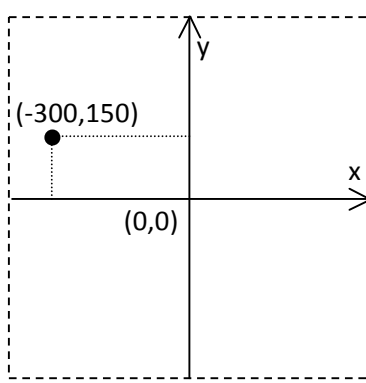


<p align="center"><b>Opérations sur les types</b></p> <p><code>type()</code> : pour connaître le type d'une variable  <code>int()</code> : permet la transformation en un entier  <code>float()</code> : permet la transformation en flottant  <code>str()</code> : permet la transf. en chaîne de caractères</p>	<p align="center"><b>Définition d'une fonction</b></p> <pre>def nom_fonction(arg) :     instructions     return valeurs_ou_variables</pre>
<p align="center"><b>Instructions de lecture et écriture console</b></p> <pre>print(a1,a2,..., an, sep=xx, end = yy)</pre> <ul style="list-style-type: none"> <li>• Pour imprimer une suite d'arguments de a1 à an</li> <li>• <code>sep</code> : permet de définir le séparateur affiché entre chaque argument (par défaut " ")</li> <li>• <code>end</code> : permet de définir le séparateur qui sera affiché entre chaque argument (optionnel, par défaut : saut de ligne)</li> </ul> <pre>res = input(message)</pre> <ul style="list-style-type: none"> <li>• Pour lire une suite de caractères au clavier terminée par &lt;Enter&gt;</li> <li>• La chaîne de caractères résultante est affectée (ici à la variable <code>res</code>).</li> <li>• l'argument est optionnel : c'est un message explicatif destiné à l'utilisateur</li> </ul>	<p align="center"><b>Instructions conditionnelles</b></p> <pre>if condition :     instructions  if condition :     instructions else :     instructions  if condition1 :     instructions elif condition2 :     instructions else :     instructions</pre>
<p align="center"><b>Opérateurs arithmétiques</b></p> <p><code>+</code> : addition, <code>-</code> : soustraction  <code>*</code> : multiplication, <code>**</code> : puissance,  <code>/</code> : division, <code>//</code> : division entière (quotient),  <code>%</code> : reste de la division entière</p>	<p align="center"><b>Turtle : instructions et fenêtre</b></p> <p>Curseur initialement en (0,0), dirigé vers la droite.</p> <pre>reset() : efface tout goto(x, y) : place le crayon au point x, y forward(d) : avance de d backward(d) : recule de d up() : relève le crayon down() : abaisse le crayon left(a) : tourne à gauche de a° right(a) : tourne à droite de a°</pre> 
<p align="center"><b>Opérateurs booléens</b></p> <p><code>and</code> : et logique <code>or</code> : ou logique <code>not</code> : négation</p>	
<p align="center"><b>Opérateur de comparaison</b></p> <p><code>==</code> égalité <code>!=</code> différence  <code>&lt;</code> inférieur, <code>&lt;=</code> inférieur ou égal  <code>&gt;</code> supérieur, <code>&gt;=</code> supérieur ou égal</p>	
<p align="center"><b>Opérateurs sur les chaînes de caractères</b></p> <p><code>len(s)</code> : renvoie la longueur de la chaîne <code>s</code>  <code>s1 + s2</code> : concatène les chaînes <code>s1</code> et <code>s2</code>  <code>s * n</code> : pour construire la répétition de <code>n</code> fois <code>s</code>  exemple : <code>"ta" * 3</code> donne <code>"tatata"</code></p>	
<p align="center"><b>Itération tant que</b></p> <pre>while condition :     instructions</pre>	
<p align="center"><b>Itération for, et range</b></p> <pre>for e in list_chaine_ou_dico :     instructions</pre> <p>Itère les instructions avec <code>e</code> prenant chaque valeur dans la liste, chaîne ou dictionnaire.</p>	<pre>for var in range (deb, fin, pas) :     instructions</pre> <p><b>range(a)</b> : séquence des valeurs [0, a[  <b>range (b,c)</b> : séquence des valeurs [b, c[ (pas de 1)  <b>range (b, c, g)</b> : idem avec un pas de g  <b>range(b,c,-1)</b> : valeurs de b(incl.) à c (excl.) , pas -1</p>

Listes	Dictionnaires
<p><b>list_exple = []</b> Création d'une liste vide</p>	<p><b>dico_exple = {}</b> Création d'un dictionnaire vide</p>
<p><b>list_exple = [e1, e2, e3]</b> Création d'une liste, ici à 3 éléments e1, e2, et e3</p>	<p><b>dico_exple = { c1:v1, c2:v2, c3:v3 }</b> Création d'un dictionnaire, ici à 3 entrées</p>
<p><b>list_exple[i]</b> obtenir l'élément à l'index <i>i</i> (<i>i</i> ≥ 0) Les éléments sont indexés à partir de 0 Si <i>i</i> &lt; 0, les éléments sont accédés à partir de la fin de la liste. Ex : <b>list_exple[-1]</b> permet d'accéder au dernier élément de la liste</p>	<p><b>e = dico_exple[c1]</b> Les valeurs du dictionnaire sont accessibles par leurs clés. Ici, <b>e</b> prendra la valeur <b>v1</b>. Provoque une erreur si la clé n'existe pas.</p>
<p><b>list_exple.append(element)</b> Ajout d'un seul élément à la fin de la liste</p>	<p><b>dico_exple[c3] = v3</b> Pour ajouter une nouvelle valeur au dictionnaire (<b>ici v3</b>) avec une clé (<b>ici c3</b>). Si la clé existe déjà, la valeur associée est modifiée.</p>
<p><b>list_exple.extend(liste2)</b> Ajout de tous les elem. de la liste <b>liste2</b> à la fin de la liste</p>	<p><b>del dico_exple[C3]</b> Pour supprimer une association dans le dictionnaire. La clé doit exister.</p>
<p><b>list_exple.insert(i, element)</b> Ajout d'un seul élément à l'index <i>i</i></p>	<p><b>c in dico_exemple</b> Pour vérifier l'existence d'une clé dans le dictionnaire. Vaut <b>True</b> ou <b>False</b>.</p>
<p><b>res = list_exple.pop(index)</b> retire l'élément présent à la position <b>index</b> et le renvoie, ici dans la variable <b>res</b>.</p>	<p><b>dic2 = dico_exple</b> crée un synonyme (donne un 2<sup>ème</sup> nom au dictionnaire)</p>
<p><b>list_exple.remove(element)</b> retire l'élément donné (le premier trouvé)</p>	<p><b>dic3 = dict(dico_exple)</b> Pour créer une copie (= un clone)</p>
<p><b>len(list_exple)</b> Pour avoir le nombre d'éléments d'une liste</p>	<hr/> <b>Gestion des fichiers</b>
<p><b>l2_exple = list_exple</b> crée un synonyme (donne un 2<sup>ème</sup> nom à la liste)</p>	<p><b>f=open('data.txt')</b> Pour ouvrir un fichier en lecture seule <b>f=open('data.txt', 'w')</b> Pour ouvrir un fichier en écriture</p>
<p><b>l3_exple = list(list_exemple)</b> Pour créer une copie (= un clone)</p>	<p><b>texte = f.read()</b> pour lire tout le fichier en une seule fois <b>lignes = f.readlines()</b></p>
<p><b>elem in list_exple</b> Pour tester si un élément est dans une liste. Vaut <b>True</b> ou <b>False</b>.</p>	<p>Lire en une fois toutes les lignes du fichier et les stocker dans une liste (un élément = une ligne) <b>for ligne in f:</b> <b>instructions</b></p>
<p><b>list_exple.shuffle()</b> (module random) Mélange la liste (effet de bord).</p>	<p>Lire le fichier ligne par ligne dans une boucle <b>for</b> <b>f.write(texte)</b></p>
<hr/> <b>Chaîne de caractères (complément)</b>	<p>Pour écrire dans un fichier (<b>texte</b> doit oblig. être un <b>str</b>). Ne saute pas de ligne automatiquement à la fin du texte. <b>\n</b> code un saut de ligne.</p>
<p><b>ch.split(arg)</b> où <b>ch</b> est une chaîne de carac. Retourne la liste des mots ici de <b>ch</b>, en coupant à chaque occurrence de <b>arg</b> (par défaut <b>arg = " "</b> (un espace))</p>	<p><b>f.close()</b> Pour fermer un fichier</p>