
j3

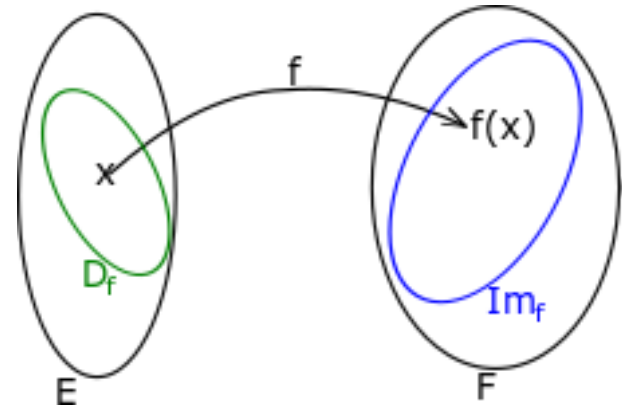


J3 : Objectifs

- Fonctions
- Commentaires
- Types élémentaires
- Transtypage
- Comparaison
- Expressions
- Mots-clés
- Indentation
- Tabulation
- Point-virgule
- Longueur des lignes
- Action conditionnelle
- Action alternative
- Action vide, pass
- Fonction de bibliothèque, import
- Définition de fonction
- Exercices

Fonctions en Maths

- Une fonction f
- Un ensemble de départ E
- Un ensemble d'arrivée F
- Domaine de définition $D(f)$
- Ensemble image $\text{Im}(f)$



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 ".

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 secondaire" ou "effet de bord" - *side effect*.
- Certaines fonctions sont surtout utiles par leur effet de bord (print, par exemple)
- En dehors du domaine de définition, le résultat du calcul est imprédictible (et souvent catastrophique TSVP)



Fonctions Python

- Prédéfinies
- Fonctions de bibliothèques
- Fonctions que l'on définit soi-même...



Fonctions prédéfinies

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

- Consulter `dir(__builtins__)`

Classeur Jupyter Dir.ipynb

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 ou de tests
- Ignorés par l'interpréteur

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 à typage dynamique.
- Python 3.6 a ajouté le *type hint* (une indication du type attendu)
 - a: intDes outils (pycharm) vérifient alors le type.

Transtypage

- *type cast*
- `int(x)` convertit x en int.

De même ...

- `float(x)`
- `str(x)`
- `bool(x)`
- `complex(x)`

- Utile dans les entrées sorties
 `type(input("Entrez un entier"))`
 `int(input("Entrez un entier"))`

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 ?

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 $a==b$, $\text{not}(a==b)$, $a!=b$

Exercices

- Rien
 - Faire un script Python rien.py qui ne fait rien comme ci-contre
- Bonjour
 - Faire un script Python bonjour.py qui affiche "Bonjour"
- Addition
 - Faire un script Python qui lit deux entiers, i1 et i2, au clavier et qui affiche leur somme sur l'écran

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

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

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

Les mots-clés, mots réservés de Python3

False	None	True	and	as	assert
async	await	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)
```

Les mots-clés, déjà connus

False	None	True	and	as	assert
async	await	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	

Noter que les mots-clés nous sont donnés dans l'ordre alphabétique. Expliquer !

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


Indentation

- Les autres langages utilisent le principe des parenthèses pour exprimer
 - la séquence
 - l'inclusionL'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
- L'inclusion est représentée par un décalage vers la droite (indentation).
 - le retour vers la gauche correspond à la fermeture de l'inclusion

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

Nombreux exemples à suivre

Tabulation

- TAB = 4 espaces.
- Recommandation ne pas utiliser CTL+I, Ascii 9
- Interdiction de mixer TAB=CTL+I et et TAB=4 espaces

Point-virgule

- Si l'on tient à introduire plusieurs actions sur la même ligne, alors il faut les séparer par un point-virgule
- Le point virgule en fin de ligne est contre-productif
 - ajout d'une instruction vide

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

Longueur des lignes

- Une ligne de source ne doit pas être prolongée au-delà de la 72^{ème} 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 : [({

if

- Action conditionnelle
- Les : sont indispensables
- L'indentation, aussi
- Inclusion d'une action ou d'une séquence d'actions
- Essayer d'écrire sans indenter

```
if condition :  
    action1  
    ...  
    actionN  
suite
```

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

if else

- Action alternative

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

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 Fortrann'existent pas en Python3
on fait autrement...

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

Condition

- Condition = expression booléenne
- 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
- `bool(expr)` force la conversion de l'expression `expr` en booléen

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

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

```
if bool(expr)==True: # inélégant

if expr:             # Pépette contente!
```


Expression conditionnelle

- Opérateur ternaire
 - 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

```
val1 if cond else val2
```

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

Action

- Ne rien faire
- Affectation
- Appel de fonction prédéfinie
- Fonction importée
 - Importer une fonction de bibliothèque
- Fonction définie
 - Définir une fonction

pass

- Ne rien faire
- no op
- *placeholder*
- prend la place d'une action dans un algorithme
 - conserver la structure correct du code et passer l'étape de vérification de la syntaxe
- souvent pour quelque chose qui reste à développer

```
...:  
    pass  
suite
```

import

- importer une bibliothèque
- importer une fonction d'une bibliothèque
- importer une fonction d'une bibliothèque en la baptisant d'un nom commode

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

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

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

def

- Définition d'une fonction
 - son nom
 - ses arguments "(...)"
 - les ":"
 - une docstring """ ... """
 - l'algorithme qui retourne la valeur
- Appel de la fonction

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

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

Vocabulaire

- Définition de la fonction f
 - x est un "Argument".
 - On dit aussi "Argument formel" ou "Paramètre formel"
 - "Formel" parce qu'il sert à définir la "*forme*", c'est à dire, l'algorithme de la fonction.
- Appel de la fonction f
 - 2 est un "Paramètre"
 - on dit aussi "Argument effectif" ou "Paramètre effectif"
 - Effectif parce qu'il sert à effectuer le calcul de la valeur de la fonction
- Correspondance des arguments dans la définition et dans l'utilisation de la fonction
 - par la position – arguments positionnels
 - par le nom – arguments nommés – voir plus loin

```
def f(x):
```

```
f(2)
```

Don't Repeat Yourself

Faites plutôt
des fonctions !



Exercice IMC

- Demander la masse en kg
- Demander la taille en mètres
- Calculer l'IMC
 - Indice de Masse CorporellePar la formule
 - $\text{Masse} / (\text{Taille}^2)$
- Si la valeur obtenue est inférieure à 18.5, conseiller de plus manger
- Si la valeur obtenue est supérieure à 25, conseiller de faire un régime pour maigrir
- Sinon, dire que tout va bien



Exercice Factorielle

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

$n!$ se lit "Factorielle n "

$$1! = 1$$

$$2! = 1 \times 2$$

$$3! = 1 \times 2 \times 3$$

...

On convient que $0! = 1$

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`.

Exercice Listes

- A partir des primitives
 - prem, reste, videconstruire les fonctions
 - deuxième
 - troisième
 - dernier
 - card
 - appartient

Classeur Jupyter Listes.ipynb

Exercice Trinôme

- Soient a, b, c des réels.
- Ecrire la fonction `trinome(a,b,c, x)` qui calcule ax^2+bx+c
- Ecrire la fonction `solve(a,b,c)` qui recherche les solutions de l'équation $ax^2+bx+c=0$
- Prendre en compte les cas dégénérés
- 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

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

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

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

- Expliquer

Exercice Convertisseur de devises

- Réaliser un convertisseur de devises qui, étant donnés
 - Une devise d'origine dev1 ; par exemple "EUR"
 - Une devise destination dev2 ; par exemple "USD"
 - Un taux de change t ; par exemple 1,19724
- Lire un montant m
- Afficher le montant converti
 - 100 EUR = 110.191 USD

Les mots-clés ajoutés

False	None	True	and	as	assert
async	await	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	

Merci !

- Restons en contact :
 - Georges Georgoulis – ggeorgoulis@alteractifs.org – 06 12 68 40 06



Coopérative d'activité et d'entrepreneurs www.alteractifs.org