

TYPE et CONVERSIONS DE VARIABLES

Types courants : Entier (*int*) Réel (*float*) Texte (*str*) Booléen (*bool*)

Déclaration de variables :

T = "Bonjour" Assigne à la variable texte (string) *T* le texte « *Bonjour* »

N = 2 Assigne à la variable entière (int) *N* la valeur 2

Conversion de variables :

T = **str**(**N**) transforme un entier (int) ou réel (float) *N* en texte (string)

N = **int**(**T**) ou **N** = **float**(**T**) transforme un texte *T* (string) en entier (int) ou réel (float)

MEMENTO

PYTHON 3

PHYSIQUE-CHIMIE

O. Chaumette - Lycée JP SARTRE - 69500 BRON

ENTREES et SORTIES : demander une valeur à l'utilisateur et afficher à l'écran.

Entrée au clavier :

Un_Texte = **input**("question") Pose « *question* » à l'utilisateur et met la réponse dans la variable « *texte* » : **Un_Texte**

Un_Entier = **int**(**input**("question")) Réponse mise dans la variable « *entière* » : **un_entier**

Sortie écran :

print("Texte", *variable*) écrit sur l'écran *Texte* suivi du contenu de *Variable* séparés par un espace

print(*variable1* + *variable2*) Uniquement si *variable1* et *variable2* sont de même type

print(*variable1*, *variable2*) **type nombre** : les ajoute **type texte** : les colle (les « concatène »)

print("%.2e"%*N*) Ecrit le réel (ou entier) *N* en écriture scientifique avec 2 décimales (donc 3 chiffres significatifs)

CALCULS /ARRONDIS

round(*x*) arrondit un "réel" *x* vers l'entier le plus proche

round(*x*, *n*) arrondit un "réel" à la décimale *n*. *n* négatif permet un arrondi à la dizaine, centaine... près.

****** marque l'exposant et a la priorité sur +, -, *, /

pow(*x*, *y*) renvoie *x* à la puissance *y*, équivaut à *x* ** *y*

abs() renvoie la valeur absolue d'un nombre (sans le signe)

TESTS et CONDITIONS :

Test simple:

if *Condition*:
 Instructions si « Condition » est vraie

Test avec SINON (else):

if *Condition*:
 Instructions si « Condition » est vraie
else:
 Instructions si « Condition » est fausse

Test avec SINON SI (elif):

if *Condition*:
 Instructions si « Condition » est vraie
elif *Condition2*:
 Instructions si « Condition2 » est vraie
else:
 Instructions si « Condition1 et 2 » sont fausses

Test avec conditions multiples:

if *Condition1* **and/or** *Condition2*:
 Instructions

and: condition 1 ET 2 respectées

or: condition 1 OU 2 respectées

Opérateurs dans les conditions: ATTENTION le signe = est réservé à l'affectation de variables

== :égal **!=** :différent **not** : contraire de la condition

> (ou **<**):supérieur (ou inférieur) **>=** (ou **<=**):sup (ou inf) ou égal

BOUCLES :

Boucle FOR générale: Dans le cas où on connaît le nombre de répétitions

for *Variable* **in** *Ensemble de valeurs*:
 Instructions *Variable* va prendre toutes les valeurs de « ensemble de valeurs »
 bien penser à l'indentation !!!

Boucle FOR avec des nombres:

for *Compteur* **in** **range**(*Nombre*): *Compteur* varie de 0 à *Nombre-1*

for *Compteur* **in** **range**(*début*, *fin*): *Compteur* varie de *début* à *fin-1*

for *Compteur* **in** **range**(*début*, *fin*, *pas*): *Compteur* varie de *début* à *fin-1* par sauts de *pas*

Boucle WHILE (tant que) : Dans le cas où on ne connaît pas le nombre de répétitions

while *Condition*: bien penser aux « : » et à l'indentation

Instructions tant que « Condition » est vraie

Modifier la variable intervenant dans la condition sinon la boucle serait infinie !

MODULE RANDOM : hasard

Déclaration :`import random`**Fonctions :**`random.choice (Ma_Liste)` choisit un élément de la liste `Ma_Liste``random.randrange (borne1 , borne2)` renvoie un entier au hasard entre `borne1` (inclusive) et `borne2` (exclue)**MODULE NUMPY : FONCTIONS COURANTES**`import numpy as np` importe Numpy sous l'alias `np`

Fonctions trigonométriques			Fonctions trigonométriques inverses			π	\sqrt{x}
<code>np.sin(x)</code>	<code>np.cos(x)</code>	<code>np.tan(x)</code>	<code>np.arcsin(x)</code>	<code>np.arccos(x)</code>	<code>np.arctan(x)</code>	<code>np.pi</code>	<code>np.sqrt(x)</code>
ln(x) et e^x	log(x) et 10^x	Val absolue	Signe (renvoie 1 si positif et -1 si négatif)		arrondi à ndécimales		y^x
<code>np.log(x)</code> <code>np.exp(x)</code>	<code>np.log10(x)</code> <code>10**x</code>	<code>np.abs(x)</code>	<code>np.sign(x)</code>		<code>np.around(x, n)</code>		<code>y**x</code>

AFFICHAGE avec MATPLOTLIB`import matplotlib.pyplot as plt` importe pyplot sous l'alias `plt`**Affichage d'un point ou des valeurs d'un tableau Numpy**`plt.plot(X, Y, "style")` Place un point en `X,Y` avec un certain `style`. `X` et `Y` peuvent être des tableaux Numpy`plt.plot([X1, X2], [Y1, Y2], "style")` Trace un segment entre les points de coordonnées (`X1,Y1`) et (`X2,Y2`)`plt.plot(X, Y, "style", options)` Point en `X,Y` avec `options` :

`linewidth= valeur` épaisseur du trait
`label= "texte"` texte à lier à la courbe (s'utilise avec `legend()`)
`markersize= valeur` taille des marques (points)

Styles disponibles : à placer entre guillemets dans l'ordre : 1. Couleur 2. Type de point 3. Type de tracé

Couleurs						Type de points tracés					Tracé	
<code>r</code>	<code>b</code>	<code>g</code>	<code>k</code>	<code>m</code>	<code>c</code>	<code>o</code>	<code>.</code>	<code>x</code>	<code>+</code>	<code>v</code>	<code>-</code>	<code>--</code>
Rouge	Bleu	vert	noir	magenta	cyan	Gros point	Petit point	Croix	Croix +	Triangle	Points reliés	Points reliés en pointillé

Exemple de style : `"rx-"`**Affichage d'un texte**`plt.text(X, Y, "Texte à afficher", color='C')` Affiche le texte aux coordonnées `X,Y` de couleur `C``plt.legend()` Affiche le texte de l'option `label` de `plot` (utile si plusieurs courbes)**Affichage d'un vecteur**`plt.quiver(X, Y, Vx, Vy, color='C', scale=20)` vecteur en (`X, Y`) de coordonnées (`Vx, Vy`) de couleur `C`**Affichage de la fenêtre MATPLOTLIB (à placer tout à la fin)**`plt.show()`**GESTION DE LA FENÊTRE ET DES AXES MATPLOTLIB**

`plt.figure("NOM de la Fenetre")` Donne un nom à la fenêtre **à placer au début**
`plt.title("TITRE du graphique")` Donne un titre au graphique
`plt.xlabel("NOM de l'axe des X")` Donne un nom à l'axe des abscisses (`ylabel` pour les ordonnées)
`plt.invert_xaxis()` Change le sens de l'axe X (`invert_yaxis()` pour l'axe Y)
`plt.axis([Xmin, Xmax, Ymin, Ymax])` Définit les valeurs min et max des abscisses et ordonnées
`plt.grid()` Affichage de la grille

TABLEAUX NUMPY`T = np.array([valeur1, valeur2, valeur3, etc...])` créé un tableau contenant `valeur1, valeur2, etc...``T[index]` Renvoie l'élément situé à l'emplacement `index` (qui commence à 0)`T[-1]` Renvoie le **dernier** élément (si -2 : l'avant dernier etc...)`T[pos] = "a"` met 'a' à la position `pos` (début=0) (**en écrasant** l'él qui s'y trouve)`len(T)` Renvoie le nombre d'éléments du tableau. C'est un **entier**.`for Element in T:` `Element` prend successivement le contenu de chaque élément du tableau