Révisions

PSI 2 - Lycée Kléber

17 septembre 2018

1. Représentation des nombres
   - Entiers positifs
   - Entiers relatifs
   - Réels
   - Booléens
2. Objets listes et chaînes de caractères
   - Listes
   - Chaînes de caractères
3. Boucles
   - Boucles For
   - Boucles while (tant que)
4. Conditionnelles
   - Instructions

Base de numération

Un entier positif N est représenté en base b (entier supérieur à 1) par :

Représentation d'un entier dans une base

$$N = \sum_{n=0}^{+\infty} x_n b^n,$$

où les $x_n \in [\![0, b-1]\!]$ et sont nuls à partir d'un certain rang.

Base de numération

Un entier positif N est représenté en base b (entier supérieur à 1) par :

Représentation d'un entier dans une base

$$N = \sum_{n=0}^{+\infty} x_n b^n,$$

où les $x_n \in [\![0, b-1]\!]$ et sont nuls à partir d'un certain rang.
On écrit plutôt :

$$N = \overline{x_k x_{k-1} \dots x_1 x_0}^b.$$

Base de numération

Représentation d'un entier dans une base

$$\overline{3742}^{8} = 3 \times 8^3 + 7 \times 8^2 + 4 \times 8^1 + 2 \times 8^0 = 2018.$$

Base de numération

Représentation d'un entier dans une base

$$\overline{3742}^{8} = 3 \times 8^3 + 7 \times 8^2 + 4 \times 8^1 + 2 \times 8^0 = 2018.$$

Mieux (Hörner) : $\overline{3742}^{8} = ((3 \times 8 + 7) \times 8 + 4) \times 8 + 2.$

Base de numération

Représentation d'un entier dans une base

$$\overline{3742}^{8} = 3 \times 8^3 + 7 \times 8^2 + 4 \times 8^1 + 2 \times 8^0 = 2018.$$

Mieux (Hörner) : $\overline{3742}^{8} = ((3 \times 8 + 7) \times 8 + 4) \times 8 + 2$.

Intérêt : on a un algorithme de décomposition par divisions successives.

Base de numération

Exercice 1

Décomposer 2018 en base 2 puis en base 16.

Base de numération

Exercice 1

Décomposer 2018 en base 2 puis en base 16.

$$2018 = \overline{11111100010}^{2}$$

Base de numération

Exercice 1

Décomposer 2018 en base 2 puis en base 16.

$$2018 = \overline{11111100010}^{2}$$
$$2018 = \overline{7e2}^{16}$$

Codage en complément à 2

Principe

Dans ce codage sur n bits des entiers de $[\![-2^{n-1}, 2^{n-1}-1]\!]$:

- le code binaire d'un entier $k \in [\![0, 2^{n-1}-1]\!]$ est sa représentation binaire ;
- le code binaire d'un entier $k \in [\![-2^{n-1}, -1]\!]$ est la représentation binaire de $k + 2^n = 2^n - |k|$.

Codage en complément à 2

Principe

Dans ce codage sur n bits des entiers de $[\![-2^{n-1}, 2^{n-1}-1]\!]$:

- le code binaire d'un entier $k \in [\![0, 2^{n-1}-1]\!]$ est sa représentation binaire ;
- le code binaire d'un entier $k \in [\![-2^{n-1}, -1]\!]$ est la représentation binaire de $k + 2^n = 2^n - |k|$.

Remarque : on distingue facilement par ce codage les entiers de signe positifs (leur premier bit est 0) des entiers de signe négatif (leur premier bit est 1). On dit que le premier bit est le bit de signe.

Codage 8 bits

Sur 8 bits, on code les entiers relatifs de -128 à $+127$.

Codage 8 bits

Sur 8 bits, on code les entiers relatifs de -128 à $+127$.
$-128 = \overline{10000000}^2$

Codage 8 bits

Sur 8 bits, on code les entiers relatifs de -128 à $+127$.

$-128 = \overline{10000000}^2$

$-1 = \overline{11111111}^2$

Codage 8 bits

Sur 8 bits, on code les entiers relatifs de -128 à $+127$.

$-128 = \overline{10000000}^2$

$-1 = \overline{11111111}^2$

$127 = \overline{01111111}^2$

Codage 8 bits

Sur 8 bits, on code les entiers relatifs de -128 à $+127$.

$-128 = \overline{10000000}^2$

$-1 = \overline{11111111}^2$

$127 = \overline{01111111}^2$

Exercice 2

Écrire et additionner en complément à 2 sur 8 bits :

Codage 8 bits

Sur 8 bits, on code les entiers relatifs de -128 à $+127$.

$-128 = \overline{10000000}^2$

$-1 = \overline{11111111}^2$

$127 = \overline{01111111}^2$

Exercice 2

Écrire et additionner en complément à 2 sur 8 bits :

120, -120, $-45 + 17$, $-45 + 60$.

Codage 8 bits

Sur 8 bits, on code les entiers relatifs de -128 à $+127$.
$-128 = \overline{10000000}^2$
$-1 = \overline{11111111}^2$
$127 = \overline{01111111}^2$

Exercice 2

Écrire et additionner en complément à 2 sur 8 bits :
120, -120, $-45 + 17$, $-45 + 60$.
$120 = \overline{01111000}^2$,

Codage 8 bits

Sur 8 bits, on code les entiers relatifs de -128 à $+127$.
$-128 = \overline{10000000}^2$
$-1 = \overline{11111111}^2$
$127 = \overline{01111111}^2$

Exercice 2

Écrire et additionner en complément à 2 sur 8 bits :
120, -120, $-45+17$, $-45+60$.
$120 = \overline{01111000}^2$,$-120 = \overline{10001000}^2$,

Codage 8 bits

Sur 8 bits, on code les entiers relatifs de -128 à $+127$.

$-128 = \overline{10000000}^2$

$-1 = \overline{11111111}^2$

$127 = \overline{01111111}^2$

Exercice 2

Écrire et additionner en complément à 2 sur 8 bits :

$120,\ -120,\ -45+17,\ -45+60.$

$120 = \overline{01111000}^2, -120 = \overline{10001000}^2, -45 = \overline{11010011}^2,$

Codage 8 bits

Sur 8 bits, on code les entiers relatifs de -128 à $+127$.
$-128 = \overline{10000000}^2$
$-1 = \overline{11111111}^2$
$127 = \overline{01111111}^2$

Exercice 2

Écrire et additionner en complément à 2 sur 8 bits :
120, -120, $-45+17$, $-45+60$.
$120 = \overline{01111000}^2$,$-120 = \overline{10001000}^2$,$-45 = \overline{11010011}^2$,
$17 = \overline{00010001}^2$,

Codage 8 bits

Sur 8 bits, on code les entiers relatifs de -128 à $+127$.

$-128 = \overline{10000000}^2$

$-1 = \overline{11111111}^2$

$127 = \overline{01111111}^2$

Exercice 2

Écrire et additionner en complément à 2 sur 8 bits :

$120,\ -120,\ -45+17,\ -45+60.$

$120 = \overline{01111000}^2, -120 = \overline{10001000}^2, -45 = \overline{11010011}^2,$

$17 = \overline{00010001}^2, \overline{11100100}^2 = -28,$

Codage 8 bits

Sur 8 bits, on code les entiers relatifs de -128 à $+127$.

$-128 = \overline{10000000}^2$

$-1 = \overline{11111111}^2$

$127 = \overline{01111111}^2$

Exercice 2

Écrire et additionner en complément à 2 sur 8 bits :

$120,\ -120,\ -45+17,\ -45+60.$

$120 = \overline{01111000}^2, -120 = \overline{10001000}^2, -45 = \overline{11010011}^2,$

$17 = \overline{00010001}^2, \overline{11100100}^2 = -28, 60 = \overline{00111100}^2,$

Codage 8 bits

Sur 8 bits, on code les entiers relatifs de -128 à $+127$.
$-128 = \overline{10000000}^2$
$-1 = \overline{11111111}^2$
$127 = \overline{01111111}^2$

Exercice 2

Écrire et additionner en complément à 2 sur 8 bits :
120, -120, $-45+17$, $-45+60$.
$120 = \overline{01111000}^2$,$-120 = \overline{10001000}^2$,$-45 = \overline{11010011}^2$,
$17 = \overline{00010001}^2$,$\overline{11100100}^2 = -28$,$60 = \overline{00111100}^2$,
$\overline{00001111}^2 = 15$.

Norme IEEE 754

Le stockage des réels en norme IEEE 754 se fait sous la forme scientifique binaire :

$$x = \pm\overline{1, x_{-1}x_{-2}\ldots}^{2} \times 2^{k}.$$

Norme IEEE 754

Le stockage des réels en norme IEEE 754 se fait sous la forme scientifique binaire :

$$x = \pm\overline{1,x_{-1}x_{-2}\ldots}^{2} \times 2^{k}.$$

Représentation sur 64 bits (Python)

1 bit de signe ; 11 bits pour l'exposant k ; 52 bits de mantisse.

Les 11 bits de l'exposant représentent $k + 1023$.

$$-1022 \leqslant k \leqslant 1023$$

Norme IEEE 754

Exercice 3

Donner la représentation du réel :
3.0

Norme IEEE 754

Exercice 3

Donner la représentation du réel :

3.0 0 10000000000 1000000000000...0000

Norme IEEE 754

Exercice 3

Donner la représentation du réel :

3.0 0 10000000000 1000000000000...0000

0.3

Norme IEEE 754

Exercice 3

Donner la représentation du réel :

3.0	0 10000000000 1000000000000...0000
0.3	0 01111111101 0011000110011...0011

Exercice 4

Quelle est le plus grand réel ainsi codé ?

Norme IEEE 754

Exercice 3

Donner la représentation du réel :

3.0 0 10000000000 1000000000000...0000

0.3 0 01111111101 0011000110011...0011

Exercice 4

Quelle est le plus grand réel ainsi codé ? $\sim 2^{1024} \simeq 1,8 \times 10^{308}$

Quelle est la précision de la mantisse ?

Norme IEEE 754

Exercice 3

Donner la représentation du réel :

3.0 0 10000000000 1000000000000...0000

0.3 0 01111111101 0011000110011...0011

Exercice 4

Quelle est le plus grand réel ainsi codé ? $\sim 2^{1024} \simeq 1,8 \times 10^{308}$

Quelle est la précision de la mantisse ? $2^{-52} \simeq 10^{-16}$

True et False

Les booléens sont utilisés pour vérifier une condition logique. Ils constituent un type spécial avec deux valeurs possibles : *True* (si la condition est vrai) et *False* (si la condition est fausse).

» 5>6
False
» a=9>=8
» a
True

Affectation et test d'égalité

Rôle du signe =

Quelle est la différence entre les deux lignes suivantes :

» a=5

» a==9

Affectation et test d'égalité

Rôle du signe =

Quelle est la différence entre les deux lignes suivantes :

» a=5

» a==9

La première *affecte* la valeur 9 à la variable « a ».

Affectation et test d'égalité

Rôle du signe =

Quelle est la différence entre les deux lignes suivantes :

» a=5

» a==9

La première *affecte* la valeur 9 à la variable « a ».
La seconde est un test qui compare le contenu de la variable « a » à la valeur 9. Le résultat est un booléen.

Définition des listes

Une liste est un n-uplet dont on peut changer les valeurs de composantes.

En Python, on les construits entre crochets ("[ ]") et on sépare les différents termes par des virgules :

```
» L=[1,5,9,6]
» L
[1, 5, 9, 6]
```

Définition des listes

Une liste est un n-uplet dont on peut changer les valeurs de composantes.

En Python, on les construits entre crochets ("[ ]") et on sépare les différents termes par des virgules :

```
» L=[1,5,9,6]
» L
[1, 5, 9, 6]
```

On peut récupérer un terme d'une liste en utilisant des indices.
Attention : La numérotation des indices commence à 0.

```
» L[0]
1
» L[1]
5
```

Opérations sur les listes

Concaténation

La concaténation de deux listes correspond à leur adjonction (mises bout à bout) et utilise le symbole « + » dans Python :

» L1=[1,5,9,6]
» L2=[7,8]
» L1+L2
[1, 5, 9, 6, 7, 8]

Opérations sur les listes

Longueur

La longueur d'un liste est donnée par l'opérateur len :

```
» L1=[1,5,9,6]
» len(L1)
4
```

Opérations sur les listes

Slicing

La sélection (slicing) d'une liste se fait à l'aide du signe « : ».
On note L[i1 :i2], la sélection en partant du i1 ème terme jusqu'au (i2 - 1) ème termes de la liste :

» L1=[1,5,9,6]
» L1[1 :2]
[5]

Chaînes de caractères

Les chaines de caractères (ou strings en anglais) permettent de repré- senter du texte.
On les note entre simple (') ou double guillemets ("").
Une chaine de caractères est considéré comme un n-uplets de caractères.

Chaînes de caractères

On peut donc utiliser les propriétés des listes :

- pour avoir accès à un caractère :

» c='Bonjour tout le monde'
» c[3]
'j'

Chaînes de caractères

On peut donc utiliser les propriétés des listes :

- pour avoir accès à un caractère :

» c='Bonjour tout le monde'
» c[3]
'j'

- pour donner la longueur d'une chaine caractères (en comptant les espaces) :

» c='Bonjour tout le monde'
» len(c)
21

Chaînes de caractères

- pour concaténer des chaines de caractères :

» c1='Bonjour '
» c2='tout le monde'
» c1+c2
'Bonjour tout le monde'

- pour sélectionner une partie de chaine de caractères :

» c='Bonjour tout le monde'
» c[0 :7]
'Bonjour'

Définition

Une boucle *for* permet de répéter une même instruction un nombre fini et déterminé à l'avance de fois.
Pour compter le nombre d'itérations, on utilise une variable de type entier et la fonction range (itérateur).

Exercice

Que renvoient les instructions suivantes ?

```
for i in range(10):
    print(i**2)

for n in range (5,10):
    print (n)

for n in range (0,10,2):
    print (n)

for n in range (10,0,-1):
    print (n)
```

Listes en compréhension

Il est également possible de créer une liste en compréhension, c'est à dire de la spécifier directement à l'aide d'une boucle for.

Exemple d'une liste en compréhension

```
L=[n**2 for n in range(10)]
```

Listes en compréhension

Il est également possible de créer une liste en compréhension, c'est à dire de la spécifier directement à l'aide d'une boucle for.

Exemple d'une liste en compréhension

```
L=[n**2 for n in range(10)]
```

```
>>> L
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

Définition

Structure

```
while condition :
        instructions
```

Une boucle while permet de répéter une même instruction tant qu'une condition reste vrai.

Définition

Structure

```
while condition :
        instructions
```

Une boucle while permet de répéter une même instruction tant qu'une condition reste vrai.
Elle est souvent appropriée pour répéter des instructions sans connaître au préalable le nombre de fois.

Définition

Structure

```
while condition :
        instructions
```

La condition qui figure après le mot clé while est évaluée à l'entrée de la boucle :

- si elle est vrai (booléen égal à True) alors on entre de la boucle et l'instruction est exécutée ;

Définition

Structure

```
while condition :
        instructions
```

La condition qui figure après le mot clé while est évaluée à l'entrée de la boucle :

- si elle est vrai (booléen égal à True) alors on entre de la boucle et l'instruction est exécutée ;
- si elle est fausse (booléen égal à False) alors on sort de la boucle et l'instruction n'est pas exécutée.

Difficulté de la boucle while

Attention

On peut très facilement créer une boucle while infinie.

Difficulté de la boucle while

Attention

On peut très facilement créer une boucle while infinie.
Il est nécessaire de prouver la terminaison d'une boucle en vérifiant que la condition liée à cette boucle devient forcément fausse au bout d'un moment.

Difficulté de la boucle while

Exemple (mauvais)

Que manque-t-il à ces lignes ?

```
n=1
while n < 10:
    print(n**3)
```

Difficulté de la boucle while

Exemple (mauvais)

Que manque-t-il à ces lignes ?

```
n=1
while n < 10:
    print(n**3)
```

Exemple (bon)

```
n=1
while n < 10:
    print(n**3)
    n+=1
```

Difficulté de la boucle while

Mésaventure du lecteur MP3 Zune

Le 30 décembre 2008 à minuit, tous les appareils se sont arrêtés de fonctionner. Les batteries se vidaient et l'appareil s'éteignait.

Difficulté de la boucle while

Mésaventure du lecteur MP3 Zune

Le 30 décembre 2008 à minuit, tous les appareils se sont arrêtés de fonctionner. Les batteries se vidaient et l'appareil s'éteignait. Recharger la batterie restait sans effet.

Difficulté de la boucle while

Mésaventure du lecteur MP3 Zune

Le 30 décembre 2008 à minuit, tous les appareils se sont arrêtés de fonctionner. Les batteries se vidaient et l'appareil s'éteignait.
Recharger la batterie restait sans effet.
Il fallait attendre le lendemain pour que tout redevienne normal.

Difficulté de la boucle while

Mésaventure du lecteur MP3 Zune

```
def Bissextile(an):
    if an%100==0:
        return (an//100)%4==0
    else:
        return an%4==0

while jour > 365:
    if Bissextile(an) == True:
        if jour > 366:
            jour-=366
            an+=1
    else:
        jour-=365
        an+=1
```

Exercice

Suite

On considère la suite définie par son premier terme $u_0 = 1$ et la relation de récurrence $u_{n+1} = 0{,}5u_n + 10$.
On établit que sa limite est 20.
Écrire un programme qui pour une précision e (exemple $e = 0{,}00001$) retourne l'indice n du premier terme pour lequel $|u_n - 20| < e$.

Exercice

Suite

```
e=0.00001 # erreur acceptable
n=0       # indice de l'élément de suite
u=1       # valeur initiale de la suite
while abs(u-20) >= e: # condition de parcours
    u=.5*u+10         # calcul de l'élément suivant
    n+=1              # incrément de l'indice
print(n)              # affiche la valeur de l'indice
    lors de la sortie de la boucle
```

Définition

Une instruction conditionnelle permet d'exécuter au choix une instruction en fonction de la valeur d'une condition logique. On utilise pour cela la commande *if*.

Définition

Syntaxe

La syntaxe est la suivante :

- *if* est seul : si la condition évaluée qui suit *if* est vrai alors l'exécution de l'instruction indentée juste en dessous est exécutée.

Définition

Syntaxe

La syntaxe est la suivante :

- *if* est seul : si la condition évaluée qui suit *if* est vrai alors l'exécution de l'instruction indentée juste en dessous est exécutée.

```
N=10
if N > 5:
    print(N)
if N < 15:
    print(N**2)
if N < 9:
    print(N**3)
```

Définition

Syntaxe

La syntaxe est la suivante :

- *if* est suivi de *else* : si la condition évaluée qui suit *if* est vraie alors l'exécution de l'instruction indentée juste en dessous est exécutée. Si elle est fausse alors l'instruction qui suit *else* est exécutée.

Définition

Syntaxe

La syntaxe est la suivante :

- *if* est suivi de *else* : si la condition évaluée qui suit *if* est vraie alors l'exécution de l'instruction indentée juste en dessous est exécutée. Si elle est fausse alors l'instruction qui suit *else* est exécutée.

```
N=10
if N > 5:
    print(N)
else:
    print(N**2)
if N < 9:
    print(N**3)
```

Définition

Syntaxe

La syntaxe est la suivante :

- *if* est suivi d'une ou plusieurs instructions *elif* : le programme sélectionne la première condition vraie qui suit *if* et *elif* et exécute l'instruction associée. Si toutes les conditions sont fausses, alors le programme exécute l'instruction associée au *else*. À défaut de *else*, aucune instruction n'est exécutée.

Définition

Syntaxe

La syntaxe est la suivante :

• *if* est suivi d'une ou plusieurs instructions *elif* : le programme sélectionne la première condition vraie qui suit *if* et *elif* et exécute l'instruction associée. Si toutes les conditions sont fausses, alors le programme exécute l'instruction associée au *else*. À défaut de *else*, aucune instruction n'est exécutée.

```
N=3
if N > 5:
    print(N)
elif N < 15:
    print(N**2)
elif N < 9:
    print(N**3)
```

Exercice

Trinôme

Écrire un programme qui demande trois nombres a, b et c et qui donne les racines du trinôme $ax^2 + bx + c$.

Exercice

Trinôme

```
a=int(input('a = ? '))
b=int(input('b = ? '))
c=int(input('c = ? '))

delta=b**2-4*a*c

if delta > 0:
    print('x1 = ',(-b-np.sqrt(delta))/2/a,'\nx2 = ',(-b
        +np.sqrt(delta))/2/a)
elif delta =0:
    print('x1 = x2 = ',-b/2/a)
else:
    print('pas de solutions !')
```

Exercice

Jeu de math

L'ordinateur choisit un nombre entier au hasard entre 100 et 400.
Vous êtes chargés de deviner ce nombre en moins de 10 coups.
Il vous répond par « C'est plus » ou « C'est moins » puis « Perdu ! » ou « Gagné en ... coups ! »

Exercice

Jeu de math

```
import random as rd # importation de la bibliothèque

NombreOrdi=rd.randint(100,400) # génération du nombre à
    trouver
NombreJoueur=0 # Nombre choisi par le joueur initialisé
    à 0
Coup=1 # Nombre de coups joués
```

Exercice

Jeu de math

```
while NombreJoueur != NombreOrdi and Coup < 11:
    NombreJoueur=int(input('Quelle est votre
        proposition ? '))
    if NombreJoueur > NombreOrdi:
        print('Moins')
    elif NombreJoueur < NombreOrdi:
        print('Plus')
    Coup+=1

if Coup==11:
    print('Perdu !')
else:
    print('Gagné en ',Coup-1,' coups !')
```