MEMENTO PYTHON 3 POUR LE LYCÉE

VARIABLES

Mettre (=assigner) une valeur/un texte dans une variable

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

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

Noms de variables

pas d'accent - pas d'espace - pas d'instruction python - différence majuscule/minuscule (casse) importante.

Piège : « lambda » (tout en minuscule) est une instruction python

AFFICHAGE d'un texte, d'une variable…

`print("Longueur =",L,"mètres")` écrit dans la console "*Longueur=*" suivi du contenu de *L* suivi de "*mètres*"

Mise en forme

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

`print("bonjour\nle monde")` `\n` saute une ligne au milieu d'un texte

ATTENTION !

, (virgule) se tape : **. (point)**
(exple : 3,14 se tape `3.14`)

Multiplier se tape : *****

Puissances de 10/exposants

`2E4` signifie $2x10^{4}$ `1E-2` signifie $1x10^{-2}$

`2**4` signifie 2^{4} `10**-4.8` signifie $10^{-4,8}$

Lettres grecques il n'est pas conseillé de mettre une lettre grecque comme nom de variable mais elle peut être affichée :

`print(chr(945))` affiche la lettre α (qui correspond au code 945)

`print("L'angle",chr(945),"vaut",45, "°")` affiche : *L'angle α vaut 45 °*

code des lettres grecques :

945	946	947	948	949	951	952	954	955	956	957	960	961	963	964	965	966	968
α	β	γ	δ	ε	η	θ	κ	λ	μ	ν	π	ρ	σ	τ	υ	φ	ψ

969	916	923	928	931	934	936	937	8771	8800	8804	8805	8734
ω	Δ	Λ	Π	Σ	Φ	Ψ	Ω	≃	≠	≤	≥	∞

TESTS et CONDITIONS :

Test simple: **bien penser aux « : » et à l'indentation**

```
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 Dans le cas où on connaît le nombre de répétitions

```
for Compteur in range(Nombre):
    Instructions
```

Compteur varie de ***0*** à ***Nombre-1*** **bien penser aux « : »**

bien penser à l'indentation !!!

`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:
    Instructions tant que « Condition » est vraie
    Modifier la variable intervenant dans la condition
```

bien penser aux « : »

bien penser à l'indentation !!!

sinon la boucle serait infinie !

IMPORTATION DES MODULES

À placer tout au début du programme et une seule fois

`import random`	importe le module pour gérer le HASARD
`import numpy as np`	Numpy sous l'alias **np** pour créer des tableaux/fonctions maths
`import matplotlib.pyplot as plt`	importe pyplot sous l'alias **plt** pour afficher des courbes

HASARD : le module RANDOM

`random.randrange(`*borne1*`,`*borne2*`)` renvoie un entier au hasard entre *borne1* (incluse) et *borne2* (exclue)

Fonctions MATHÉMATIQUES courante avec le module NUMPY

Fonctions trigonométriques			Fonctions trigonométriques inverses			π	$\sqrt{x}$
`np.sin(x)`	`np.cos(x)`	`np.tan(x)`	`np.arcsin(x)`	`np.arccos(x)`	`np.arctan(x)`	`np.pi`	`np.sqrt(x)`

ln(x) et e^x	log(x) et 10^x	Val absolue	Signe (renvoie 1 si positif et -1 si négatif)	arrondi à n décimales	y^x
`np.log(x)` `np.exp(x)`	`np.log10(x)` `10**x`	`np.abs(x)`	`np.sign(x)`	`np.around(x,n)`	`y**x`

TABLEAUX de données avec le module NUMPY

Création de tableaux *en physique-chimie, un tableau = une grandeur*

`T = np.array([`*val1*`,` *val2*`,` *val3* `,` *etc…*`])` créé «à la main» un tableau contenant *val1, val2, etc…*

`T = np.linspace(`*début*`,`*fin*`,`*n*`)` créé automatiquement un tableau de *n* valeurs entre *début* et *fin* (inclus)

`T = np.arange(`*début*`,`*fin*`,`*pas*`)` créé automatiquement un tableau de *début* et ***fin-1*** par pas de *pas*

`U = 200*I` créé un nouveau tableau U dont les éléments valent 200 fois ceux du tableau I

Manipulation de tableaux

`T[`*n*`]`	Valeur du *(n+1)* ème élément du tableau T (situé à l'index *n*. Les index commencent à 0)
`T[-1]`	Valeur du dernier élément (si -2 : l'avant dernier etc…)
`T = np.append(T,N)`	Ajoute la valeur de N à la fin du tableau T
`T[`*n*`] = "a"`	met `"a"` à l'index *n* (début=0) (**en écrasant** l'élt 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

TRACÉ DE COURBES avec le module MATPLOTLIB

Tracer la courbe Y en fonction de X

`plt.plot(`***X***`,`***Y***`,"`***style***`")` X et Y sont des tableaux Numpy **de même taille** (ils peuvent aussi être des nombres)

Styles disponibles : à placer entre guillemets *dans l'ordre* : **1. Couleur 2. Type de point 3. Type de tracé** Exemple de style : `"rx-"`

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

Ajouter une légende à une courbe

`plt.plot(`***X***`,` ***Y***`,"`***style***`",label="`***texte***`")` Affiche la courbe et une légende associée au tracé : *texte*

`plt.legend()` À écrire une seule fois. Indispensable pour afficher le texte de l'option `label` de `plot`

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*** (r,b,g etc…)
scale : échelle à choisir empiriquement.

Titre des axes, du graphique, grille…

`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.axis([`***Xmin***`,` ***Xmax***`,` ***Ymin***`,` ***Ymax***`])` Définit les valeurs min et max des abscisses et ordonnées

`plt.grid()` Affichage de la grille

Affichage de la fenêtre MATPLOTLIB ***INDISPENSABLE***

`plt.show()` ***à placer tout à la fin du code***

Sauver l'image de la courbe :

Cliquer sur l'icône de la fenêtre Matplotlib : 

ERREURS FRÉQUENTES

- Parenthèses/crochets pas fermés
- Virgule dans un nbre à la place d'un point
- x à la place de * pour multiplier
- Indentation pas respectée (conditions ou boucles)
- Oubli des « : » dans boucles ou conditions
- « = » au lieu de « == » dans une condition
- Fenêtre Matplotlib pas fermée