Programmation orientée objet avec Python

Mars 2021

académie Versailles
GRETA
DES HAUTS-DE-SEINE
FORMEZ-VOUS À DEMAIN
RÉGION ACADÉMIQUE ÎLE-DE-FRANCE
LUDIKSCIENCES

Entiers

- integer, int, <class 'int '>
- Littéraux : -1, 0, 12847
- Les littéraux sont évalués et affichés (« REPL »)
- Opérateurs : + - * / (opérations sur les nombres entiers)
  - Noter que +- désigne le signe des nombres entiers relatifs (opérateurs unaires) mais aussi les opérations d'addition et de soustraction (opérateurs binaires).
  - // division entière
  - % modulo (reste dans la division entière)
  - ** élévation à la puissance
  - Priorité des opérateurs - essayer 1+2*3
  - Les produits et les divisions sont calculés avant les additions et les soustractions
    - */ sont prioritaires sur +-
    - L'élévation à la puissance est prioritaire sur * et /
  - Quand on rencontre des opérateurs de même poids, c'est l'ordre donné par la lecture de gauche à droite qui s'applique.
  - Pour contrarier ces comportements de base, utiliser des parenthèses
    - Ce qui est entre parenthèse est évalué avant ce qui est autour.
- Ecriture binaire (en base 2), octale (en base 8), hexadécimale (en base 16)
  - 0b1010
  - 0o12
  - 0xA

Classeur Jupyter Entiers.ipynb

LUDIKSCIENCES

Les chaînes de bits

- & et binaire
- | ou binaire
- ^ ou exclusif binaire
- ~ non binaire, complémentation
- << décalage à gauche, ajout de 0 à droite
- >> décalage à droite,

Il n'y a pas de rotation à droite/à gauche

Exercice : Que valent ?

1&2

1|2

1^2

2^2

~2

2>>1, 2>>2

2<<1, 2<<2

Expliquer !

Classeur Jupyter Bits.ipynb

Table de vérité de ET		
a	b	a ET b
0	0	0
0	1	0
1	0	0
1	1	1

Table de vérité de OU		
a	b	a OU b
0	0	0
0	1	1
1	0	1
1	1	1

Table de vérité de XOR (OU exclusif)		
a	b	a XOR b
0	0	0
0	1	1
1	0	1
1	1	0

Priorités : ">>, <<" >NOT>AND>XOR>OR

Binaires entiers négatifs

Un nombre entier sur 4 bits peut prendre 16 valeurs (2^4) soit de 0 à 15

Comment encoder les nombres négatifs ?

Le bit de poids le plus fort représente le signe + (0) ou – (1)

Reste 2^3 = 8 valeurs possibles pour chaque signe

0111	7
0110	6
0101	5
0100	4
0011	3
0010	2
0001	1
0000	0
1000	0
1001	-1
1010	-2
1011	-3
1100	-4
1101	-5
1110	-6
1111	-7

On doit pouvoir additionner deux nombres : **2+(-2) = 0**

```
  1
 0010
+1010
----------
=1100 => -4
```

		Complément
1111	-1	16-15
1110	-2	16-14
1101	-3	16-13
1100	-4	16-12
1011	-5	16-11
1010	-6	16-10
1001	-7	16-9
1000	-8	16-8
0111	7	
0110	6	
0101	5	
0100	4	
0011	3	
0010	2	
0001	1	
0000	0	

```
 1 1
 0010
+1110
----------
=0000 => 0
```

Comment passer de 2 à –2 ?

Le complément à 2
=
Complément à 1
+ Ajouter 1 aux bits inversés

```
2       0010
NOT 2   1101
+1      0001
        ----------
=       1110 => -2
NOT-2   0001
+1      0001
        ----------
=       0010 => 2
```

LUDIKSCIENCES

Booléens

- Boolean, bool, <class ‘bool’>
- Littéraux ou constantes : True, False
- Observer que Python3 ne connaît pas true et false : la casse (distinction majuscules/minuscules) est importante
- Opérateurs : not (opérateur unaire) and, or(opérateur binaire)
- Construire les tables de vérité de ces opérateurs
- Priorité des opérateurs : not est prioritaire sur and qui est prioritaire sur or

Classeur Jupyter Booléens.ipynb

LUDIKSCIENCES

Flottants

- On parle de nombres réels, mais les ordinateurs représentent des nombres rationnels uniquement
  - mantisse exposant
  - Par exemple +3,14 est représenté par + 0.314 E+1
- Nombres flottants, float, <class ‘float’>
- C’est le . qui sert de marque décimale (et non la virgule)

- Opérateurs : +, -, *, /, **

- Observer le résultat de 2.3 *10

Classeur Jupyter Flottants.ipynb

LUDIKSCIENCES

Représentation des nombres flottants

- Mantisse
  Signe + ou -
  Valeur absolue 0.xxxxxx
- Exposant
  Signe+ -
  Valeur absolue xxx

Classeur Jupyter Flottants.ipynb

LUDIKSCIENCES

Complexes

- Nombres imaginaires a+b*j où a est la partie réelle et b, la partie imaginaire

  $j^2 = -1$
- complex, <class ‘complex’>
- Evaluer 3 *(2+4j)

Classeur Jupyter Complexes.ipynb

LUDIKSCIENCES

Chaînes de caractères

- String, <class ‘str’>
- 'toto' ou « toto », utile pour "J'ai dit", 'Il a dit "Merci"', ou encore en échappant l'apostrophe 'J\'ai dit'
- Retour à la ligne " Il était \n une fois "
- Tabulation "Prix \t 3,50 Euros"
- Une chaine sur plusieurs lignes

```
"""Maître corbeau sur un arbre perché
Tenait en son bec un fromage
Maître renard par l'odeur alléché
Lui tint à peu près ce langage
 """
```

- Opérateur : + concaténation * répétition

```
"Salut"+"les amis"
"*"*10
```

Classeur Jupyter Strings.ipynb

LUDIKSCIENCES

Séquences d'échappement

\fin de ligne	
\\	backslash
\', \"	quotes 1x et 2x
\n	Line feed
\r	Carriage Return
…	…
\t	Tab
\ooo	octal
\xhh	hexa
\uxxxx	unicode 16 bits – 4 hex
\Uxxxxxxxx	unicode 32 bits – 8 hex
\N{name}	nom du point de code

LUDIKSCIENCES

Chaînes d'octets

- *bytes*
- 8 bits
- octet = 1 valeur entre 0 et 255 écrite en hexa, par ex \xff ou en octal \777

LUDIKSCIENCES

Unicode

- Pour représenter les caractères, les octets (*bytes*) sont privilégiés
  - Caractères de la machine à écrire (moins de 100) + 32 caractères non imprimables utilisés pour les transmissions (accessibles au clavier avec la touche CTL…)
  - ASCII (America Standard Coding for Information Interchange) et d'autres (passés sous silence)
    - représentés sur 7 bits
  - Compléter l'emploi des 256 positions pour mettre les caractères accentués de chaque langue
    - ISO 8859-1 ou ISO Latin-1 (ajout de 96 symboles supplémentaires)
    - Windows CP1252 (ajout de 27 symboles supplémentaires)
- Les octets sont donc saturés.
  - Mis bout à bout, les alphabets internationaux demandent ensemble beaucoup plus que 255 positions.
  - Solution : invention d'Unicode, capacité de 1,1Millions de *code points*.
  - 10% sont utilisés pour le moment.
  - Unicode se confond avec l'ASCII pour les premières positions et l'étend.

LUDIKSCIENCES

UTF-8

- Pour représenter un point de code par des octets, il faut un codage
  - *encoding*

- UTF-8 est un codage Unicode
  - Pour les points de code <128, UTF-8 est identique au code ASCII
  - Pour le points de code >=128, UTF-8 utilise une séquence d'octets (2, 3 ou 4)
  - D'autres codages existent
    - UTF-16, …

- Avantages
  - Tous les points de code sont représentés
  - Pas de place perdue,
  - Pas de perturbation des transmissions avec 0 dans les chaînes de caractères qui bloqueraient les strcpy()
  - Compatible avec ASCII
  - Resynchronisation efficace après un caractère erroné

LUDIKSCIENCES

UTF-8 et Python

Python 3 est nativement UTF-8 partout

- dans le source
- dans la représentation du type str

Python2 avait un type distinct pour les chaînes Unicode

- un type str qui confondait la notion de caractère avec celle d'octet
- des fonctions de conversions et des exceptions perturbantes pour le programmeur

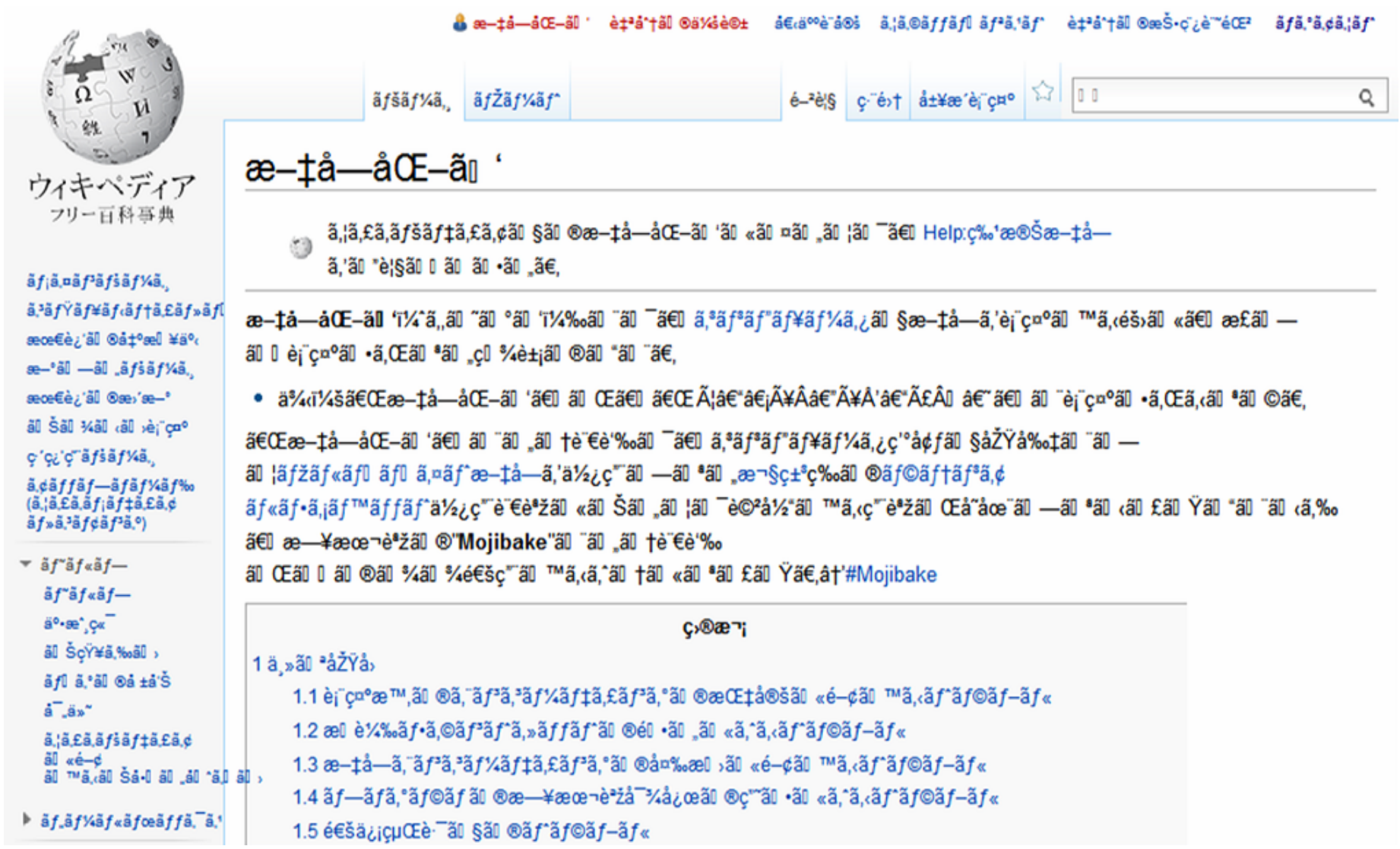
En Python 3, str et bytes sont des classes (types) différents

=> tester les fonctions chr(97) et ord('a')

Recommandations

1. Dans votre code privilégier UTF-8
2. Dans les interfaces avec l'extérieur, identifier le codage utilisé (le mettre en doute…) et faire la conversion vers UTF-8 le plus tôt possible (et dans l'autre sens, le plus tard)
3. Dans vos tests, inclure des caractères Unicode exotiques pour éviter le *mojibake*

LUDIKSCIENCES



By Wikimedia Foundation - Wikimedia Foundation, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=17303619

LUDIKSCIENCES

Names

Identificateurs – pour Python ce sont des noms

- Commencent par une lettre Unicode ou par un _
- Continuent par des lettres, des _ ou des chiffres

Utilisés à chaque fois qu'il nous faut baptiser

- une variable, une fonction, une constante

Eviter les collisions avec les mots-réservés du langage Python3

Signalées par les outils

Pas de confusion possible avec les littéraux

LUDIKSCIENCES

Variable

Exemple: x

Constitue la *référence* d'un objet qui stocke la valeur de x

Python3 distingue les objets modifiables (*mutable*) et non modifiables (*immutable*)

Dans le cas des objets non modifiables, dire que la variable x référence *la valeur* de x est acceptable.

Dans le cas d'objets modifiables, la variable peut faire référence à *un objet* (toujours le même) mais la valeur de cet objet a pu changer.

LUDIKSCIENCES

Variable _

dans l'interpréteur, _ désigne la valeur de la dernière expression évaluée

dans le code, désigne une variable que l'on souhaite ignorer ou une variable que l'on ne se donne pas la peine de baptiser

Classeur Jupyter Underscore.ipynb

LUDIKSCIENCES

Affectation

Le signe égal ne signifie pas l'égalité mais se lit plutôt comme "prend la valeur de" ou bien "reçoit" (*assignment*)

Dans certains langages, on l'écrit comme une flèche ←

L'expression à droite du signe = est évaluée et le résultat est conservé dans un objet en mémoire

L'expression à gauche du signe égal est la plupart du temps le nom qui sert d'étiquette pour l'objet

```
x = 1          s = 3.14 * 12**2
```

L'affectation crée un lien entre un nom et un objet qui a une valeur

affectation augmentée

x += 1 équivaut à x=x+1 (x prend l'ancienne valeur de x à laquelle on ajoute 1)

x -= 2

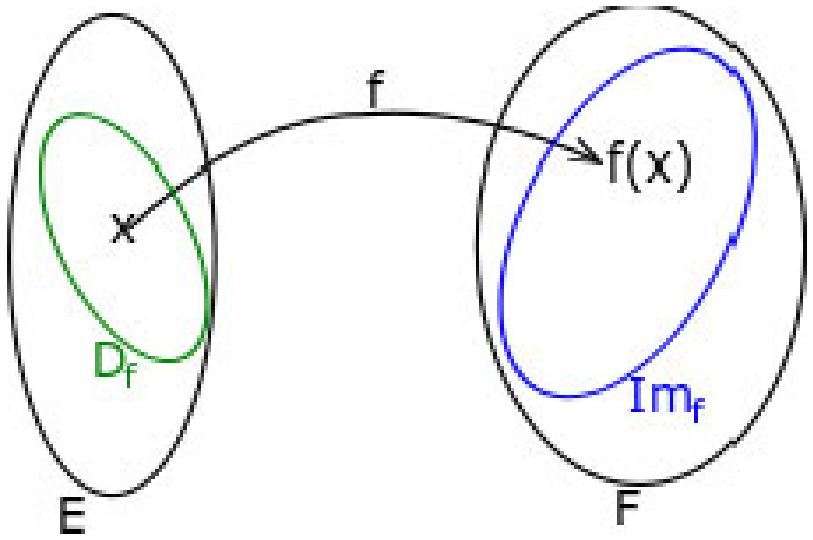
affectation parallèle d'un tuple : x, y = 0.157 , True

Que signifient x = y = 0 x, y = y, x ?

LUDIKSCIENCES

Fonctions en math

Une fonction	f
Un ensemble de départ	E
Un ensemble d'arrivée	F
Domaine de définition	D(f)
Ensemble image	Im(f)



Tout élément de l'ensemble de départ a ***au plus*** une image par f

L'image de x par f s'écrit : f(x)

Se lit "f de x".

LUDIKSCIENCES

Fonctions en informatique

Un algorithme qui calcule une (1) valeur

Cette définition est restrictive.
Python –comme d'autres langages - étend le concept à des algorithmes qui calculent plus d'une valeur

périmètre, surface = cercle (rayon)

ou aucune valeur

print ("x = ", x)

Dans d'autres langages, on parle de procédures plutôt que de fonctions
Tout ce qui n'est pas le calcul de la valeur de la fonction en un point et qui affecte l'environnement d'exécution est un "effet de bord" - *side effect*.
Certaines fonctions sont surtout utiles par leur effet de bord

cas de la fonction print

En dehors du domaine de définition, le résultat du calcul est imprédictible (et souvent catastrophique TSVP)

LUDIKSCIENCES

LUDIKSCIENCES

Fonctions en Python

Prédéfinies

Fonctions de bibliothèques

Fonctions que l'on définit soi-même…

LUDIKSCIENCES

Fonctions prédéfinies

__import__	abs	all	any	ascii	bin
bool	breakpoint	bytearray	bytes	callable	chr
classmethod	compile	complex	delattr	dict	dir
divmod	enumerate	eval	exec	filter	float
format	frozenset	getattr	globals	hasattr	hash
help	hex	id	input	int	isinstance
issubclass	iter	len	list	locals	map
max	memoryview	min	next	object	oct
open	ord	pow	print	property	range
repr	reversed	round	set	setattr	slice
sorted	staticmethod	str	sum	super	tuple
type	vars	zip			

Commentaires

Les différentes écritures de commentaires en Python

- Ligne qui commence par un #
- Texte au-delà du # en fin de ligne
- Texte sur 1 ou plusieurs lignes entre délimiteurs """

Utiles au développeur, au lecteur, pour la maintenance, pour la constitution de documentation

Ignorés par l'interpréteur et le compilateur

LUDIKSCIENCES

Types

Types élémentaires

- int,
- float
- bool,
- complex,
- str,
- byte

Pour Python, une variable a un type à un instant donné. On doit l'utiliser de manière cohérente. Ce type peut changer. Python est un langage dynamique.

Python 3.6 ajoute le *type hint* (une indication du type attendu)

a: int

Des outils (pycharm) vérifient alors le type à la compilation.

LUDIKSCIENCES

Transtypage

type cast

int(x) convertit x en int

float(x)

str(x)

bool(x)

complex(x)

Utile dans les entrées sorties

type(input("Entrez un entier"))

int(input("Entrez un entier"))

LUDIKSCIENCES

Opérateurs de comparaison

égalité ==

inférieur <, <=

supérieur >, >=

différent !=

Quel est le type de l'expression a==b ?

Essayer avec des int, bool, str, float

Attention au test d'égalité sur les nombres flottants.
Quelle est la bonne méthode pour tester cette égalité ?

LUDIKSCIENCES

Expression

Une expression combine

- ( )
- opérateurs
- y compris, opérateurs de comparaisons
- littéraux et variables
- applications de fonctions
- composition de fonctions

Une valeur (et un type)

Quels sont les types de

- 3/2, 10/5, 3.14 * 12
- a=3, b="3", quel est le type de a==b, not (a==b), a!=b

LUDIKSCIENCES

Exercices

Rien

Faire un script Python rien.py qui ne fait rien comme ci-contre

```
$ python rien.py
$
```

Bonjour

Faire un script Python bonjour.py qui affiche "Bonjour"

```
$ python bonjour.py
Bonjour
$
```

Addition

Faire un script Python qui lit deux entiers, i1 et i2, au clavier et qui affiche leur somme sur l'écran

```
$ python addition.py
Entrez i1 : 89
Entrez 12 : 12
i1+i2 = 101
$
```

LUDIKSCIENCES

Les mots-clés, mots réservés de Python3

False	None	True	and	as	assert
break	class	continue	def	del	elif
else	except	finally	for	from	global
if	import	in	is	lambda	nonlocal
not	or	pass	raise	return	try
while	with	yield			

```
import keyword
print(keyword.kwlist)
```

LUDIKSCIENCES

Les mots-clés déjà vus

False	None	True	and	as	assert
break	class	continue	def	del	elif
else	except	finally	for	from	global
if	import	in	is	lambda	nonlocal
not	or	pass	raise	return	try
while	with	yield			

LUDIKSCIENCES

Indentation

Les autres langages utilisent le principe des parenthèses pour exprimer

- la séquence
- l'inclusion
- L'indentation y est optionnelle
- *pretty print*

```
Pascal: begin action1;… actionN end
C, Java, …: {a1 ; a2; …aN;}
Lisp: (cond (c1 a1) (c2 a2) (t a3))
Basic: IF… ENDIF – WHILE … WEND
```

Python se sert de l'indentation. Elle n'est pas optionnelle mais fait partie du langage.

La séquence est représentée par l'alignement sur une même verticale

```
import keyword
print(keyword.kwlist)
```

L'inclusion est représentée par un décalage vers la droite (indentation).

- le retour vers la gauche correspond à la fermeture de l'inclusion

```
Nombreux exemples à suivre
```

LUDIKSCIENCES

Tabulation

TAB = 4 espaces

Recommandation ne pas utiliser CTL+I, Ascii 9

Interdiction de mixer TAB = CTL+I et TAB = 4 espaces

LUDIKSCIENCES

Point-virgule

Si l'on tient à introduire plusieurs actions sur la même ligne, alors il faut les séparer par un point-virgule

```
import keyword ; print(keyword.kwlist)
```

Le point virgule en fin de ligne est contre-productif

=> ajout d'une instruction vide

LUDIKSCIENCES

Longueur des lignes

Une ligne de code ne doit pas être prolongée au-delà de la 72ème colonne

Pour continuer sur la ligne suivante , 2 possibilités

- un antislash figure comme dernier caractère : \
- un caractère ouvrant est présent et il n'est pas encore fermé : [({

LUDIKSCIENCES

If…else

Action conditionnelle

```
if condition :
    action1
    …
    actionN
suite
```

Les : sont indispensables

L'indentation, aussi

Inclusion d'une action ou d'une séquence d'actions

Essayer d'écrire sans indenter

```
if a!=0:
print(a)
```

Action alternative

```
if condition :
    action1
else:
    action2
suite
```

LUDIKSCIENCES

elif

Alternative complexe

else-if

Tester une condition après l'autre

Aller directement à la bonne clause

- switch de Java ou C
- case de Pascal
- goto calculé de Fortran
- n'existent pas en Python3
  - on fait autrement…

```
if condition1 :
    action1
elif condition2 :
    action2
elif condition…:
    …
else:
    actionN
suite
```

LUDIKSCIENCES

Condition

Condition = expression booléenne

```
if(a%2)==1 :    #si a est impair
```

Plus généralement

expression dont le résultat est interprété comme True ou False

Sont interprétés comme

False

False, 0, chaîne vide, None, autres structures vides (), [], {}, objets nuls

True

Tout le reste

```
if a%2 :        #si a est impair
    """
    il suffit de vérifier que le
    reste est non nul
    """
```

bool(expr) force la conversion de l'expression expr en booléen

```
if bool(expr)==True: # inélégant

if expr:          # Pépette contente!
```

LUDIKSCIENCES

Expression conditionnelle

Opérateur ternaire

```
val1 if cond else val2
```

- 3 opérandes, 2 mots clés
- 1 condition : cond
- La valeur si vrai : val1
- la valeur si faux : val2

Intérêt :

- concision
- possibilité de l'inclure une expression plus complexe

Exemple : le sup de deux nombres

```
print(a if a>= b else b)
```

LUDIKSCIENCES

Action

Ne rien faire

Affectation

Appel de fonction prédéfinie

Fonction importée

- Importer une fonction de bibliothèque

Fonction définie

- Définir une fonction

LUDIKSCIENCES

pass

Ne rien faire

```
…:
    pass
suite
```

no op

placeholder

prend la place d'une action dans un algorithme

souvent pour quelque chose qui reste à développer

LUDIKSCIENCES

import

importer une bibliothèque

```
import math
x1 = -(b-math.sqrt(delta))/(2*a)
```

importer une fonction d'une bibliothèque

```
from math import sqrt
x1 = -(b-sqrt(delta))/(2*a)
```

importer une fonction d'une bibliothèque en la baptisant avec un alias pour un usage plus simple

```
from math import sqrt as racine
x1 = -(b-racine(delta))/(2*a)
```

LUDIKSCIENCES

def

Définition d'une fonction

- son nom
- ses arguments "(…)"
- les ":"
- une docstring """ … """
- l'algorithme qui retourne la valeur

```
def f (x, a, b, c):
    """ calcule le trinôme du 2nd degré
        ax² + bx +c
    """
    t= a*x**2+b*x+c
    return t
```

Appel de la fonction

```
print (f(0, 1.2, 3, 4.5))
```

Distinction

- Définition de la fonction
  - paramètres formels ou "arguments"
- Appel de la fonction
  - paramètres effectifs ou "paramètres"

LUDIKSCIENCES

Don't Repeat Yourself

Plus de 2 lignes de code identiques = **fonction**

LUDIKSCIENCES

Exercice Factorielle

Ecrire la fonction fac (n) qui calcule le produit des n premiers nombres entiers naturels non nuls

n! se lit "Factorielle n"

1! = 1

2! = 1 x 2

3! = 1 x 2 x 3

…

On convient que 0!=1

LUDIKSCIENCES

Exercice PGCD

Ecrire la fonction pgcd (a, b) qui donne le pgcd de deux entiers a et b

PGCD = Plus Grand Commun Diviseur

Quelques exemples :

pgcd(4, 8) =4
pgcd (10,4)= 2
pgcd (2,3) =1

Méthode d'Euclide : si b divise a, leur pgcd est b. Sinon c'est le pgcd de b et du reste dans la division entière de a par b

Rappel : a//b = division entière a%b = reste

10/3 = 3.3333333333333335

10//3 = 3

10%3 = 1

Exercice Trinôme

Soient a, b, c des réels

Ecrire la fonction trinome(a,b,c, x) qui calcule ax^2+bc+c

Ecrire la fonction solve(a,b,c) qui recherche les solutions de l'équation $ax^2+bx+c=0$

Pour chaque solution X trouvée, calculer le résidu trinome (a,b,c, X)

Montrer que tous les cas sont pris en compte par votre algorithme

Calculer les résidus pour différents trinômes

$x^2+2x-8=0$

$x^2+111.11x+1.2121=0$

$x^2+1634x+2=0$

LUDIKSCIENCES

Exercice Convertisseur de devises

Réaliser un convertisseur de devises :

Le programme demande :

- Une devise d'origine dev1 ; par exemple "EUR"
- Une devise destination dev2 ; par exemple "USD"
- Un taux de change t ; par exemple 1.10191
- Un montant à convertir m ; par exemple 100 de la devise d’origine vers la devise de destination

Le programme affiche le montant converti :

ex : 100 EUR = 110.191 USD

LUDIKSCIENCES