

# PYTHON

FRÉDÉRIC DROUHIN

Corrections : M. Lochtenbergh, A. Albert

Rev. 05/2020

ASUR

IUT de Colmar / Département Réseaux & Télécom.

## SOMMAIRE

### LES ÉLÉMENTS DU LANGAGE

LES INSTRUCTIONS

LES LISTES, TUPLES, SETS ET DICTIONNAIRES

MODULE, FONCTION, PROCÉDURE, MAIN

EXPRESSIONS RÉGULIÈRES

FICHIERS ET COMMANDES SYSTÈME

# INTRODUCTION

## Langage de programmation

- Orienté objets
- Petits programmes simples (scripts)
- Applications
- Langage interprété



## Nombreux modules

<https://www.python.org/>

- Version 3.8.2 (oct. 2019)
- Licence BSD

Le créateur :  
Guido van Rossum

<http://apprendre-python.com/page-syntaxe-differentes-python2-python3-python-differences>

# QUELQUES RÉFÉRENCES

Appréhender le cours :

- <https://www.w3schools.com>
- <https://www.w3schools.com/python/exercise.asp>

Le site de Python :

- [www.python.org](http://www.python.org)

Quelques sites que j'ai consultés :

- <https://www.developpez.com/>
- <http://apprendre-python.com>
- <https://openclassrooms.com>

# EDITEURS & COMPILEUR

- Editeurs
  - PyCharm (MacOs, Windows, Linux)
  - Visual Studio (si si Microsoft ®)
  - Eclispe (tous les systèmes)
  - vi / vim (c'est pour faire plaisir à Sébastien)
  - Emacs (bon là, c'est moi ...)
  - Et plein d'autres (mais pas nano ! Et pas un notepad !)
- Compilateurs :
  - <https://www.pypy.org/>
  - <http://www.py2exe.org/>

# EXTENSION DE VOS SCRIPTS

- .py : script modifiable
- .pyc : script compilé
- .pyw : script exécuté sans lancement de terminal sous windows
- Vous pouvez les rendre exécutables
  - chmod / propriétés windows
  - Ligne de shebang dont je reparle par la suite
  - `#!/usr/bin/env python3`

# SYNTAXE : IDENTATION

Indentation → point clé pour Python

- Indentation = décalage d'une instruction par rapport à la marge de gauche
- Toute ligne se terminant par « : » doit être indenté

## Indentation

Ligne d'entête :

\_\_\_\_\_ instruction(s)

Pas de « ; » dans le code

## Exemple : un simple if

```
if a > 0:  
    print ("a est positif")  
else:  
    print("a est négatif")  
    a = -a  
print(a)
```

Les fameux 2 points « : »  
print dans le if  
print dans le else  
affectation dans le else  
print effectué que a > 0 ou non → en dehors de la condition

Il n'est pas nécessaire d'avoir des tabulations trop importantes (1 espace peut suffire)

# SYNTAXE : COMMENTAIRES

## Les commentaires

- Eléments importants du langage
- Commentaire d'une ligne #
- Bloc de commentaire """ ... """

## Exemple : des commentaires

```
""" Ce programme prend
    la valeur absolue d'une variable
    ici a
"""

if a > 0:
    print ("a est positif")
else:
    print("a est négatif")
    a = -a
# Affichage de a (commentaire de ligne)
print(a)
```

# VARIABLE

## Utilisation

variable

- (i) Non fortement typé

## Affectation

variable=valeur

- (i) Opération `+=` ; `-=` ; `*=` ; `/=`

- (ii) `a=b=c=1` initialisation multiple

## Exemple

```
str="bla bla bla"  
a=3  
print(str,"=",a)  
x1=input("Entrez x1")  
print("x1=",x1,sep=' ')
```

# TYPE DE DONNÉES

Type	Type en python	Opérateur	Représentation	Remarque
Entiers signés	int	+ ; - ; * ; ** ; // ; %	32 bits	Depuis python 3
Réels signés	float	+ ; - ; * ; **	64 bits	Attention à la précision
Chaîne de caractères	str string basestring	+ ; *		Iterable
Booléen	bool	and ; or	True ; False	

## Type de variable

```
type(variable)
<class 'type'>
```

# QUELQUES EXEMPLES

Exemple dans Python console

## La fonction type

```
>>> a= True
>>> type(a)
<class 'bool'>

>>> b=3
>>> type(b)
<class 'int'>

>>> str="3"
>>> type(b)
<class 'str'>
```

## Conversion

```
>>> a=int(str)
>>> print(a)
3

>>> f=float(str)
>>> print(f)
3.0
```

## Conversion

```
>>> str="aaa"
>>> a=int(str)
Traceback (most recent call last):
  File "<input>", line 1, in <module>
ValueError: invalid literal for int()
with base 10: 'aaa'
```

## Précision

```
>>> 0.1 + 0.2
0.3000000000000004
```

## Iterable

```
>>> str="aaa"
>>> for c in str:
            print(c)
a
a
a
>>> len(str)
3
```

# LES MOTS RÉSERVÉS

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

## SOMMAIRE

LES ÉLÉMENTS DU LANGAGE  
LES INSTRUCTIONS  
LES LISTES, TUPLES, SETS ET DICTIONNAIRES  
MODULE, FONCTION, PROCÉDURE, MAIN  
EXPRESSIONS RÉGULIÈRES  
FICHIERS ET COMMANDES SYSTÈME

# CONDITION

## Condition simple

```
if expression:  
    instruction(s)
```

## Condition et alternative

```
if expression:  
    instruction(s)  
elif expression:  
    instruction(s)  
else:  
    instruction(s)
```

# EXEMPLE DE CONDITIONS

## Exemple

```
#!/usr/bin/env python3
import sys
if len(sys.argv) < 5:
    print(sys.argv[0],": error in usage")
else:
    print("Nombre d'arguments",len(sys.argv))
    if "coucou" not in sys.argv[1]:
        print(sys.argv[1],"ne contient pas coucou")
```

# `sys.argv` est un tableau & `len` une fonction  
# pas de `main` (mais nous allons y revenir)  
# vous ai-je expliqué le `sep` dans `print` ?  
# et pourquoi # ?  
# et la ligne de shebang

# OPÉRATEURS

Ecriture de l'opérateur en Python	Exemple	
	Expression	Résultat
<code>==</code>	<code>2 == 2</code>	<code>True</code>
<code>!=</code>	<code>3 != 5</code>	<code>True</code>
<code>&lt;</code>	<code>-1 &lt; -5</code>	<code>False</code>
<code>&gt;</code>	<code>4 &gt; 4</code>	<code>False</code>
<code>&lt;=</code>	<code>4 ≤ 4</code>	<code>True</code>
<code>&gt;=</code>	<code>8 ≥ 3</code>	<code>True</code>

# BOUCLES

for

```
for x in tableau:  
    instructions  
  
for i in range(0,10):  
    instructions
```

*range(stop) |  
range(start,stop) |  
range(start,stop,step)*

while

```
while expression:  
    instructions  
  
(i) not expression
```

# EXEMPLE DE BOUCLES

## Exemple

```
#!/usr/bin/env python3
import sys
for x in range(1, len(sys.argv)):
    print(sys.argv[x])

for x in sys.argv:
    print(x)
```

```
flag=False
while not flag:
    str=input("Q pour quitter : ")
    if "Q" in str:
        flag=True
```

Tiens un booléen

# BOUCLES

If : tester deux conditions en même temps

```
if expression1 and expression2:  
    instructions  
    Valable pour un elif  
  
if expression1 or expression2:  
    instructions
```

while : tester deux conditions en même temps

```
while expression1 and expression2:  
    instructions  
  
while expression1 or expression2:  
    instructions
```

## SOMMAIRE

LES ÉLÉMENTS DU LANGAGE  
LES INSTRUCTIONS  
LES LISTES, TUPLES, SETS ET DICTIONNAIRES  
MODULE, FONCTION, PROCÉDURE, MAIN  
EXPRESSIONS RÉGULIÈRES  
FICHIERS ET COMMANDES SYSTÈME

# LISTE (TABLEAU ?)

## Déclaration

```
variable = [élément 1, élément 2, ...]
```

(i) Type hétérogène et longueur « infinie » (donc plutôt liste)

## Utilisation, taille, iterable

```
variable[index]
```

```
variable[index] = valeur
```

```
variable = variable[index]
```

```
len(variable)
```

## Exemple

```
divers=["coucou",1,3.0,True]
```

```
divers[0]="truc"
```

```
a=divers[1]
```

```
type(a)
```

```
for x in divers:
```

```
    print(x)
```

```
for i in range(len(divers)):
```

```
    print(divers[i])
```

# LISTE (TABLEAU)

## Déclaration

```
liste = [élément 1, élément 2, ...]
(i)    Type hétérogène et longueur « infinie »
(ii)   Python ne prend pas en charge les tableaux utilisation des listes Python
```

## Utilisation, taille, iterable

```
liste[index]
liste[index] = valeur
liste = variable[index]
len(liste)
```

## Exemple

```
divers=["coucou",1,3.0,True]
divers[0]="truc"
a=divers[1]
type(a)
for x in divers:
    print(x)
for i in range(len(divers)):
    print(divers[i])
```

```
# Elément dans la liste
if "coucou" in divers:
    print("coucou dans divers")
else:
    print("coucou not in divers")
print(divers)
```

# LISTE

Méthode	Description	Exemple
<code>append()</code>	Ajout d'un élément à la fin	<code>divers.append("truc")</code>
<code>clear()</code>	Supprime tous les éléments	<code>divers.clear()</code>
<code>copy()</code>	Renvoie une copie de la liste	<code>copie = divers.copy()</code>
<code>count()</code>	Renvoie le nombre d'éléments	<code>taille = divers.count()</code>
<code>extend()</code>	Ajout une liste à la fin	<code>divers.extend(copie)</code>
<code>index()</code>	Renvoie l'index d'un élément	<code>divers.index("Truc")</code>
<code>insert()</code>	Insère un élément	<code>divers.insert(1, "bidule")</code>
<code>pop()</code>	Renvoie et supprime l'élément à l'indice donné	<code>elt = divers.pop(1)</code>
<code>remove()</code>	Renvoie et supprime l'élément spécifié (1 <sup>er</sup> élément)	<code>elt=divers.remove("Truc")</code>
<code>reverse()</code>	Inverse la liste	<code>divers.reverse()</code>
<code>sort()</code>	Trie la liste (attention aux types)	<code>divers.sort()</code>

# TUPLE

- **Tuple : liste non modifiable**
- **Iterable**

## Déclaration

```
monTuple = (élément 1, élément 2, ...)  
(i)    Type hétérogène et longueur « infinie »  
monTuple [index]  
variable = monTuple [index]
```

Méthode	Description	Exemple
<code>count()</code>	Nombre de fois ou un élément apparaît	<code>monTuple.count("truc")</code>
<code>index()</code>	Renvoie l'index d'un élément	<code>monTuple.index("truc")</code>

# SET

- **Set : liste non indexée et non ordonnée**
- **Ajout ou suppression d'éléments mais pas de changement**
- **Iterable**

## Déclaration

```
monSet = { élément 1, élément 2, ... }
(i)      Type hétérogène et longueur « infinie »
```

Quelques Méthodes	Description	Exemple
<code>add()</code>	Ajout d'un élément à la fin	<code>monSet.add("truc")</code>
<code>clear()</code>	Supprime tous les éléments	<code>monSet.clear()</code>
<code>copy()</code>	Renvoie une copie de la liste	<code>setCopy = monSet.copy()</code>
<code>difference()</code>	Différence entre deux sets	<code>z = monSet.difference(setCopy)</code>
<code>pop()</code>	Renvoie et supprime l'élément à l'indice donné	<code>elt = divers.pop(1)</code>
<code>remove()</code>	Renvoie et supprime l'élément spécifié (1 <sup>er</sup> élément)	<code>elt=divers.remove("truc")</code>

# DICTIONNAIRE

- **Association clé / valeur**
- **Non ordonnée, modifiable, indexé**

## Déclaration

```
dictionnaire = {  
    clé : valeur          (i) Type hétérogène et longueur « infinie »  
    clé : valeur  
    ...  
}  
  
dictionnaire[clé]  
dictionnaire.get(clé)  
variable = dictionnaire[clé]
```

# DICTIONNAIRE

## Exemple

```
contacts = {  
    "Frédéric": "054566",  
    "Michaël": "054566",  
    "Jacques": "01167"  
}  
  
print(contacts)  
print(contacts["Frédéric"])  
val=contacts.get("Frédéric")  
contacts["Jacques"] = "3455"  
  
# Clé existe  
if "Laurent" not in contacts:  
    contacts["Laurent"] = "123"  
else:  
    print("Laurent est déjà présent")
```

```
# Affiche les clés  
for cle in contacts:  
    print(cle)  
  
# Affiche les valeurs  
for cle in contacts:  
    print(contacts[cle])  
for valeur in contacts.values():  
    print(valeur)  
  
# Clés et valeurs  
for cle,valeur in contacts.items():  
    print(cle,":",valeur)
```

# DICTIONNAIRE

Méthode	Description	Exemple
<code>clear()</code>	Supprime tous les éléments	<code>contacts.clear()</code>
<code>copy()</code>	Renvoie une copie de la liste	<code>copie = contacts.copy()</code>
<code>fromkeys()</code>	Création d'un dictionnaire	<code>x = ("Jacques", "Frédéric") y = (1, 2) myDict = dict.fromkeys(x, y) {'Jacques': (1, 2), 'Frédéric': (1, 2)}</code>
<code>get()</code>	Renvoie la valeur pour la clé	<code>contacts.get("Jacques")</code>
<code>items()</code>	Renvoie la paire clé, valeur	<code>contacts.items()</code>
<code>keys()</code>	Renvoie l'ensemble des clés	<code>contacts.keys()</code>
<code>pop()</code>	Renvoie et supprime l'élément pour la clé donnée	<code>elt = contacts.pop("Jacques")</code>
<code>popitem()</code>	Renvoie le dernier tuple clé, valeur et le supprime	<code>elt=contacts.popitem()</code>
<code>setdefault()</code>	Si la clé n'existe pas, ajoute la clé et la valeur (None si pas de valeur)	<code>contacts.setdefault("Laurent", 123) contacts.setdefault("Arnauld")</code>
<code>update()</code>	Ajoute une paire clé, valeur	<code>contacts.update("Benoît": "06")</code>
<code>values()</code>	Renvoie les valeurs	<code>contacts.values()</code>

## SOMMAIRE

LES ÉLÉMENTS DU LANGAGE  
LES INSTRUCTIONS  
LES LISTES, TUPLES, SETS ET DICTIONNAIRES  
**MODULE, FONCTION, PROCÉDURE, MAIN**  
EXPRESSIONS RÉGULIÈRES  
FICHIERS ET COMMANDES SYSTÈME

# IMPORT

Nom du module	Utilisation
random	Valeurs aléatoires
math	Fonctions mathématiques
<b>sys</b>	<b>Fonctions systèmes</b>
<b>os</b>	<b>Interaction avec le système d'exploitation (fichiers, ...)</b>
time	Fonctions de calendrier
profile	Analyse de l'exécution des fonctions
urllib2	Récupération d'informations sur internet
<b>re</b>	<b>Expressions régulières (voir plus loin)</b>
<b>scappy</b>	<b>Manipulation de paquets réseaux</b>

## Exemples

```
import os
fd = "GFG.txt"
os.rename(fd,'New.txt')
os.rename(fd,'New.txt')
```

```
import calendar
obj = calendar.Calendar()
# iteratign with yeardatescalendar
for day in obj.yeardatescalendar(2018, 1):
    print(day)
```

<https://www.geeksforgeeks.org/os-module-python-examples/>

# LES IMPORTS

- **Pour ajouter des fonctions ou des packages**

## Import

```
import module
from module import fonction
from module import Classe
from package.sous_package import module
```

## Exemple

```
import os
from hashlib import md5
from xml.etree import Element
from xml.sax import saxutils
```

# MAIN

main

```
if __name__ == '__main__':
    instructions
```

- Pas de fonction ou méthode main()
- Explicite is better than implicit
- *Si le code est exécuté en tant que script principal (appelé directement avec Python et pas importé), alors exécuter cette fonction.*

# PROCÉDURE & FONCTION

## Procédure & Fonction

```
def procédure(paramètres) :  
    instructions  
  
def fonction(paramètres) :  
    instructions  
    return valeurderetorno
```

Remarque :

Vous avez aussi l'opérateur `lambda` pour créer des fonctions simples et rapides dans le code directement

# PROCÉDURE & FONCTION

## Exemple

```
#!/usr/bin/env python3
import sys
def usage(scriptName,error):
    print(error)
    print("Usage:",scriptName,"arguments",sep=' ')
    exit(1)

def factorielle(n):
    if n < 0:
        usage(sys.argv[0],"n ne peut être négatif")
    elif n == 0:
        return 1
    else:
        return n * factorielle(n-1)

if __name__ == '__main__':
    if len(sys.argv) <= 1:
        usage(sys.argv[0],"Pas d'entier spécifier")
    else:
        n=int(sys.argv[1])
        res = factorielle(n)
        print(n,"! =",res,sep='')
```

# PROCÉDURE ET FONCTION : ANNOTATIONS

## Exemple : sans annotation

```
#!/usr/bin/env python3
import sys
def usage(scriptName, error):
    print(error)
    print("Usage:", scriptName)
    exit(1)

def factorielle(n):
    if n < 0:
        usage(sys.argv[0], "n < 0")
    elif n == 0:
        return 1
    else:
        return n * factorielle(n-1)
```

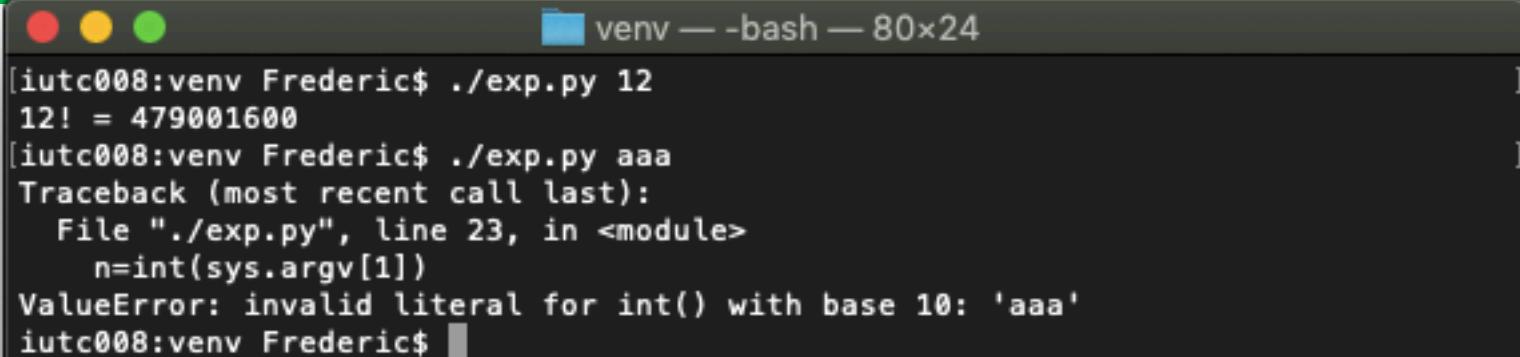
## avec annotation

```
#!/usr/bin/env python3
import sys
def usage(scriptName : str, error : str):
    print(error)
    print("Usage:", scriptName)
    exit(1)

def factorielle(n : float) -> float:
    if n < 0:
        usage(sys.argv[0], "n < 0")
    elif n == 0:
        return 1
    else:
        return n * factorielle(n-1)
```

# PROCÉDURE & FONCTION

## Exemple



A screenshot of a terminal window titled "venv — -bash — 80x24". The window shows the following command-line interaction:

```
[iutc008:venv Frederic$ ./exp.py 12
12! = 479001600
[iutc008:venv Frederic$ ./exp.py aaa
Traceback (most recent call last):
  File "./exp.py", line 23, in <module>
    n=int(sys.argv[1])
ValueError: invalid literal for int() with base 10: 'aaa'
iutc008:venv Frederic$ ]
```

C'est pour vous montrer la ligne de commande !

# GESTION DES ERREURS PAR EXCEPTIONS

## Exception

```
try:  
    instruction(s)  
    instruction  
    instruction(s)  
except Error1:  
    instruction(s)  
except (Error2,Error3):  
    instruction(s)  
except:  
    instruction(s)  
finally:  
    instruction(s)
```

## Quelques exceptions

```
except (RuntimeError, TypeError, NameError):  
    pass
```

<https://docs.python.org/2/library/exceptions.html>

# PROCÉDURE & FONCTION

## Exemple

```
[iutc008:venv Frederic$ ./exp.py aaa
n doit être un entier
Usage: ./exp.py : valeur n à utiliser
iutc008:venv Frederic$ ]
```

## Exemple

```
if __name__ == '__main__':
    if len(sys.argv) <= 1:
        usage(sys.argv[0], "Pas d'entier spécifier")
    else:
        try:
            n=int(sys.argv[1])
            res=factorielle(n)
            print(n,"! = ",res,sep=' ')
        except ValueError:
            usage(sys.argv[0], "n doit être un entier")
```

# CLASSE

Je ne vais pas m'étendre ici sur la notion de classe, héritage, agréation, polymorphisme, etc.

## Classe Simple

```
class NomClasse:  
    # variable statique  
    # Constructeur  
    def __init__(self,arguments)  
        initialisation des attributs  
  
    # Méthodes : fonctions ou procédures  
  
    # Création d'un objet  
monObjet=NomClasse()  
# Suppression de l'objet  
del objet
```

# CLASSE

## Exemple

```
#!/usr/bin/env python3

class Article:

    createdArticle = 0

    def __init__(self, designation, prixHT, quantite):
        self.designation = designation
        self.prixHT = prixHT
        self.quantite = quantite
        Article.createdArticle += 1

    def prixTTC(self,tva):
        return self.prixHT * (1 + tva / 100)

if __name__ == '__main__':
    a=Article("coucou",100,200)
    print(a.prixTTC(20))
    print("Jusqu'ici tu as créé : ", Article.createdArticle,
          " article(s)", sep=' ')
    del a
```

## SOMMAIRE

LES ÉLÉMENTS DU LANGAGE  
LES INSTRUCTIONS  
LES LISTES, TUPLES, SETS ET DICTIONNAIRES  
MODULE, FONCTION, PROCÉDURE, MAIN  
EXPRESSIONS RÉGULIÈRES  
FICHIERS ET COMMANDES SYSTÈME

# EXPRESSIONS RÉGULIÈRES (IMPORT)

- Traitements conditionnels sur les chaînes de caractères
- La recherche d'un motif dans une chaîne de caractères (correspondance, pattern-matching).
- La substitution.

## Import

```
import re

re.findall(...)
re.search(...)
re.split(...)
re.sub(...)
```

<https://docs.python.org/2/library/re.html>

<http://apprendre-python.com/page-expressions-regulieres-regular-python>

# EXPRESSIONS RÉGULIÈRES (MÉTHODES)

Méthode	Description	Exemple
<code>findall()</code>	Renvoie l'ensemble des éléments sous forme de liste	<pre>&gt;&gt;&gt; str = "Mon chat est jaune" &gt;&gt;&gt; re.findall("a",str) ['a', 'a']</pre>
<code>search()</code>	Renvoie un objet de la classe <code>_sre.SRE_Match</code> , <code>None</code> si inexistant	<pre>&gt;&gt;&gt; ar = re.search("a",str) &gt;&gt;&gt; ar.start()</pre>
<code>split()</code>	Renvoie une liste de chaînes de caractères en utilisant le délimiteur spécifié	<pre>&gt;&gt;&gt; sp = re.split(" ",str) ['Mon', 'chat', 'est', 'jaune'] &gt;&gt;&gt; np = re.split("a",str,_1) ['Le ch', 't est jaune']</pre>
<code>sub()</code>	Remplace la chaîne de caractère par une autre chaîne	<pre>&gt;&gt;&gt; re.sub(" ","-",str) 'Mon-chat-est-jaune' &gt;&gt;&gt; re.sub(" ","-",str,_2) 'Mon_chat_est jaune'</pre>
<code>compile()</code>	Permet de compiler une expression régulière avant son utilisation	<pre>&gt;&gt;&gt; regex=re.compile("^( [A-Z] ) ") &gt;&gt;&gt; if regex.match(str) is not None:     print(s,"Commence par une MAJ")</pre>

# EXPRESSIONS RÉGULIÈRES (SYMBOLES)

	<b>Le motif présent</b>	<b>Exemple</b>	<b>Mots matchés</b>
*	0 fois ou plus	a*	Mot vide, a, aa, aaa, bab, baab, ...
+	1 fois ou plus	a+	a, aa, aaa, bab, baab, ...
?	0 ou une fois	a?	Mot vide, a, bab, ...
{n}	n fois exactement	a{4}	aaaa, baaaab, ...
{n,}	au moins n fois	a{3,}	aaa, aaaa, baaab, ...
{,n}	au plus n fois	a{,2}	Mot vide, a, aa, bab, ...
{m,n}	Entre n et m fois	a{2,5}	aa, aaa, aaaa, aaaaa, baaaaab, ...

# EXPRESSIONS RÉGULIÈRES (EXPRESSION)

## Intervalle de caractères : [...]

Tous les caractères minuscules (ensemble [ ])

Ex : [a-z] ; [2a-zA-Z] ; [A-Z] ; [0-9] ; [R-Z];[aze] ;[aze] ;  
[aeiouyie] est équivalent à [aeiouy]

## Tiret

Intervalles ou se termine par un motif :[a-z][4-]

## Intervalle au milieu d'un motif

motif[...]motif : tous les caractères minuscules (ensemble [ ])

Ex : ch[ao0-9]t : chat, chot, ch0t, ...

tot[a-zA0-9]V : totaV, totbV ... totzV, totAV, tot0V ... tot9V.

## Quelques raccourcis

\d : un chiffre, équivalent à [0-9] (d = digit)

\D : un non-numérique, équivalent à [^0-9]

\w : un alphanumérique, équivalent à [0-9a-zA-Z\_] (w = word)

\W : un non-alphanumérique, équivalent à [^0-9a-zA-Z\_]

\s : un espace, équivalent à [\n\t\r\f] (s comme space)

\S : un non-espace, équivalent à [^\n\t\r\f]

« . » : n'importe quel caractère (par exemple t.t.)

« ^ » :

En début : commence par : ^[A-Z][a-z]+

En milieu de motif : tout sauf : t[^ao]t.

« \$ » : se termine par (en fin de motif)

« | » : ou : mots | mot (contient mots ou mot)

# EXPRESSIONS RÉGULIÈRES (EXEMPLE)

## Exemple

```
regex=re.compile("^( [+-] ? (\d) + \. ? (\d) *) \$")
s=input("Elément à tester : ")
if regex.match(s) is not None:
    print("C'est un décimal ou un entier")
else:
    print("C'est pas un nombre")
```

Symbole	Explication
^	Commence par
[+-] ?	« + » OU « - », « ? » → 0 ou 1 (optionnel)
\d+	Au moins un digit ([0-9])
\. ?	« . », « ? » → 0 ou 1 (optionnel)
\d*	Des digits (optionnel)
\$	Se termine par

Vérifier ces éléments :

+123.4	-123.45
123.45	
-123.	123
abc	1abc.12

# EXPRESSIONS RÉGULIÈRES (GROUP())

- Récupération de certains éléments d'une chaîne
- Nommage des groupes

## Exemple

```
# La chaîne de caractères correspond (ou pas)
regex=re.search("\w+;\w+;\w+", "drouhin;frédéric;12345")
print(regex.group()) # ou regex.group(0) -> affiche la totalité de la chaîne

# Récupération d'élément(s) de la chaîne
regex=re.search("( \w+ ) ; ( \w+ ) ; ( \w+ )", "drouhin;frédéric;12345")
print(regex.group())
print(regex.group(1))
print(regex.group(2))
print(regex.group(3))

# Récupération d'élément(s) de la chaîne et nommage dans un dictionnaire
regex=re.search(" (?P<nom>\w+) ; (?P<pre>\w+) ; (?P<num>\w+)", "drouhin;frédéric;12345")
print(regex.group())
print(regex.group("nom"))
print(regex.group("pre"))
print(regex.group("num"))
```

# EXPRESSIONS RÉGULIÈRES (SLIPT())

- Séparation d'une chaîne suivant un caractère spécifique

## Exemple

```
# Je sépare la ligne suivant un caractère
liste=re.split(";", "drouhin;frédéric;12345")
print("Le nom est :", liste[0])
for x in liste:
    print(x)
```

# QUELQUES EXERCICES

**Immatriculation d'une voiture**

**Adresse web**

**Saisie d'un nombre réel / entier**

**Variable suivant une convention de nommage « Java »**

**Détection des noms/prénoms composés**

**Email uha**

## SOMMAIRE

LES ÉLÉMENTS DU LANGAGE  
LES INSTRUCTIONS  
LES LISTES, TUPLES, SETS ET DICTIONNAIRES  
MODULE, FONCTION, PROCÉDURE, MAIN  
EXPRESSIONS RÉGULIÈRES  
**FICHIERS ET COMMANDES SYSTÈME**

# FICHIER

## Accès aux Fichiers

### Ouverture

```
open(fichier, mode) : flux
    fichier : nom du fichier
    mode : mode d'accès (r, w, a, r+)
    flux : flux du fichier (TextIOWrapper)
    encoding='utf-8'
```

### Lecture par itération dans le flux (mode = r)

attention aux \n\r

Utilisation de la méthode ligne.rstrip("\n\r")

### Lecture par méthode (mode = r)

TextIOWrapper : read(), read(int), readline()

### Ecriture (mode = w) et écriture en fin de fichier (mode = a)

TextIOWrapper : write(string)

### Fermeture du fichier

flux.close()

# FICHIER

## Accès aux Fichiers

### Et si le fichier a un problème (existence, accès, ...) → Exception

- Attention à la fermeture du fichier en cas d'échec d'écriture ou de lecture
- Ecriture un peu fastidieuse pour s'assurer de la fermeture → mot clé `with`

### Mot clé `with`

- `with open(filename, mode) as file:`
- instruction(s) # pas besoin de fermer le fichier ici
- `file.closed() # true fichier fermé par le with !`

Attention, vous aurez quand même des exceptions !

### D'autres accès

`TextIOWrapper::name`

`TextIOWrapper::seek()`

# FICHIER

## Exemple

```
nom = "auth.txt"

try:
    with open(nom, "r") as infile, open("copie.txt", "w") as outfile:
        # Lecture d'une ligne
        ligne = infile.readline()
        print("ligne 1 = ", ligne)
        input("Press enter to continue")

        # Lecture du restant des lignes
        for ligne in infile:
            ligne = ligne.rstrip("\n\r")
            regex = re.search("toto", ligne)
            if regex:
                print(ligne)
                outfile.write(ligne+"\n")

except (FileNotFoundException, PermissionError, IOError):
    print("Problem with file:", nom, "or file", "copie.txt")
```

# COMMANDES SYSTÈME

## Commandes Système

Il existe plein de commande système :

- <https://docs.python.org/2/library/os.html>
- Dépendante du système pour certaines
- `import os`

En voici quelques unes :

- `os.chdir()`  
    changement de répertoire
- `os.listdir()`  
    récupération des fichiers et répertoires
- `os.getcwd()`  
    récupération du répertoire courant

Vous avez également le package `pwd`

- `pwd.getpwuid(uid)`
- Objet contenant les informations sur l'utilisateur

# COMMANDES SYSTÈME

## Commandes Système

Il existe plein de commande système :

- <https://docs.python.org/2/library/os.html>
- Dépendante du système pour certaines
- import os

En voici quelques unes :

- os.chdir()
  - changement de répertoire
- récupération des fichiers et répertoires
- os.getcwd()
- récupération du répertoire courant
- os.stat()
  - Données du système de fichiers

Vous avez également le package pwd

- pwd.getpwuid(uid)
- Objet contenant les informations sur l'utilisateur

## Exemple

```
# Qui suis-je ?  
user = pwd.getpwuid(os.getuid())  
print("Hello", user.pw_name)  
  
for p in user:  
    print(p)  
  
  
# Dernière modification  
from time import ctime  
dir=os.environ["HOME"]  
os.chdir(dir)  
print("Check", os.getcwd())  
for f in os.listdir():  
    stats=os.stat(f)  
    d=ctime(stats.st_mtime)  
    print(f, "last modifications", d)
```

# COMMANDES SYSTÈME : OS.STAT

Champs	Signification
st_mode	Nombre codé sur 16 bits, décrivant le type et les permissions du fichier
st_ino	Numéro d'inode. Identifiant unique de l'inode sur le périphérique
st_dev	Numéro de device, qui identifie le périphérique de façon unique sur le système
st_nlinks	Nombre d'entrées de répertoire qui pointent vers cet inode
st_uid	Identifiant de l'utilisateur propriétaire du fichier
st_gid	Identifiant du groupe propriétaire du fichier
st_size	Taille du fichier en octets
st_atime	Access time. Date du dernier accès, sous la forme d'un timestamp Unix. Ce champ est mis à jour chaque fois qu'un programme lit les données du fichier
st_mtime	Modification time. Date de la dernière modification. Ce champ est mis à jour chaque fois qu'un programme écrit des données dans le fichier.
st_ctime	Status change time. Date de la dernière modification de cet inode. Attention, ce n'est PAS la date de création du fichier.

# COMMANDES SYSTÈME : STAT PACKAGE

Fonctions	Résultats	<a href="https://docs.python.org/2/library/stat.html">https://docs.python.org/2/library/stat.html</a>
<code>stat.S_ISDIR(mode)</code>	Return non-zero if the mode is from a directory.	
<code>stat.S_ISCHR(mode)</code>	Return non-zero if the mode is from a character special device file.	
<code>stat.S_ISBLK(mode)</code>	Return non-zero if the mode is from a block special device file.	
<code>stat.S_ISREG(mode)</code>	Return non-zero if the mode is from a regular file.	
<code>stat.S_ISFIFO(mode)</code>	Return non-zero if the mode is from a FIFO (named pipe).	
<code>stat.S_ISLNK(mode)</code>	Return non-zero if the mode is from a symbolic link.	
<code>stat.S_ISSOCK(mode)</code>	Return non-zero if the mode is from a socket.	
<code>stat.S_IMODE(mode)</code>	Return the portion of the file's mode that can be set by <code>os.chmod()</code> —that is, the file's permission bits, plus the sticky bit, set-group-id, and set-user-id bits (on systems that support them).	
<code>stat.S_IFMT(mode)</code>	Return the portion of the file's mode that describes the file type (used by the <code>S_IS*</code> () functions above).	
<code>stat.S_ISDIR(mode)</code>	Return non-zero if the mode is from a directory.	

## Exemple

```
stats=os.stat(f)
If stat.S_ISDIR(stats.st_mode):
    print("f","est un dossier")
```

MERCI !