Modélisation numérique en physique

Bilan (correction)
: Les bases de Python

Ressources : https://phys-mod.github.io/

Diata Traore (diata.traore@sorbonne-universite.fr)

2021/22

Objectifs pédagogiques

- *Décrire, affecter, manipuler, déterminer et convertir les différents types de variables*
- *Décrire et utiliser les opérateurs spécifiques à chaque type de variable*
- *Créer et manipuler les listes à l'aide des méthodes spécifiques*
- *Créer et manipuler les tableaux Numpy à l'aide des fonctions et des méthodes spécifiques*
- *Décrire et utiliser l'indexation des tableaux Numpy pour extraire et créer de nouveaux tableaux*
- *Créer une représentation discrète d'une fonction mathématique à l'aide des tableaux Numpy*
- *Créer un graphique à partir de tableaux Numpy en choisissant les intervalles de visualisation et le type d'axes*
- *Manipulation du type datetime64 de Numpy*

(D'après la liste des objectifs définis par l'équipe pédagogique lors de la création du format actuel de l'ue.)

Objectif #1 : décrire, affecter, manipuler, déterminer et convertir les différents types de variables :

Affecter la valeur 10.0 à la variable v :

```
v = 10.0
```

Afficher la variable v :

```
print(v) ; print("v =", v) ; print("v= " + str(v))
```

Les différents types de variables :

Type	Catégorie Python	Exemple
Entier	int	4
Nombre à virgule	float	4.0
chaîne de caractère	str	quatre
Liste	list	[1, deux, 3.0]

Afficher le type de v :

```
print(type(v))
```

Convertir v en entier :

```
int(v)
```

Objectif #2 : décrire et utiliser les opérateurs spécifiques à chaque type de variable

Opérations	Opérateurs	Exemples	
Addition, Soustraction	+	3 + 4 = 7 3 + 4.0 = 7.0	“Hello “ + “world” = “Hello world”
Soustraction	-	3 - 4 = -1 3 - 4.0 = 1.0	“Hello ” - “world” = error
Multiplication	*	3 * 4 = 12 3 * 4.0 = 12.0	“Hello “ * 2 = “Hello Hello “
Division	/	3 / 4 = 0.75 3 / 4.0 = 0.75	“Hello “ / 2 = erro
Quotient de la division entière	//	3 // 4 = 0 3.0 // 4.0 = 0.0	
Reste de la division entière modulo	%	3 % 4 = 3 3.0 % 4 = 3.0 3.0 % 4.0 = 3.0	
Puissance	x**n ou pow(x,n)	3**2 = 9 3.0**2 = 9.0 pow(3,2.0) = 9.0	“Hello “ ** 2 = error

Objectif #3 : créer et manipuler les listes à l'aide des méthodes spécifiques

Créer une liste L_1 contenant les éléments 1, 3.0, a :

```
L_1 = [1, 3.0, "a"]
```

Quel est l'indice de l'élément a ?

```
L.index("a") ;  2
```

Soit la liste L_2 = [1.0, 5, [3.5, 6]],
concaténer les listes L_1 et L_2 :

```
L_3 = L_1 + L_2
```

Quels sont les éléments de la liste L_4 = L_3[1:5] ?

```
L_4 = [3.0, "a", 1.0, 5]
```

Quels sont les éléments de la liste L_5 = L_3[:5] ?

```
L_5= [1, 3.0, "a", 1.0, 5]
```

Quels sont les éléments de la liste L_6 = L_3[3:] ?

```
L_6= [1.0, 5, [3.5, 6]]
```

Quel est l'élément i = L_3[5][1] ? Son type ?

```
i = 6, int
```

Quel est l'élément j = L_3[5] ? Son type ?

```
j = [3,5, 6], list
```

Supprimer le 2ème élément de L_3 :

```
L_3.pop(1) ou L_3.remove(value)
```

Afficher la longueur de la liste L_3 :

```
len(L_3)
```

Objectif #4 : créer et manipuler les tableaux Numpy à l'aide des fonctions et des méthodes spécifiques

Importer le module Numpy :

```
import numpy as np
```

Convertir une liste L = [[“a”, “b”, “c”],[“d”,”e”,”f”]] en tableau Numpy T :

```
T = np.array(L)
```

Nombre de dimensions de T :

```
np.dim(T)
```

Taille de T dans chaque dimension :

```
np.shape(T)
```

Nombre d'éléments de T :

```
np.size(T)
```

Copier le tableau T dans T_2 :

```
T_2 = np.copy(T)
```

Objectif #5 : décrire et utiliser l'**indexation** des tableaux Numpy pour extraire et créer de nouveaux tableaux

On considère le tableau A = np.array([4, 6, 1, 23, 3, 8, 9]).

Créer le tableau Tranche contenant 1 élément sur 2 du tableau A :

```
Tranche = A[::2]
```

On considère les tableaux A = np.array([[-1, 6, 3],[4,2,8]]) et B=np.array([3,6,0]).

```
A * B = array([ [-3  36  0] [12 12 0] )
```

```
A + B = array([ [ 2 12  3] [ 7  8  8] ])
```

```
A[0,:] + B = array([ 2 12  3])
```

```
A[:,0] + B[0] = array([2, 9, 6])
```

Objectif #6 : créer une *représentation discrète* d'une fonction mathématique à l'aide de tableaux Numpy

Soit la fonction $y(x) = e^{-x^2}$.
Ecrire les tableaux des valeurs de x et y pour x$\in$[-50.0 , 50.0], par pas de p=0.1 :

```
import numpy as np
```

```
x_min = -50.0
x_max = 50.0

#Nombre de points :
N = int((x_min - x_max)/2 + 1)
```

```
x = np.linspace(x_min , x_max , N)
```

```
y = np.exp(-x**2)
```

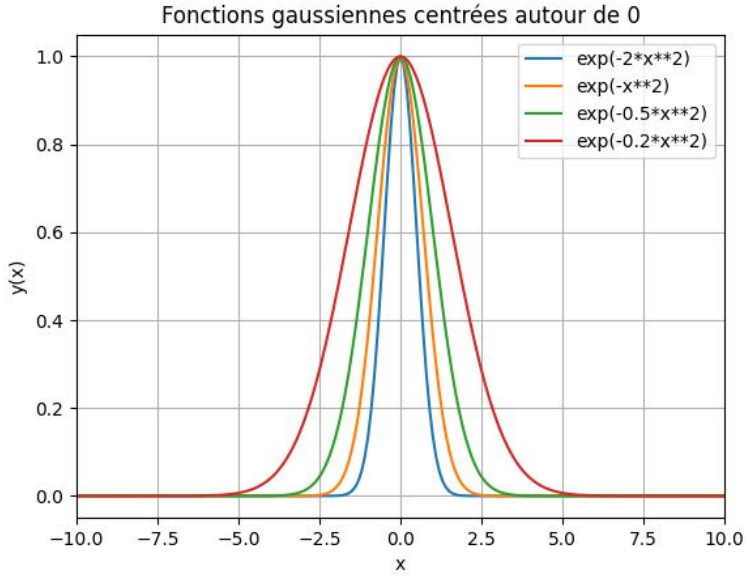
Objectif #7 : créer un graphique à partir de tableaux Numpy en choisissant les intervalles de visualisation et le type d'axes

Importer la bibliothèque ?????? , permettant de créer un graphique :

```
import matplotlib.pyplot as plt
```

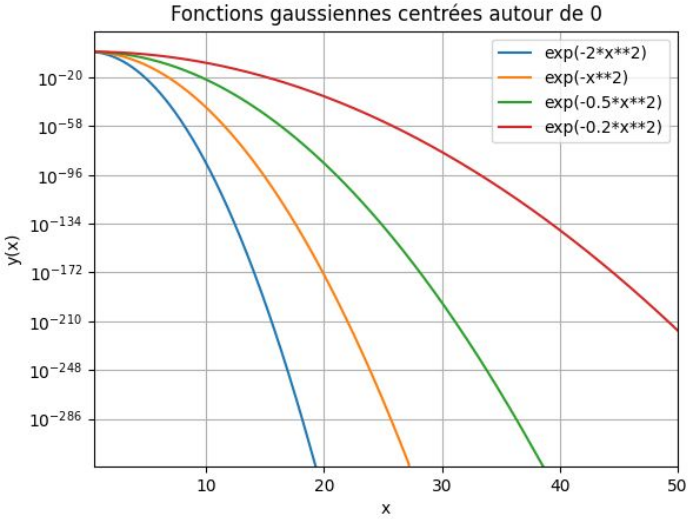
Tracer y_1(x), y_2(x), y_3(x) et y_4(x) *(voir tableau)*:

```
# Tracer :
plt.plot(x, y_1, x, y_2, x, y_3, x, y_4)
# Limiter l'axe des abscisses à [-10.0, 10.0] :
plt.xlim(-10.0 , 10.0)
# Ajouter un titre au graphique :
plt.title('Fonctions gaussiennes centrées autour de 0)
# Donner des titres aux axes :
plt.xlabel("x")
plt.ylabel("y")
# Ajouter la légende des courbes :
plt.legend([exp(-2*x**2) , exp(-x*2**2) , exp(-0.5*x**2),...])
# Afficher la grille :
plt.grid()
# Afficher le graphique :
plt.show()
```



Tracer en échelle logarithmique :

```
# log(x) / log(y) :
plt.loglog()
# x / log(y) (voir figure ci-contre):
plt.semilogx()
# log(x) / y :
plt.semilogy()
```



Tracer seulement les points, sans les lignes :

```
plt.plot(x , y , .’.’)
```

Objectif #8 : Savoir utiliser le type datetime64 de Numpy

Créer une variable date, de type datetime64 et lui attribuer la date du jour :

```
import numpy as np

date = np.datetime64(“2022-01-28”)
```

Nombre de jours entre date et le 28 février 2022 :

```
duree = np.datetime(“2022-02-28”) - date
```

Afficher l’année de la variable date :

```
np.datetime64(date, ‘Y’)
```

Créer un tableau de dates (avec la fonction arange) entre novembre 2021 et février 2022 :

```
dates = np.arange(‘2021-11’, ‘2022-02’,
dtype=’datetime64[D]’)
```

Quelques problèmes rencontrés :

Etu - 4 : Module numpy :

Calcul de **la moyenne**, de **la médiane** ou de **l'écart-type** d'un tableau de dimension 2 :

exemple :

```
villes = [[2240621, 105.40],    # Paris
          [852516, 240.62],     # Marseille
          [496343, 47.87],      # Lyon
          [453317, 118.30],     # Toulouse
          [343629, 71.92],      # Nice
          [291604, 65.19],      # Nantes
          [274394, 78.26],      # Strasbourg
          [268456, 56.88]]      # Montpellier

# Création du tableau Numpy à 2 dimensions
np_villes = np.array(villes)
```

mean_villes = np.mean(**np_villes, axis=0**) -> mean_villes = [moyenne_nb_habitants , moyenne_superficie]

OU

mean_nb_habitants = np.mean(**np_villes[:,0]**) -> mean_nb_habitants = valeur_1

mean_superficie = np.mean(**np_villes[:,1]**) -> mean_superficie = valeur_2

A faire pour la prochaine séance :

1) Mettre à jour le Trello.

2) M’envoyer une photo/scan/pdf de vos **notes de cours** par mail avant **Dimanche 06/02 à 18h**. (ou les apporter lundi 07/02 à la séance de td)