ ]:
```
a = 4
a >= 4
```

In [ ]:
```
a == 4
```

In [ ]:
```
a != 4
```

In [ ]:
```
a == 4 and 3 > 5
```

In [ ]:
```
a == 4 or 3 > 5
```

In [ ]:
```
a == 4 or 3 < 5
```

In [ ]:
```
not(a==4)
```

On peut écrire des expressions plus complexes avec les booléens. Ce n'est pas forcément recommandé, ni interdit non plus. Si vous faites comme ci-dessous, veillez à ne pas trop perdre en lisibilté et en clarté du code.

In [ ]:
```
a =int(input("Donnez un nombre"))
b = (a > 50)    # les parenthèses ne sont pas obligatoires mais permettent plus de lisibilité
print("b vaut",b,", et est de type ",type(b))
c = (a == 3 or not(b) or not(a != 25) )   # si vous reussissez a deviner la valeur de c, bravo !
                                          # ici c'est illisible...
```

Instructions conditionelles

Lire le cours (VI 6)

Que fait le programme ci-dessous ? Expliquer son fonctionnement (vous pouvez écrire vos idées après "commentaires" avec un double clic sur la ligne "commentaires", puis ctrl + entrée pour avoir le format texte)

In [ ]:
```
a = float(input("rentrez un nombre quelconque :"))
if a//2 == a/2:
    print(a," est un entier pair")
elif a//1 == a:
    print(a," est un entier impair")
else:
    print(a," n'est pas un entier")
```

Commentaires:

Exercice

Ecrire un programme qui demande votre âge, et qui dit:

- que vous n'avez pas le droit de travailler si vous avez moins de 16 ans
- que vous pouvez travailler si vous avez entre 16 et 67 ans
- que vous êtes à la retraite si vous avez plus de 67 ans

Instruction itérative "tant que"

Faire tourner le programme ci-dessous. Comprendre son fonctionnement.

Le changer de manière à ce qu'il compte de 10 jusqu'à 0 inclus. Vous pouvez comparer la solution trouvée avec celles de vos voisins; en effet il y a de nombreuses méthodes.

In [ ]:
```
n = 10
while n > 0:
    print(n)
    n = n - 2
```

On donne la suite de nombres qui commence par 5, et où l'obtient le nombre suivant en multpliant le précédent par 2 et en enlevant 1. On veut trouver après combien d'étapes de calcul on obtient un résultat > 1000000.
L'algorithme est le suivant :

Début

> $u \leftarrow 5$ # notre nombre de départ
>
> $n \leftarrow 0$ # le nombre d'étapes
>
> **Tant que** $u \leq 1000000$:
>
> > $u \leftarrow 2 \times u - 1$
> > $n \leftarrow n + 1$
>
> **Fin tant que**
> **afficher** n

Fin

Programmer cet algorithme, on utilisera la syntaxe de l'exemple ci-dessus (le cours est très succint : VI 7).

L'instruction "range" : l'outil de base pour l'instruction itérative "pour"

L'instruction "range" permet de parcourir un ensemble de valeurs. Tester sur les exemples ci dessous. Vous pouvez mettre des commentaires dans la cellule du même nom

In [ ]:
```
for i in range(10):
    print(i)
```

In [ ]:
```
for i in range(0,10,1):
    print(i)
```

Commentaires :

Exercice : modifier la deuxième boucle ci-dessus de manière à:

1. compter jusqu'à 10 inclus;
2. compter jusqu'à 10 inclus, de 2 en 2;
3. compter de 10 à 0 dans l'ordre décroissant.

Exercice avec "pour"

On reprend kla suite de nombres de l'exemple précédent (on commence par 5, et où l'obtient le nombre suivant en multpliant le précédent par 2 et en enlevant 1).

On veut calculer la valeur obtenue à la 25000-ème étape.

L'algorithme est le suivant :

Début

> $u \leftarrow 5$
> **Pour** *n* de 0 jusqu'à 25000 :
>
> > $u \leftarrow 2 \times u - 1$
>
> **Fin Pour**
> **afficher** *u*
> *cela peut être une bonne idée d'afficher n aussi pour vérifier que l'on a bien ce que l'on veut*

Fin

Programmer cet algorithme, on utilisera la syntaxe de l'exemple ci-dessus (le cours est très succint : VI 7).

Une remarque sur les booléens dans les itératives et conditionnelles.

Les instructions itératives et conditionnelles évaluant des booléens, on ne code pas ceci, qui est un pléonasme:

```
bool = True
if bool == True:
    ...
```

Mais cela:

```
bool = True
if bool :
    ...
```

Frederic Mandon (mailto:frederic.mandon@ac-montpellier.fr), Lycée Jean Jaurès - Saint Clément de Rivière - France (2015-2017)