

VALEURS ET TYPES LIST, TUPLE, DICT (VOIR PAGES SUIVANTES)

Types numériques

|         |                     |  |
|---------|---------------------|--|
| int     | a = 5               | integer (entier compris entre -∞ ... +∞) |
| float   | c = 5.6 , c = 4.3e2 | floating point number (nombre décimal)   |
| complex | d = 5 + 4j          | complex numbers (nombres complexes)      |

Strings (Types d'objets itérables, mais non modifiables)

|     |             |  |
|-----|-------------|--|
| str | e = "hello" | Character string, chaîne de caractères |
|-----|-------------|--|

Conversion de type

|             |   |
|-------------|---|
| int(s)      | convertir chaîne s en nombre entier       |
| float(s)    | convertir chaîne s en nombre décimal      |
| str(number) | convertir nombre entier/décimal en string |
| list(x)     | convert tuple, range or similar to list   |

Noms des variables ⇒ case sensitive (différence entre caractères majuscules et minuscules)

Certains mots réservés ne sont pas autorisés :

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 (print, sum ⇒ not recommended, else internal functions will be overridden)

|                                |  |
|--------------------------------|--|
| lettres (a...z , A...Z)        | caractères autorisés, doit commencer par une lettre                        |
| chiffres (0...9)               |  |
| _ (underscore, blanc souligné) |  |
| i, x                           | boucles et indices ⇒ lettres seules en minuscule                           |
| get_index()                    | modules, variables, fonctions et méthodes<br>⇒ minuscules + blanc souligné |
| MAX_SIZE                       | (pseudo) constantes ⇒ majuscules et blanc souligné                         |
| CamelCase                      | nom des classes ⇒ CamelCase  |

CHAÎNES DE CARACTÈRES (=SÉQUENCES NON-MODIFIABLES, IMMUTABLE)

Les caractères d'une chaîne ne peuvent pas être modifiés. Python ne connaît pas de caractères. Un caractère isolé = chaîne de longueur 1. Dans les exemples suivants : s = chaîne de caractères

- ord('A') ⇒ return integer Unicode code point for char (e.g. 65)
- chr(65) ⇒ return string representing char at that point (e.g. 'A')

String literals

|                                     |   |
|-------------------------------------|---|
| "texte" ou 'texte'                  | délimiteurs doivent être identiques                   |
| """ chaîne sur plusieurs lignes """ | chaîne sur plusieurs lignes, délimitée par """ ou ''' |
| 'abc\''def' ou 'abc\'def'           | inclure le délimiteur dans la chaîne                  |
| \n                                  | passage à la ligne suivante                           |
| \\                                  | pour afficher un \                                    |

Caractères et sous-chaînes (Voir les exemples sous ⇒ Listes-Affichage)

Opérateurs

|                              |                                |
|------------------------------|--------------------------------|
| "abc" + "def" ou "abc" "def" | ⇒ "abcdef" (concaténation)     |
| "abc" * 3 ou 3 * "abc"       | ⇒ "abcabcabc" (multiplication) |

Affichage ⇒ f-string (formatted strings), chaîne de char. préfixée par f ou F

|                                      |   |
|--------------------------------------|---|
| f"{var1} x {var2} = {var1 * var2}"   | {...} = remplacé par variables ou expression  |
| "{} x {} = {}".format(v1, v2, v1*v2) | ou bien : str.format()                        |
| "{0}{1}{0}".format('abra', 'cad')    | ⇒ "abracadabra" (on peut aussi les numéroter) |

Placeholder options

|  |  |
|--|--|
| {:format-spec}                           | {:4} ou {:>4} ⇒ padding of 4, right aligned  |
| format-spec is : [fill]align             | {:5} ⇒ truncate to 5 chars                   |
| fill = espace (par défaut)               | {:10.5} ⇒ padding of 10, truncate to 5       |
|  | {:.2f} ⇒ display as float with 2 decimals    |
| {x:3d} ⇒ display as integer, padding = 3 | {:6.2f} ⇒ float with 2 decimals, padding = 6 |

|        |                                       |  |
|--------|---------------------------------------|--|
| align: | < left-aligned                        | = padding after sign, but before numbers |
|        | > right-aligned (default for numbers) | ^ centered                               |

Utiliser une variable var1 dans format-spec : "{:{var1}}...".format(..., var1 = value, ...)

Méthodes

|                                     |  |
|-------------------------------------|--|
| s.capitalize()                      | renvoie une copie avec le premier caractère en majuscule   |
| s.lower()   s.upper()               | renvoie une copie en lettres minuscules   majuscules   |
| s.strip()                           | renvoie une copie et enlève les caractères invisibles (whitespace) au début et à la fin de s   |
| s.strip(chars)                      | renvoie une copie et enlève les caractères chars au début et à la fin de s   |
| s.split()                           | renvoie une liste des mots (délimités par whitespace), pas de mots vides   |
| s.split(sep)                        | renvoie une liste des mots (délimités par sep), sous-chaînes vides si plusieurs sep consécutifs  |
| s.find(sub[, start[, end]])         | renvoie l'indice de la 1 <sup>ère</sup> occurrence de sub dans la sous-chaîne [start:end] de s, renvoie -1 si pas trouvé idem, mais exception ValueError si pas trouvé |
| s.index(sub[, start[, end]])        | renvoie une copie avec les n (default = toutes) premières occurrences de old remplacés par new   |
| s.replace(old, new[, n])            |  |
| s.isalpha()                         | True si au moins un caractère et que des lettres   |
| s.isdigit()                         | True si au moins un chiffre et que des chiffres  |
| s.isalnum()                         | True si au moins un caractère et que des lettres ou chiffres   |
| s.islower()                         | True si au moins une lettre et que des minuscules  |
| s.isupper()                         | True si au moins une lettre et que des majuscules  |
| s.isspace()                         | True si au moins un whitespace et que des whitespace   |
| for char in s :                     | parcourir les lettres de la chaîne de caractères   |
| s.join(iterable)                    | returns a string created by joining the elements of an iterable by string separator  |
| "xx".join("123") ⇒ "1xx2xx3"        |  |
| s.join([str(elem) for elem in lst]) | convertir liste en chaîne avec séparateur s  |

LISTES (=SÉQUENCES MODIFIABLES) ⇒ [] type: list

Dans une même liste ⇒ variables de différents types = possible.

Création

\* = unpack operator

|   |  |
|---|--|
| lst = []                                    | créer une liste vide                         |
| lst = [item1, item2, ... ] , lst = [23, 45] | créer une liste avec des éléments            |
| new_lst = lst1 + lst2 ( = [*lst1, *lst2] )  | Attention : crée une nouvelle liste          |
| list(x), ex : lst = list(range(5))          | Convertir uplet, range ou semblable en liste |

Remarque

|                              |   |
|------------------------------|---|
| A = B = [] ⇒ A = [] et B = A | les 2 noms (A et B) pointent vers la même liste |
|------------------------------|---|

list comprehensions (computed lists)

|   |  |
|---|--|
| lst = [expr for var in sequence]        | expr is evaluated once for every item in sequence, |
| lst = [expr for var in sequence if ...] | (if is optional)                                   |

Exemple : création d'une matrice 3x3

|  |   |
|--|---|
| p = [x[:] for x in [[0]*3]*3] ou p = [[0,0,0], [0,0,0], [0,0,0]] | 1. construire 3 vecteurs, chacun avec 3 composants nuls<br>2. une copie est placée dans p, pour obtenir 3 vecteurs-lignes indépendants, ne pointant pas sur le même objet |
|--|---|

Affichage et sous-listes

premier élément d'une liste ⇒ index 0

|                       |   |
|-----------------------|---|
| lst[index]            | retourne l'élément à la position index<br>(un index < 0 ⇒ accède aux éléments à partir de la fin) |
| lst[start :end]       | retourne une sous-liste de l'indice start à end (non compris)                                     |
| lst[start :end :step] | (seuls les éléments avec index = multiple de step inclus)   |

|           |  |
|-----------|--|
| lst[-1]   | retourne le dernier élément de lst                                       |
| lst[2:-1] | sous-liste à partir de l'indice 2 jusqu'à l'avant dernier                |
| lst[:4]   | sous-liste à partir du début jusqu'à l'indice 3                          |
| lst[4:]   | sous-liste à partir de l'indice 4 jusqu'à la fin                         |
| lst[:]    | retourne la liste entière, pour copier une liste dans une autre variable |
| lst[::2]  | retourne sous-liste des éléments à index pair                            |
| lst[::-1] | retourne sous-liste des éléments dans l'ordre inverse                    |

Pour copier une liste

|                                      |  |
|--------------------------------------|--|
| lst = [2, 3, 4, 5]                   | 1 <sup>st</sup> level copy (copie = lst ne fonctionne pas, car variables pointent alors sur la même liste) |
| copie = lst[:] ou copie = lst.copy() | variables pointent alors sur la même liste   |
| copie = [x[:] for x in lst]          | copier une liste de listes (2 <sup>nd</sup> level copy, shallow copy)                                      |
| copie = copy.deepcopy(lst)           | import copy (any level copy, deep copy)  |

Modification

|                                  |  |
|----------------------------------|--|
| lst[index] = item                | modifie l'élément à la position index  |
| lst[start :end] = [...]          | remplace la sous-liste à partir de start jusqu'à end (exclu), même de taille différente      |
| lst.append(item)                 | add item as single element to end of existing list   |
| lst.extend(iterable)             | add each element of iterable (all items) to the existing list by iterating over the argument |
| lst += [item1, ..., item_n]      |  |
| lst = lst + [item1, ..., item_n] | Attention: create new list and add all items from both                                       |
| del lst[index] , del(lst[index]) | supprime l'élément à la position index   |
| lst.remove(item)                 | supprime le premier élément avec la valeur item  |
| lst.pop()                        | enlève et retourne le dernier élément de la liste (à la position indiquée par index)         |
| lst.pop(index)                   |  |
| lst.reverse()                    | inverse les items d'une liste (modifie la liste)   |
| new_lst = reversed(lst)          | retourne une liste inversée (lst = unchanged)  |
| lst.sort()                       | trier la liste (modifie la liste)  |
| new_lst = sorted(lst)            | retourne une liste triée (lst = unchanged)   |
| lst.insert(index, item)          | insère l'item à la position donnée par index   |

Attention :

|   |   |
|---|---|
| lst = [1, 2, 3, 4]                      | lst = [1, 2, 3, 4]                        |
| lst[2] = [7,8,9] ⇒ [1, 2, [7, 8, 9], 4] | lst[2:2] = [7,8,9] ⇒ [1, 2, 7, 8, 9 , 4]  |
| (liste imbriquée)                       | (élément remplacé par plusieurs éléments) |

Divers

|                                     |  |
|-------------------------------------|--|
| print(lst)                          | affiche le contenu de la liste   |
| len(lst)                            | nombre d'items dans lst  |
| lst.count(item)                     | nombre d'occurrences de la valeur item   |
| lst.index(item)                     | retourne l'index de la 1 <sup>ère</sup> occurrence de item, sinon ⇒ exception ValueError                         |
| item in lst (item not in lst)       | indique si l'item se trouve dans lst (n'est pas dans)  |
| min(lst) / max(lst)                 | retourne l'élément avec la valeur min. / max.  |
| sum(lst[,start])                    | retourne la somme à partir de start (= 0 par défaut)   |
| for item in lst:                    | parcourir les éléments   |
| for index in range(len(lst)):       | parcourir les indices  |
| for index, item in enumerate(lst):  | parcourir l'indice et les éléments   |
| for item in reversed(lst):          | parcourir dans l'ordre inverse   |
| for item in lst[:]:                 | effacer éléments d'une liste ⇒ utiliser copie de lst   |
| for i in range(len(lst)-1, -1, -1): | effacer certains éléments d'une liste ⇒ il faut parcourir la liste de la fin au début, si on a besoin de l'index |
| ... code pour effacer des items     |  |
| while i < len(lst):                 | effacer certains éléments d'une liste  |
| if ... code pour effacer items      |  |
| else:                               |  |
| i = i + 1                           |  |
| if lst: ou if len(lst) > 0:         | test si la liste lst n'est pas vide  |

RANGE (=SÉQUENCES NON MODIFIABLES)

Retourne une séquence non modifiable d'entiers

|                                |   |
|--------------------------------|---|
| range([start], stop[, step])   | retourne une séquence d'entiers sans la valeur stop     |
| (start, stop, step = integers) | range(n) ⇒ [0,1,2, ..., n-1], ex.: range(3) ⇒ [0, 1, 2] |
|                                | range(2, 5) ⇒ [2, 3, 4]                                 |
|                                | range(0, -10, -2) ⇒ [0, -2, 4, -6, -8]                  |

LES UPLETS (TUPLES) -> () type: tuple

Uplet = collection d'éléments séparés par des virgules. Comme les chaînes **pas modifiables**

Création

|                                     |  |   |
|-------------------------------------|--|---|
| <code>tuple = (a, b, b, ...)</code> | <code>t = ("a", 2.4, 45)</code>              | créer un uplet                              |
| <code>tuple = a, b, c, ...</code>   | <code>t = (1,)</code> ou <code>t = 1,</code> | (on peut omettre les parenthèses, si clair) |
| <code>tuple1 = tuple2</code>        |  | copier un uplet                             |

Extraction

|  |                                  |
|--|----------------------------------|
| <code>(x, y, z) = tuple</code> ou <code>x, y, z = tuple</code> | extraire les éléments d'un uplet |
|--|----------------------------------|

Affichage ⇨ voir listes

Premier élément d'un uplet ⇨ index 0

|                               |   |
|-------------------------------|---|
| <code>tuple[index]</code>     | retourne l'élément à la position index<br>(un index < 0 ⇨ accède aux éléments à partir de la fin) |
| <code>tuple[start:end]</code> | retourne un sous-uplet de l'indice [start ; end[  |

LES DICTIONNAIRES -> {} type: dict

Les dictionnaires sont modifiables, mais pas des séquences. L'ordre des éléments est aléatoire. Pour accéder aux objets contenus dans le dictionnaire on utilise des clés (keys). Classe : **dict**

Création

|   |  |
|---|--|
| <code>dic = {}</code> ou <code>dic = dic()</code> | créer un dictionnaire vide   |
| <code>dic = {key1: val1, key2: val2, ...}</code>  | créer un dictionnaire déjà rempli :<br><code>d = {"nom": "John", "age": 24}</code>             |
| <code>dic[key] = value</code>                     | ajouter une clé:valeur au dictionnaire si la clé n'existe pas encore, sinon elle est remplacée |

key peut être alphabétique, numérique ou type composé (ex. uplet)

Affichage

|   |  |
|---|--|
| <code>dic[key]</code>                                     | retourne la valeur de la clé keys. Si la clé n'existe pas une exception <b>KeyError</b> est levée              |
| <code>dic.get(key, default = None)</code>                 | retourne la valeur de la clé, sinon <b>None</b> (ou la valeur spécifiée comme 2 <sup>e</sup> paramètre de get) |
| <code>dic.keys()</code>                                   | retourne les clés du dictionnaire  |
| <code>list(dic.keys())</code> ou <code>list(dic)</code>   | ... comme liste  |
| <code>tuple(dic.keys())</code>                            | ... comme uplet  |
| <code>sorted(dic.keys())</code>                           | renvoie une liste des clés dans l'ordre lexicographique  |
| <code>dic.values()</code> <code>list(dic.values())</code> | renvoie les valeurs du dictionnaire / comme liste  |
| <code>dic.items()</code> <code>list(dic.items())</code>   | renvoie les éléments du dictionnaire sous forme d'une séquence de couples / d'une liste de couples             |

Modification

|   |  |
|---|--|
| <code>dic[key] = value</code>                           | ajouter une clé:valeur au dictionnaire, si la clé n'existe pas encore (sinon elle est remplacée) |
| <code>del dic[key]</code> ou <code>del(dic[key])</code> | supprime la clé key du dictionnaire  |
| <code>dic.pop(key)</code>                               | supprime la clé key du dictionnaire et renvoie la valeur supprimée                               |

Divers

|   |  |
|---|--|
| <code>len(dic)</code>   | renvoie le nombre d'éléments dans le dictionnaire  |
| <code>if key in dic: , if key not in dic</code>                 | tester si le dictionnaire contient une certaine clé  |
| <code>for c in dic.keys():</code> ou <code>for c in dic:</code> | parcourir les clés d'un dictionnaire   |
| <code>for c, v in dic.items():</code>                           | parcourir les éléments du dictionnaire   |
| <code>copie = dic.copy()</code>                                 | crée une copie (shallow copy) du dictionnaire (une affectation crée seulement un nouveau pointeur sur le même dictionnaire) - 1 <sup>er</sup> level copy |
| <code>copie = copy.deepcopy(dic)</code>                         | <code>import copy</code> (any level copy)  |
| <code>max(dic, key=len)</code>                                  | retourne la clé la plus longue   |
| <code>dic1.update(dic2)</code>                                  | combine 2 dictionnaires en un seul (dic1), les clés de dic2 sont prioritaires  |

Expressions et opérateurs

Opérateurs entourés d'espaces. Utiliser des parenthèses pour grouper des opérations (modifier la priorité)

Opérateurs mathématiques

La 1<sup>ère</sup> colonne indique la priorité des opérateurs

|                 |  |  |
|-----------------|--|--|
| 1. <b>**</b>    | exponentiation   | <code>6 ** 4 ⇨ 1296</code>                                 |
| 2. <b>- , +</b> | signe  | <code>-5</code>  |
| 3. <b>*</b>     | multiplication   | <code>x *= 3 ⇨ x = x * 3</code>                            |
| <b>/</b>        | division (entière ou réelle)                                     | <code>x /= 3 ⇨ x = x / 3</code>                            |
| <b>//</b>       | quotient de la division entière (arrondi vers le négatif infini) | <code>6 // 4 ⇨ 1</code><br><code>-6.5 // 4.1 ⇨ -2.0</code> |
| <b>%</b>        | modulo, reste (positif) de la division entière                   | <code>6 % 4 ⇨ 2, -6.5 % 4.1 ⇨ 1.7</code>                   |
|                 | obtient le signe du diviseur                                     | <code>6 % -4 ⇨ -2</code>                                   |
| 4. <b>+</b>     | addition   | <code>x += 3 ⇨ x = x + 3</code>                            |
| <b>-</b>        | soustraction   | <code>x -= 3 ⇨ x = x - 3</code>                            |

Opérateurs relationnels

retournent **True** ou **1** si l'expression est vérifiée, sinon **False** ou **0**

|              |                         |   |
|--------------|-------------------------|---|
| 5. <b>==</b> | égal à                  |   |
| <b>!=</b>    | différent de            |   |
| <b>&gt;</b>  | strictement supérieur à |   |
| <b>&lt;</b>  | strictement inférieur à |   |
| <b>&gt;=</b> | supérieur ou égal à     | (exemple : <code>x &gt;= a</code> ou <code>b &gt;= x &gt;= a</code> pour <code>a &lt;= b</code> ) |
| <b>&lt;=</b> | inférieur ou égal à     | (exemple : <code>x &lt;= b</code> ou <code>a &lt;= x &lt;= b</code> )                             |

chaînes de caractères ⇨ ordre lexicographique, majuscules précèdent les minuscules

Opérateurs logiques

|                   |  |
|-------------------|--|
| 6. <b>not</b> x   | <b>non</b> (retourne <b>True</b> , si x est faux, sinon <b>False</b> )   |
| 7. x <b>and</b> y | <b>et</b> (retourne x, si x est faux, sinon y)<br><b>and</b> ne vérifie le 2 <sup>e</sup> argument que si le 1 <sup>er</sup> argument est vrai |
| 8. x <b>or</b> y  | <b>ou</b> (retourne y, si x est faux, sinon x)<br><b>or</b> ne vérifie le 2 <sup>e</sup> argument que si le 1 <sup>er</sup> argument est faux  |

Affectation

L'affectation attribue un type bien déterminé à une variable.

|                                    |   |
|------------------------------------|---|
| <code>variable = expression</code> | Affectation simple, attribuer une valeur à une variable |
| <code>a = b = c = 1</code>         | affectation multiple                                    |
| <code>x, y = 12, 14</code>         | affectation parallèle                                   |
| <code>x, y = y, x</code>           | échanger les valeurs des 2 variables (swap)             |

Entrée / Sortie

Entrée

|  |  |
|--|--|
| <code>var = input()</code>             | renvoie une chaîne de caractères                       |
| <code>var = input(message)</code>      | renvoie une chaîne de caractères et affiche le message |
| <code>int = int(input(...))</code>     | renvoie un entier                                      |
| <code>float = float(input(...))</code> | renvoie un nombre décimal                              |

Sortie

|  |   |
|--|---|
| <code>print(text, end="final")</code>        | affiche text et termine avec final (par défaut end="\n")      |
| <code>print("abc", "def")</code>             | ⇨ abc def (arguments séparés par espace, nouvelle ligne)      |
| <code>print("abc", end="+")</code>           | ⇨ abc+ (pas de passage à la ligne)                            |
| <code>print(var)</code>                      | var est converti en chaîne et affichée                        |
| <code>print("value=", var)</code>            | affiche le texte suivi d'une espace, puis de la valeur de var |
| <code>print()</code>                         | simple passage à la ligne                                     |
| <code>print(str * n)   print(n * str)</code> | afficher n fois le texte str                                  |

Les commentaires

|  |   |
|--|---|
| <code># commentaire</code>                                 | sur une seule ligne                     |
| <code>'''comments'''</code> ou <code>"""comments"""</code> | sur plusieurs lignes (= string literal) |

Structure alternative et répétitive

Structure alternative

|   |  |
|---|--|
| <code>if condition1:</code><br><code>instruction(s)</code><br><code>elif condition2:</code><br><code>instructions(s)</code><br>...<br><code>else:</code><br><code>instruction(s)</code><br><code>&lt;on true&gt; if &lt;expr&gt; else &lt;on false&gt;</code> | <ul style="list-style-type: none"><li>exécute seulement les instructions, où la condition est vérifiée</li><li>si aucune condition n'est vérifiée, les instructions de <b>else</b> sont exécutées</li><li><b>else</b> et <b>elif</b> sont optionnels</li></ul> <ul style="list-style-type: none"><li>ternary operator (opérateur ternaire)</li></ul> |
|---|--|

Structure répétitive (boucle for)

|  |  |
|--|--|
| <code>for itérateur in liste de valeurs:</code><br><code>instruction(s)</code><br><code>for i in range(10):</code> # values 0, 1, ... 9<br><code>for _ in range(10):</code> # values 0, 1, ... 9 | <ul style="list-style-type: none"><li>répète les instructions pour chaque élément de la liste</li><li>nombre de répétitions = connu au départ</li><li><b>_</b> si valeur de l'itérateur n'est pas utilisée</li></ul> |
|--|--|

Structure répétitive (boucle while)

|   |   |
|---|---|
| <code>while condition(s):</code><br><code>instruction(s)</code> | <ul style="list-style-type: none"><li>répète les instructions tant que la condition est vraie</li><li><b>pour pouvoir sortir de la boucle, la variable utilisée dans la condition doit changer de valeur</b></li><li>nombre de répétitions != connu au départ</li></ul> |
|---|---|

A l'intérieur d'une boucle **for** ou **while**

|                 |                                      |
|-----------------|--------------------------------------|
| <b>break</b>    | quitte la boucle immédiatement       |
| <b>continue</b> | continue avec la prochaine itération |

Les fonctions

Le code de la fonction doit être placé plus haut dans le code source (avant l'appel de la fonction).

- arguments simples (nombres, chaînes, uplets) ⇨ passage par valeur (valeurs copiés)
- arguments complexes (listes, dictionnaires) ⇨ passage par référence (vers les originaux)

Définition et appels

|   |  |
|---|--|
| <code>def my_function(par1, ..., par_n):</code><br><code>instruction(s)</code><br>...<br><code>return var</code>  | définit une fonction my_function <ul style="list-style-type: none"><li>par1 ... par_n sont les paramètres</li><li>une ou plusieurs instructions <b>return</b>...</li><li>peut renvoyer plusieurs réponses (uplet, liste)</li></ul> Si la fonction ne contient pas d'instruction <b>return</b> , la valeur <b>None</b> est renvoyée |
| <code>my_function(arg1, ... arg_n)</code><br><code>var = my_function(arg1, ... arg_n)</code><br><code>my_function(*lst)</code><br><code>my_function(**dct)</code> | appel de la fonction, arguments affectés aux paramètres dans le même ordre d'apparition <ul style="list-style-type: none"><li>* to unpack list elements</li><li>* to unpack dictionary elements</li></ul>  |
| <code>def func(par1, ..., par_n = val):</code><br><code>ex : def add(elem, to = None):</code><br><code>if to is None:</code><br><code>to = []</code>              | paramètre par défaut<br><b>ATTENTION:</b><br><code>def add(elem, to = []):</code><br>ne marche pas, car les arguments par défaut ne sont évalués qu'une seule fois et réutilisés pour tous les appels de la fonction.  |
| <code>def func(par1, ..., *par_n):</code>   | *par_n = nombre variable de paramètres (liste)   |

\* = unpack operator to unpack list elements <https://docs.python-guide.org/writing/gotchas/>

Variables globales

Les paramètres et variables locales cachent les variables globales/extérieures.

|                             |  |
|-----------------------------|--|
| <code>def func(...):</code> | var est déclaré comme variable global, la variable var à l'extérieur de la boucle est <b>global</b> var donc modifiée/utilisée |
|-----------------------------|--|

UTILISATION DE MODULES (BIBLIOTHÈQUES)

Utiliser des modules

|  |  |
|--|--|
| <code>import module</code>   | importe tout le module, il faut préfixer par le nom du module .<br>Ex : <code>import math</code> ⇒ <code>math.sqrt()</code>                  |
| <code>import module as name</code><br><code>from module import *</code><br><code>*** à éviter ***</code> | intègre toutes les méthodes de module, pas besoin de préfixer le nom du module<br>ex : <code>from math import *</code> ⇒ <code>sqrt()</code> |
| <code>from module import m1, m2, ...</code><br><code>from math import sqrt, cos</code>                   | intègre seulement les méthodes mentionnées<br>⇒ <code>sqrt(...)</code> , <code>cos(...)</code>   |

MODULE: MATH import math

Built-in functions (no import required)

|                       |  |
|-----------------------|--|
| <code>abs(x)</code>   | valeur absolue (aussi nombres complexes)   |
| <code>round(x)</code> | x est arrondie vers l'entier pair le plus proche <ul style="list-style-type: none"><li><code>round(3.5)</code> ⇒ 4 (rounds to nearest EVEN integer)</li><li><code>round(4.5)</code> ⇒ 4 (rounds to nearest EVEN integer)</li></ul> |

|  |   |
|--|---|
| <code>import math</code>                     |   |
| <code>math.pi</code>                         | le nombre pi                                |
| <code>math.cos(x) / .sin(x) / .tan(x)</code> | cosinus/sinus/tangente d'un angle en radian |
| <code>math.sqrt(x)</code>                    | racine carrée                               |
| <code>math.fabs(x)</code>                    | valeur absolue ⇒ retourne un float          |
| <code>math.ceil(x) / math.floor(x)</code>    | x est arrondie vers le haut / vers le bas   |
| <code>math.trunc(x)</code>                   | retourne l'entier sans partie décimale      |
| <code>math.pow(x, y)</code>                  | x exposant y                                |
| <code>math.gcd(x, y)</code>                  | retourne le PGCD des 2 nombres              |

MODULE: RANDOM import random

|   |   |
|---|---|
| <code>random.randint(a, b)</code>   | retourne un entier au hasard dans l'intervalle [a ; b]  |
| <code>random.random()</code>  | retourne un réel au hasard dans l'intervalle [0 ; 1]  |
| <code>random.uniform(a, b)</code>   | retourne un réel au hasard dans l'intervalle [a ; b]  |
| <code>random.choice(seq)</code>   | retourne un élément au hasard de la séquence seq (si seq est vide ⇒ exception <code>IndexError</code> )   |
| <code>random.sample(seq, k)</code>  | retourne une liste de k éléments uniques (choisis au hasard) de la séquence seq   |
| <code>random.randrange(stop)</code><br><code>random.randrange(start, stop)</code><br><code>random.randrange(start, stop, step)</code><br><code>random.shuffle(seq)</code> | retourne un entier au hasard de [start ; stop]. Seuls les multiples de step sont possibles. (start = 0, step = 1 par défaut)<br>mélange aléatoirement les éléments de seq |

retourner le nombre aléatoire -1 ou 1

|                                  |                                    |   |
|----------------------------------|------------------------------------|---|
| <code>randrange(-1, 2, 2)</code> | <code>2 * randint(0, 1) - 1</code> | <code>h = [-1, 1]</code> , <code>choice(h)</code> |
|----------------------------------|------------------------------------|---|

MODULE: TIMIT import timit

|  |  |
|--|--|
| <code>t1_start = timeit.default_timer()</code><br>...<br><code>t2_stop = timeit.default_timer()</code><br><code>print(t2_stop - t1_start)</code> | Return process time of current process as float in seconds |
|--|--|

LES FICHIERS

Entrées/sorties console et redirection

|  |  |
|--|--|
| STDIN                                  | entrée standard ⇒ le clavier (pour entrer des données)       |
| STDOUT                                 | sortie standard ⇒ l'écran (pour afficher les résultats)      |
| STDERR                                 | l'écran (pour envoyer les messages d'erreur)                 |
| <code>command &gt; filename</code>     | rediriger la sortie standard vers un fichier (créé/remplacé) |
| <code>command &gt;&gt; filename</code> | rediriger la sortie standard vers un fichier (ajouté)        |
| <code>command &gt; NUL</code>          | annuler sortie vers STDOUT                                   |
| <code>command &lt; filename</code>     | rediriger entrée depuis un fichier                           |

Tubes et filtres

|                                  |   |
|----------------------------------|---|
| <code>command1   command2</code> | rediriger la sortie de <code>command1</code> comme entrée à <code>command2</code> |
|----------------------------------|---|

Manipulation de fichiers

|  |  |
|--|--|
| <code>file = open(filename, mode='r')</code><br><code>file = open(filename, mode='r', encoding='utf-8')</code>   | retourne un objet fichier <ul style="list-style-type: none"><li><code>mode='r'</code> (lecture), <code>'w'</code> (écriture), <code>'a'</code> (ajout à la fin) ou <code>'x'</code> (création)</li><li>(optionel) choisir l'encodage du fichier avec <code>encoding='utf-8'</code>, <code>'ascii'</code> ou <code>'latin-1'</code></li></ul> |
| <code>line = file.readline()</code>  | lit et retourne la prochaine ligne complète avec caractère fin de ligne (retourne une chaîne vide <code>""</code> si la fin du fichier est atteinte)   |
| <code>for line in file:</code><br>...<br><code>line = file.readline()</code><br><code>while line != "":</code><br>...<br><code>line = file.readline()</code> | lit tout le fichier ligne après ligne (voir ci-dessous)<br>lit tout le fichier ligne après ligne ⇒ utiliser <code>line.strip()</code> pour enlever les caractères invisibles (espaces, newline) au début et à la fin d'une ligne   |
| <code>lines_list = file.readlines()</code><br><code>file.read()</code><br><code>file.write(str)</code><br><code>file.close()</code>                          | lit tout le fichier et retourne une liste de chaînes<br>lit tout le fichier et retourne une chaîne<br>écrit dans <code>file</code> la chaîne <code>str</code><br>ferme <code>file</code> (si traitement du fichier est terminé)  |

Lire de STDIN en Python (manière de filtres)

|  |  |
|--|--|
| <code>import sys</code><br><code>line = sys.stdin.readline()</code><br><code>while line != "":</code><br>...<br><code>line = sys.stdin.readline()</code> | lire les données de STDIN, ou<br><code>import sys</code><br><code>for line in sys.stdin:</code><br>... |
|--|--|

To terminate readline(), when STDIN is read from keyboard, press CTRL-D (CTRL-Z on Windows)

MODULE: STRING import string

|                                     |   |
|-------------------------------------|---|
| <code>string.ascii_uppercase</code> | chaîne de caractères pré-initialisée avec <code>'ABCDEF ... XYZ'</code> |
| <code>string.ascii_lowercase</code> | chaîne de caractères pré-initialisée avec <code>'abcdef ... xyz'</code> |

MODULE: SYS import sys

|   |  |
|---|--|
| <code>sys.stdin.readline()</code>         | lit la prochaine ligne de STDIN ( <code>' '</code> si EOF)                               |
| <code>sys.maxsize</code>                  | valeur max. d'un entier en Python (32-bit ⇒ 2 <sup>31</sup> , 64-bit ⇒ 2 <sup>63</sup> ) |
| <code>sys.setrecursionlimit(limit)</code> | définir la profondeur maximale de la pile lors d'appels récursifs                        |

MODULE: COPY import copy

|                                       |   |
|---------------------------------------|---|
| <code>copie = copy.deepcopy(x)</code> | renvoie une copie récursive (ou profonde) de x (= copie de l'objet et copies des objet trouvés dans l'objet original) |
|---------------------------------------|---|

MODULES ET LIBRAIRIES (PACKAGES)

Modules

⇒ fichiers dans lesquels on regroupe différentes fonctions

|   |   |
|---|---|
| 1. créer un fichier (module) contenant des fonctions                    | ⇒ utiliser les fonctions du module  |
| 2. dans un 2 <sup>e</sup> fichier utiliser : <code>import module</code> | <b>Attention</b> : lors de modifications dans le module, il faut d'abord supprimer le fichier avec l'extension <code>.pyc</code> dans le dossier : <code>__pycache__</code> |

Librairies (packages)

⇒ dossier complet pour gérer les modules, peuvent contenir d'autres dossiers

⇒ dossier principal doit contenir le fichier vide nommé `__init__.py`

|   |                       |
|---|-----------------------|
| 1. créer un dossier   | ⇒ créer une librairie |
| 2. ajouter des modules  |                       |
| 3. créer le fichier vide <code>__init__.py</code> dans le dossier |                       |

Installer des librairies (packages) externes

PyCharm

⇒ File -> Settings -> Project: votre projet actuel

⇒ Sélectionner l'interpréteur Python (p.ex. 3.6.1), puis cliquer sur le symbole + à droite

⇒ Choisir librairie à installer dans la liste (cocher "Install to user's site packages directory" si pas administrateur)

Thonny

⇒ Tools -> Manage Packages...

⇒ Entrez le nom de la librairie pour la rechercher et cliquer sur Install

PACKAGE : PILLOW from PIL import image

Module : Image (<https://pillow.readthedocs.io/en/5.1.x/>)

|  |  |
|--|--|
| <code>PIL.Image.open(fp, mode="r")</code>          | ouvre l'image fp et retourne un objet Image  |
| <code>PIL.Image.new(mode, size, color=0)</code>    | crée un nouveau objet image et le retourne <ul style="list-style-type: none"><li><code>mode = 'RGB'</code> ⇒ 3x8 bit pixels, true color</li><li><code>size</code> = uplet (largeur, hauteur)</li></ul> |
| <code>Image.crop(box=None)</code>                  | retourne une région rectangulaire <ul style="list-style-type: none"><li><code>box</code> = uplet (left, upper, right, lower)</li></ul>   |
| <code>Image.paste(im, box=None, mask=None)</code>  | copie l'image im sur cet image <ul style="list-style-type: none"><li><code>box</code> = uplet (left, upper) ou (left, upper, right, lower)</li></ul>   |
| <code>Image.save(fp, format=None, **params)</code> | enregistre l'image sous le nom fp  |

PROGRAMMATION ORIENTÉ OBJET (POO)

OOP = object oriented programming, Python = langage orienté objet hybride

Objet

Objet = structure de données valuées et cachées qui répond à un ensemble de messages

- attributs** = données/champs qui décrivent la structure interne
- interface de l'objet** = ensemble des messages
- méthodes** = réponse à la réception d'un message par un objet

**Principe d'encapsulation** ⇒ certains attributs/méthodes sont cachés

• **Partie publique** ⇒ visible et accessible par tous

• **Partie privée** ⇒ seulement accessible et utilisable par les fonctions membres de l'objet (invisible et inaccessible en dehors de l'objet)

**Principe de masquage d'information** ⇒ cacher comment l'objet est implémenté, seul son interface publique est accessible.

Classe (= définition d'un objet)

Instantiation ⇒ création d'un objet à partir d'une classe existante (chaque objet occupe une place dans la mémoire de l'ordinateur)

|   |  |
|---|--|
| <code>class ClassName:</code><br><code>def __init__(self, par1, ... par_n):</code><br>...<br><code>self.var1 = ...</code><br><code>self.var2 = ...</code><br><br><code>def __str__(self):</code><br>...<br><code>return chaîne_de_texte</code><br><br><code>def method1(self, ...):</code><br>...<br><code>return resultat</code> | définit la classe <code>ClassName</code> (CamelCase)<br>les fonctions sont appelées <b>méthodes</b> <ul style="list-style-type: none"><li><code>__init__()</code> ⇒ constructeur, appelé lors de l'instanciation</li><li><code>__str__(self)</code> ⇒ string representation of object, e.g. <code>print(object)</code></li><li><code>self</code> doit être le 1<sup>er</sup> paramètre et référence la classe elle-même</li><li><code>self.var...</code> ⇒ attributs, accessibles de l'extérieur</li><li><code>method...()</code> ⇒ méthodes, accessible de l'extérieur</li></ul> <b>Convention</b> : utiliser le préfix <code>(_)</code> si des attributs ou méthodes ne doivent pas être accédés de l'extérieur (même s'ils sont toujours accessibles) |
| <code>obj = ClassName(...)</code>   | instancie un nouvel objet de la classe dans la mémoire   |
| <code>obj.method(...)</code>  | appel de la méthode de l'objet ( <code>self = obj</code> est toujours passé comme 1 <sup>er</sup> paramètre)   |

Récurtivité

- Algorithme récursif ⇒ algorithme qui fait appel(s) à lui-même
- Attention : il faut prévoir une condition d'arrêt (= cas de base)
- Pour changer la limite max. de récursions ⇒ voir module **sys**

PYGAME BIBLIOTHÈQUE POUR CRÉER DES JEUX

Structure d'un programme Pygame

|   |   |
|---|---|
| <pre>import pygame, sys from pygame.locals import * pygame.init() WIDTH = ... HEIGHT = ... size = (WIDTH, HEIGHT) screen = pygame.display.set_mode(size) pygame.display.set_caption(str)</pre>  | <pre># Initialisation importer les librairies et initialiser les modules de pygame  # Création de la surface de dessin définir la largeur (0...WIDTH-1) et la hauteur (0...HEIGHT-1) de la fenêtre et retourner un objet de type surface  # Titre de la fenêtre définir le titre de la fenêtre  # Effacer surface de dessin remplir arrière-plan avec couleur  # Fréquence d'image créer l'objet clock avant la boucle  # Boucle principale boucle principale (infinie)</pre> |
| <pre>screen.fill(color)</pre>   | <pre># Gestion des événements Event loop • Gestion de tous les événements dans une seule boucle for à l'intérieur de la boucle principale. • Toutes les instructions if doivent être regroupées dans une seule boucle for</pre>   |
| <pre>FPS = frequence # en Hz clock = pygame.time.Clock() done = False while not done:     for event in pygame.event.get():         if event.type == QUIT:             done = True         elif event.type == &lt;type d'événement&gt;:             &lt;instruction(s)&gt;     ...     ... dessins ...     # mise à jour de l'écran     pygame.display.update()     # Fréquence d'image     clock.tick(FPS) pygame.quit() sys.exit()</pre> | <pre>insère des pauses pour respecter FPS (appel à la fin de la boucle principale) # Fermer la fenêtre et quitter le programme</pre>  |

Types d'événements

<https://www.pygame.org/docs/ref/event.html>

Événement de terminaison

|  |   |
|--|---|
| <pre>QUIT if event.type == QUIT:     ...</pre> | <pre>L'utilisateur a cliqué sur la croix de fermeture de la fenêtre. Pour terminer correctement, utiliser : pygame.quit() et sys.exit()</pre> |
|--|---|

Événements - clavier

<https://www.pygame.org/docs/ref/key.html>

|   |   |
|---|---|
| <pre>KEYDOWN / KEYUP if event.type == KEYUP:     if event.key == K_a:         ...</pre> | <pre>une touche du clavier est enfoncée / relâchée ⇒ event.key, event.mod indique quelle touche a été enfoncée K_a, K_b, ... touche a, b,... (pareil pour le reste de l'alphabet) K_0, K_1, ... touche 0, 1, ... en haut (pareil pour les autres chiffres) K_KP0, K_KP1, ... touche 0, 1, ... sur pavé numérique (pareil ...) K_LALT, K_RALT touche ALT (à gauche   à droite) K_LSHIFT, K_RSHIFT touche SHIFT (à gauche   à droite) K_LCTRL, K_RCTRL touche CONTROL (à gauche   à droite) K_SPACE touche espace K_RETURN touche ENTER K_ESCAPE touche d'échappement K_UP, K_DOWN, K_LEFT, K_RIGHT touches flèches KMOD_NONE no modifier keys pressed (can be used to reset pressed keys on KEYUP)</pre> |
|---|---|

|   |   |
|---|---|
| <pre>keys = pygame.key.get_pressed() if keys[K_LEFT] and not keys[K_RIGHT]:     ...</pre> | <pre>renvoie dictionnaire de toutes les touches pressées (rafraichir avec ⇒ pygame.event.get()) p. ex. faire une action aussi longtemps que la touche flèche ⇔ est enfoncée</pre> |
|---|---|

Événements – souris

<https://www.pygame.org/docs/ref/mouse.html>

|  |  |
|--|--|
| <pre>MOUSEBUTTONDOWN MOUSEBUTTONUP MOUSEMOTION if event.type == MOUSE...</pre> | <pre>un bouton de la souris a été enfoncé / relâché ⇒ event.pos, event.button la souris a été déplacée ⇒ event.pos, event.rel, event.buttons</pre> |
|--|--|

Boutons de la souris

|   |  |
|---|--|
| <pre>if event.button == 1:     ... pygame.mouse.get_pressed()</pre> | <pre>indique quel bouton a déclenché l'événement 1 = left, 2 = middle, 3 = right, 4 = scroll-up, 5 = scroll-down retourne séquence de 3 valeurs pour l'état des 3 boutons de la souris (de gauche à droite), True si enfoncé. Ex. : if pygame.mouse.get_pressed() == (True, False, False):     ... ⇒ tuple pour (left, middle, right) mouse buttons Ex. : (1,0,0) ⇒ value 1 if pressed, else 0</pre> |
|---|--|

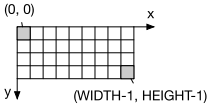
Position de la souris

|   |  |
|---|--|
| <pre>(x, y) = event.pos (x, y) = pygame.mouse.get_pos()</pre> | <pre>position du pointeur de souris à l'instant de l'événement position actuelle du pointeur de souris (uplet)</pre> |
|---|--|

La surface de dessin

Origine (0,0) = point supérieur gauche

- largeur de 0 ... WIDTH-1
- hauteur de 0 ... HEIGHT-1



Dimensions de la surface de dessin

|  |   |
|--|---|
| <pre>screen = pygame.display.get_surface() screen.get_width() screen.get_height() w, h = screen.get_size()</pre> | <pre>retourne la surface de dessin retourne la largeur de la surface de dessin retourne la hauteur de la surface de dessin retourne les dimensions de la surface de dessin sous forme d'uplet</pre> |
|--|---|

Couleurs

|  |   |
|--|---|
| <pre>color = Color(name) color = name color = Color(red, green, blue)</pre>    | <pre>renvoie la couleur du nom name (String), ex.: "white", "black", "green", "red", "blue" red, green, blue = nombres de 0 ... 255</pre> |
| <pre>Obtenir la couleur d'un point (pixel) color = screen.get_at((x, y))</pre> | <pre>retourne la couleur du point (pixel) à la position indiquée</pre>  |

Effacer/Remplir surface de dessin

|                                 |  |  |
|---------------------------------|--|--|
| <pre>screen.fill("black")</pre> | <pre>screen.fill(Color("black"))</pre> | <pre>remplir arrière-plan en noir</pre>  |
| <pre>screen.fill("white")</pre> | <pre>screen.fill(Color("white"))</pre> | <pre>remplir arrière-plan en blanc</pre> |

Dessiner une ligne/un point sur la surface (screen)

|   |   |
|---|---|
| <pre>pygame.draw.line(screen, color, start_point, end_point[, width])</pre> | <pre>• dessiner un point si start_point = end_point • start_point et end_point sont inclus • width = 1 par défaut</pre> |
| <pre>screen.set_at((x, y), color)</pre>                                     | <pre>dessiner un point (pixel) à la position (x, y)</pre>   |

Dessiner un rectangle sur la surface (screen)

|   |   |
|---|---|
| <pre>pygame.draw.rect(screen, color, rect_tuple[, width])</pre> | <pre>• rect_tuple = (x, y, width, height) avec x, y = coin supérieur gauche • ou rect_tuple = pygame.Rect(x, y, width, height) • width = 0 par défaut (= rectangle plein)</pre> |
|---|---|

Dessiner une ellipse inscrite dans le rectangle bounding\_rect sur la surface (screen)

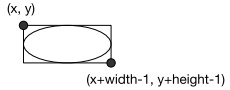
|   |   |
|---|---|
| <pre>pygame.draw.ellipse(screen, color, bounding_rect[, width])</pre> | <pre>• bounding_rect = (x, y, width, height) avec x, y = coin supérieur gauche • ou rect_tuple = pygame.Rect(x, y, width, height) • width = 0 par défaut (= ellipse pleine)</pre> |
|---|---|

Dessiner un cercle sur la surface (screen)

|   |   |
|---|---|
| <pre>pygame.draw.circle(screen, color, center_point, radius[, width])</pre> | <pre>• center_point = centre du cercle • radius = rayon • width = 0 par défaut (= cercle plein)</pre> |
|---|---|

Remarque : rect\_tuple et bounding\_rect

- coordonnées du point supérieur gauche : (x, y)
- coordonnées du point inférieur droit : (x+width-1, y+height-1)



Mise à jour de la surface de dessin

|  |   |
|--|---|
| <pre>pygame.display.update()</pre>     | <pre>rafraîchir la surface de dessin pour afficher les dessins</pre>        |
| <pre>pygame.display.flip()</pre>       |   |
| <pre>pygame.display.update(rect)</pre> | <pre>rafraîchir que la partie rect = pygame.Rect(x, y, width, height)</pre> |

Gestion du temps (fréquence de rafraîchissement)

avant la boucle principale

|  |  |
|--|--|
| <pre>FPS = frequence</pre>   | <pre>définir fréquence de rafraîchissement en Hz</pre>           |
| <pre>clock = pygame.time.Clock()</pre>   | <pre>créer un objet de type Clock</pre>                          |
| <pre>à la fin de la boucle principale (après la mise à jour de la surface de dessin)</pre> |  |
| <pre>clock.tick(FPS)</pre>   | <pre>insérer des pauses pour respecter la fréquence voulue</pre> |

pygame.Rect

|   |   |
|---|---|
| <pre>rect = Rect(left, top, width, height) rect = Rect((left, top), (width, height)) rect.normalize()</pre> | <pre>créer un nouveau objet Rect, avec left, top = coin supérieur gauche corrigé les dimensions négatives, le rectangle reste en place avec les coordonnées modifiées</pre> |
| <pre>rect.move_ip(x, y) rect.move(x, y) rect.contains(rect2)</pre>  | <pre>déplace rect de x, y pixels (retourne None) retourne un nouveau rect déplacé de x, y pixels retourne True si rect2 est complètement à l'intérieur de rect</pre>        |
| <pre>rect.collidepoint(x, y) rect.collidepoint((x, y)) rect.colliderect(rect2)</pre>                        | <pre>retourne True si le point donné se trouve à l'intérieur de rect retourne True si les 2 rectangles se touchent</pre>  |

Affichage de textes

|  |   |
|--|---|
| <pre>pygame.font.SysFont(name, size[, bold, italic])</pre>             | <pre>crée un objet de type Font à partir des polices système (bold et italic = False par défaut)</pre>                    |
| <pre>surface = font.render(text, antialias, color[, background])</pre> | <pre>dessine le texte text sur une nouvelle surface de dessin et retourne la surface (background = None par défaut)</pre> |
| <pre>screen.blit(source, dest[, area, special_flags])</pre>            | <pre>copie la surface source sur la surface screen à la position dest (coin sup. gauche)</pre>                            |
| <pre>pygame.display.update()</pre>                                     | <pre>met à jour la surface de dessin</pre>  |

Exemple:

|   |   |
|---|---|
| <pre>font = pygame.font.SysFont("comicansms", 20) surf_text = font.render("Hello", True, "green") screen.blit(surf_text, (100, 50)) pygame.display.update()</pre> | <pre>crée un objet font crée nouvelle surface avec texte copie la surface surf_text sur screen à la position indiquée et mise à jour (dest indique le coin sup. gauche)</pre> |
|---|---|

(surf\_text.get\_height(), surf\_text.get\_width()) ⇒ retourne la largeur/hauteur du texte)

Divers

|                                     |   |
|-------------------------------------|---|
| <pre>pygame.time.delay(delay)</pre> | <pre>interrompt le programme pour un nombre de ms donnés (delay) et renvoie le nombre effectif de ms utilisés</pre> |
| <pre>pygame.time.ticks()</pre>      | <pre>renvoie le temps en ms depuis l'appel de pygame.init()</pre>   |



ASCII CODES <https://theasciicode.com.ar/>

| ASCII Control characters |      |                       | ASCII printable characters |       |    |   |     |   |
|--------------------------|------|-----------------------|----------------------------|-------|----|---|-----|---|
| 00                       | NULL | (Null character)      | 32                         | space | 64 | @ | 96  | ` |
| 01                       | SOH  | (Start of Header)     | 33                         | !     | 65 | A | 97  | a |
| 02                       | STX  | (Start of Text)       | 34                         | "     | 66 | B | 98  | b |
| 03                       | ETX  | (End of Text)         | 35                         | #     | 67 | C | 99  | c |
| 04                       | EOT  | (End of Trans.)       | 36                         | \$    | 68 | D | 100 | d |
| 05                       | ENQ  | (Enquiry)             | 37                         | %     | 69 | E | 101 | e |
| 06                       | ACK  | (Acknowledgement)     | 38                         | &     | 70 | F | 102 | f |
| 07                       | BEL  | (Bell)                | 39                         | '     | 71 | G | 103 | g |
| 08                       | BS   | (Backspace)           | 40                         | (     | 72 | H | 104 | h |
| 09                       | HT   | (Horizontal Tab)      | 41                         | )     | 73 | I | 105 | i |
| 10                       | LF   | (Line feed)           | 42                         | *     | 74 | J | 106 | j |
| 11                       | VT   | (Vertical Tab)        | 43                         | +     | 75 | K | 107 | k |
| 12                       | FF   | (Form feed)           | 44                         | ,     | 76 | L | 108 | l |
| 13                       | CR   | (Carriage return)     | 45                         | -     | 77 | M | 109 | m |
| 14                       | SO   | (Shift Out)           | 46                         | .     | 78 | N | 110 | n |
| 15                       | SI   | (Shift In)            | 47                         | /     | 79 | O | 111 | o |
| 16                       | DLE  | (Data link escape)    | 48                         | 0     | 80 | P | 112 | p |
| 17                       | DC1  | (Device control 1)    | 49                         | 1     | 81 | Q | 113 | q |
| 18                       | DC2  | (Device control 2)    | 50                         | 2     | 82 | R | 114 | r |
| 19                       | DC3  | (Device control 3)    | 51                         | 3     | 83 | S | 115 | s |
| 20                       | DC4  | (Device control 4)    | 52                         | 4     | 84 | T | 116 | t |
| 21                       | NAK  | (Negative acknowl.)   | 53                         | 5     | 85 | U | 117 | u |
| 22                       | SYN  | (Synchronous idle)    | 54                         | 6     | 86 | V | 118 | v |
| 23                       | ETB  | (End of trans. block) | 55                         | 7     | 87 | W | 119 | w |
| 24                       | CAN  | (Cancel)              | 56                         | 8     | 88 | X | 120 | x |
| 25                       | EM   | (End of medium)       | 57                         | 9     | 89 | Y | 121 | y |
| 26                       | SUB  | (Substitute)          | 58                         | :     | 90 | Z | 122 | z |
| 27                       | ESC  | (Escape)              | 59                         | ;     | 91 | [ | 123 | { |
| 28                       | FS   | (File separator)      | 60                         | <     | 92 | \ | 124 |   |
| 29                       | GS   | (Group separator)     | 61                         | =     | 93 | ] | 125 | } |
| 30                       | RS   | (Record separator)    | 62                         | >     | 94 | ^ | 126 | ~ |
| 31                       | US   | (Unit separator)      | 63                         | ?     | 95 | _ |     |   |
| 127                      | DEL  | (Delete)              |                            |       |    |   |     |   |

| Extended ASCII characters |   |     |   |     |   |     |      |  |  |
|---------------------------|---|-----|---|-----|---|-----|------|--|--|
| 128                       | Ç | 160 | á | 192 | Ł | 224 | Ó    |  |  |
| 129                       | ü | 161 | í | 193 | ł | 225 | ô    |  |  |
| 130                       | é | 162 | ó | 194 | ŧ | 226 | õ    |  |  |
| 131                       | â | 163 | ú | 195 | — | 227 | ö    |  |  |
| 132                       | ä | 164 | ñ | 196 | — | 228 | ø    |  |  |
| 133                       | à | 165 | Ñ | 197 | † | 229 | õ    |  |  |
| 134                       | á | 166 | ª | 198 | â | 230 | µ    |  |  |
| 135                       | ç | 167 | º | 199 | Ã | 231 | þ    |  |  |
| 136                       | ê | 168 | ¿ | 200 | ä | 232 | þ    |  |  |
| 137                       | ë | 169 | ® | 201 | å | 233 | ú    |  |  |
| 138                       | è | 170 | ™ | 202 | Œ | 234 | û    |  |  |
| 139                       | ï | 171 | ½ | 203 | Ŧ | 235 | ü    |  |  |
| 140                       | î | 172 | ¼ | 204 | ŷ | 236 | ý    |  |  |
| 141                       | ï | 173 | ⅓ | 205 | ÿ | 237 | ÿ    |  |  |
| 142                       | Ä | 174 | ¼ | 206 | ÿ | 238 | ÿ    |  |  |
| 143                       | Å | 175 | ½ | 207 | ÿ | 239 | ÿ    |  |  |
| 144                       | É | 176 | ¾ | 208 | ð | 240 | ≡    |  |  |
| 145                       | æ | 177 | ¾ | 209 | Ð | 241 | ±    |  |  |
| 146                       | Æ | 178 | ¾ | 210 | É | 242 | ≡    |  |  |
| 147                       | ó | 179 | ¾ | 211 | Ê | 243 | ¾    |  |  |
| 148                       | ô | 180 | ¾ | 212 | Ë | 244 | ¾    |  |  |
| 149                       | õ | 181 | ¾ | 213 | Ì | 245 | ¾    |  |  |
| 150                       | ö | 182 | ¾ | 214 | Í | 246 | ¾    |  |  |
| 151                       | ù | 183 | ¾ | 215 | Î | 247 | ¾    |  |  |
| 152                       | ÿ | 184 | ¾ | 216 | Ï | 248 | ¾    |  |  |
| 153                       | Ö | 185 | ¾ | 217 | Ĵ | 249 | ¾    |  |  |
| 154                       | Ů | 186 | ¾ | 218 | Ű | 250 | ¾    |  |  |
| 155                       | ø | 187 | ¾ | 219 | Ų | 251 | ¾    |  |  |
| 156                       | £ | 188 | ¾ | 220 | Ŷ | 252 | ¾    |  |  |
| 157                       | Ø | 189 | ¾ | 221 | ŷ | 253 | ¾    |  |  |
| 158                       | × | 190 | ¾ | 222 | ı | 254 | ¾    |  |  |
| 159                       | ƒ | 191 | ¾ | 223 | ı | 255 | nbsp |  |  |

- ord('A') ⇒ return integer Unicode code point for char (e.g. 65)
- chr(65) ⇒ return string representing char at that point (e.g. 'A')

STRING CONSTANTS (MODULE : STRING) [import string](https://docs.python.org/3/library/string.html)

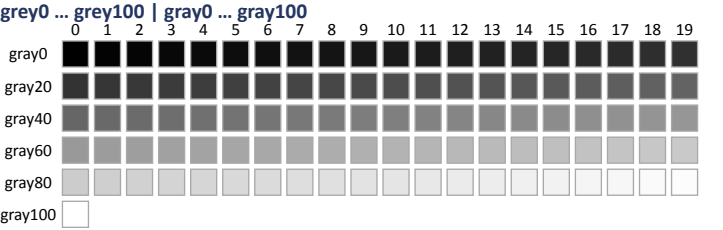
|                        |  |
|------------------------|--|
| string.ascii_lowercase | all lowercase letters: 'abcdefghijklmnopqrstuvwxyz'  |
| string.ascii_uppercase | all uppercase letters: 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'  |
| string.ascii_letters   | concatenation of the ascii_lowercase and ascii_uppercase constants                                       |
| string.digits          | the string '0123456789'  |
| string.hexdigits       | the string '0123456789abcdefABCDEF'  |
| string.octdigits       | the string '01234567'  |
| string.punctuation     | string of ASCII punctuation chars: '!"#\$%&'()*+,-./:;<=>?@[\\]^_`{ }~.'                                 |
| string.whitespace      | string containing all ASCII whitespace (space, tab, linefeed, return, formfeed, and vertical tab)        |
| string.printable       | string of printable ASCII characters (combination of digits, ascii_letters, punctuation, and whitespace) |

PYTHON SETS ⇒ { }

Set items are unordered, unchangeable and do not allow duplicate values. Items can be added or deleted.

```
s = set() create empty set
s = {"ap", "ban", "ch"} create set with items
```

PYGAME COLORS [https://github.com/pygame/pygame/blob/main/src\\_py/colordict.py](https://github.com/pygame/pygame/blob/main/src_py/colordict.py)



ÉCRIRE UNE COMMANDE PYTHON SUR PLUSIEURS LIGNES

- Utiliser la continuité implicite des lignes au sein des parenthèses/crochets/accolades
- Utiliser en dernier recours le backslash "\ " (= line break)

| continuité implicite   | backslash  |
|--|--|
| <pre>def __init__(self, a, b, c,     d, e, f, g): output = (a + b + c     + d + e + f) lst = [a, b, c,     d, e, f] if (a &gt; 5     and a &lt; 10):</pre> | <pre>output = a + b + c \     + d + e + f if a &gt; 5 \     and a &lt; 10:</pre> |

DIVERS

- Distance entre 2 points  $P_1(x_1, y_1)$  et  $P_2(x_2, y_2)$  :  $d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$