Module python lycee.py

FONCTIONS SAISIE CLAVIER

texte_demande :

texte_demande(prompteur: str) -> str

- *prompteur* est une chaîne de caractères
- Ouvre une fenêtre avec le message "prompteur" et attend une chaîne de caractères.
- retourne la chaîne de caractères entrée par l'utilisateur.

demande

demande(prompteur: str) -> float | int

- *prompteur* est une chaîne de caractères
- Ouvre une fenêtre avec le message "prompteur" et attend un nombre.
- retourne un nombre saisi par l'utilisateur ou une erreur quand le texte saisi n'est pas convertible en nombre.

liste_demande

liste_demande(prompteur: str) -> list

- *prompteur* est une chaîne de caractères
- Ouvre une fenêtre avec le message "prompteur" et attend une liste(list).
- retourne la list saisie par l'utilisateur ou une erreur quand le texte saisi n'est pas convertible en list.

FONCTIONS MATHÉMATIQUES

pgcd

pgcd(a: int, b: int) -> int

- *a* et *b* sont 2 entiers non nuls
- renvoie le Plus Grand Diviseur Commun des 2 nombres. La nullité de *a* et/ou *b* provoque une erreur.

abs2

abs2(x: float | int) -> float | int

- *x* est un nombre.
- Renvoie la valeur absolue du nombre *x*, c'est a dire sa distance à 0

reste

reste(a: int, b: int) -> int

- *a,b* sont des nombres entiers (*b* non nul)
- Cette fonction renvoie le reste de la division de *a* par *b*

quotient

quotient(a: int, b: int) -> int

- *a,b* sont des nombres entiers (*b* non nul)
- Cette fonction renvoie le quotient (entier) de la division de a par b

angleMode

angleMode(mode_angle: str = '') -> str

- cette fonction permet de définir l'unité d'angle utiliser par les fonctions trigonométriques
- du module "lycee".
- *mode_angle* : type d'unité d'angle à utiliser
- 'rad' les angles des fonctions trigonométriques seront pris comme des radians (défaut)\n
- 'deg' les angles des fonctions trigonométriques seront pris comme des degrés\n
- 'grd' les angles des fonctions trigonométriques seront pris comme des grades\n
- toute autre valeur provoque une erreur
- retourne la valeur précédente de mode et si *mode_angle* == "" la valeur actuelle de mode est retournée (str).
- La valeur par défaut est le **radian**.

cos

cos(angle: float) -> float

- retourne le cosinus de *angle* (réel) en fonction du mode choisi (défaut : radian)

sin

sin(angle: float) -> float

- retourne le sinus de *angle* (réel) en fonction du mode choisi (défaut : radian)

tan

tan(angle: float) -> float

- retourne la tangente de *angle* (réel) en fonction du mode choisi (défaut : radian)

acos

acos(value: float) -> float

- Retourne la valeur de l'angle telle que cos(angle) = *valeur* dans l'unité définie par angleMode().
- Le résultat est compris entre 0 et pi rd si mode("rad").
- Le résultat est compris entre 0 et 180° si mode("deg").
- Le résultat est compris entre 0 et 200 grd si mode("grd").

asin

asin(value: float) -> float

- Retourne la valeur de l'angle telle que sin(angle) = *valeur* dans l'unité définie par angleMode().
- Le résultat est compris entre 0 et pi rd si mode("rad").
- Le résultat est compris entre 0 et 180° si mode("deg").
- Le résultat est compris entre 0 et 200 grd si mode("grd").

atan

atan(value: float) -> float

- Retourne la valeur de l'angle telle que tan(angle) = *valeur* dans l'unité définie par angleMode().
- Le résultat est compris entre 0 et pi rd si mode("rad").
- Le résultat est compris entre 0 et 180° si mode("deg").
- Le résultat est compris entre 0 et 200 grd si mode("grd").

radians

radians(angle: float) -> float

- convertit l'*angle* dont l'unité est définie dans angleMode() en radian

degres

degres(angle: float) -> float

- convertit l'*angle* dont l'unité est définie dans angleMode() en degrés

grades

grades(angle: float) -> float

- convertit l'*angle* dont l'unité est définie dans angleMode() en grades

racine

racine(x: float | int) -> float

- x est un nombre positif
- Renvoie la racine carrée du nombre x

sqrt

idem **racine**

factoriel

factoriel(n: int) -> int

- *n* est un nombre entier positif
- Renvoie *n*! = *n* x (*n*-1) x ... x 3 x 2 x 1

floor

floor(x: float) -> int

- *x* est un nombre décimal.
- Retourne la partie entière du nombre x, c'est a dire le plus grand entier inférieur au réel *x*.

exp

exp(x: float | int) -> float

- *x* est un nombre décimal.
- Retourne l'image du nombre *x* par la fonction exponentielle (e^x).

ln

ln(x: float | int) -> float

- *x* est un nombre strictement positif.
- Retourne l'image du nombre *x* par la fonction logarithme népérien.

FONCTIONS STAT & PROBAS

binomial

binomial(n: int, p: int)

- n et p sont deux entiers.
- Retourne coefficient binomial p parmi n, c'est à dire le nombre de chemins de l'arbre réalisant p succès pour n répétitions.

randint

randint(min: int, max: int)

- *min* et *max* sont deux entiers.
- Renvoie un entier choisi de manière (pseudo)aléatoire et équiprobable dans l'intervalle [*min,max*].

choice

choice(liste: list) -> float | int | str

- *liste* est une list.
- Renvoie un élément de la liste list choisi (pseudo)aléatoirement et de manière équiprobable

random

random() -> float

- Pas d'argument.
- Renvoie au hasard un décimal de l'intervalle [0;1[

uniform

uniform(min:int|float, max:int|float)->float

- min et max sont deux nombres.
- Renvoie un nombre décimal choisi de manière (pseudo)aléatoire et uniforme de l'intervalle [min,max[.

intervalle

intervalle(debut: int, fin: int, pas: int = 'optionnel') -> list

- *debut, fin et pas* sont des entiers.
- Le paramètre pas est optionnel.
- Retourne une liste d'entiers :
  – De l'intervalle *[debut; fin*] si 2 paramètres sont renseignés
  – De l'intervalle [*debut; fin*] mais en réalisant une suite arithmétique de raison *pas* si les 3 paramètres sont renseignés.

range

range(debut: int, fin: int = 'optionnel', pas: int = 'optionnel')->list

- *debut*, *fin* et *pas* sont des entiers.
- Les paramètres *debut* et *pas* sont optionnels.
- Retourne une liste d'entiers :
  – De l'intervalle [0; *debut*[si un seul paramètre est renseigné.
  – De l'intervalle [*debut; fin*[si 2 paramètres sont renseignés
  – De l'intervalle [*debut; fin*[mais en réalisant une suite arithmétique de raison pas si les 3 paramètres sont renseignés.

sac_de_billes

sac_de_billes(billes:list, nb_tirages:int)->list[str|int|float]:

- billes : list décrivant les boules contenues dans le sac (couleur, numero,....)
- nb_tirages : int nombre de tirage avec remise effectués

- retourne une liste contenant les résultats des nb_tirages en fonction de la description des billes. exemple si billes = ["rouge","rouge","rouge","noir","noir"] pour 4 tirages le fonction pourra retourner ["rouge","rouge","noir","noir"]

lance_de_de

lance_de_de(nb_faces:int= 6)->int:

- nb_face: int nombre de face du dé. Par défaut sa valeur est 6.
- Retourne un entier [1;nb_faces]

FONCTIONS SUR LES CHAÎNES

len

len(objet: str | list) -> int

- *objet* peut être une chaîne de caractères ou une liste.
- Retourne la longueur de cette chaîne ou de cette liste

fich2chaine

fich2chaine(fichier='optionel')

- fichier est le nom complet (avec le chemin) d'un fichier contenant du texte brut.
- Si fichier n'est pas précisé, ouvre une boîte de dialogue pour
- Retourne chaîne formée du contenu du fichier 'fichier'

chaine2fich

chaine2fich(ch, fichier='optionnel')

- ch est une chaîne de caractères
- fichier est le nom complet (avec le chemin) d'un fichier contenant du texte brut.
- Si fichier n'est pas précisé, ouvre une boîte de dialogue pour
- Enregistre sous le nom 'fichier' la chaîne ch

aligne

aligne(chaine: str, taille, aligner="g")

- aligne une chaine sur un espace donné de caractères (taille)
- le paramètre aligner ("g,d, c, l, r, gauche, droite, centrer, left,right, center") permet un alignement de la chaine (alignement à gauche par défaut)

cadre

cadre(chaine: str, taille: int = 0, **kwargs)

- créer un cadre autour d'une chaine de caractère dans la console
- le paramètre optionnel aligner=("g,d, c, l, r, gauche, droite, centrer, left,right, center") permet l'alignement de la chaine (alignement à gauche par défaut) dans le cadre

affiche_tableau

affiche_tableau(tableau: list, taille: list = [], **kwargs)

- affiche *tableau* (list) a une ou deux dimension dans la console
- *taille* est une list qui permet de donner la taille de chaque colonne. Si taille n'est pas renseignée la largeur optimale est calculée.

paramètres optionnels:

- padx: espace ajouté avant et après la données
- mini: int : largeur mini d'une colonne
- maxi: int : largeur maxi d'une colonne
- sep_ligne : bool si True les lignes seront séparées
- aligner: chaine de caractère définissant l'alignement des colonnes "dcdg". La première colonne sera alignée à droite, la deuxième sera centrée, la troisième sera alignée à droite les dernières seront alignée à gauche
- entete: bool : si True la première ligne du tableau sera considérée comme une ligne d'entête
- pied: bool si True encadre la dernière ligne du tableau
- markdown : bool si True convertit le tableau au format markdown.

affiche_poly

affiche_poly(liste:list)

- liste est une list
- Affiche la liste sous forme d'un polynôme (liste[n] étant le coefficient de degré n). ex 1+2x-3x^2 ….. Si le paramètre optionnel format= "python" le polynome est retourné au format python 1 + 2*x - 3*x**2 ……
- le paramètre facultatif format=python permet d'écrire le polynôme au format python

FONCTIONS SUR LES LISTES

CSV2liste

CSV2liste(num, fichier='optionnel', sep=';', dec='.')

- num peut contenir un numéro de ligne ou un nom de colonne ('A' à 'Z') fichier est le nom complet (avec le chemin) d'un fichier contenant du texte brut.
- Si fichier n'est pas précisé, ouvre une boîte de dialogue pour le choisir
- Retourne une liste correspondant à la **ligne** ou la **colonne** du fichier 'fichier'

liste2CSV

liste2CSV(L, fichier='optionnel',affiche=False)
liste est une list
fichier est le nom complet (avec le chemin) d'un fichier contenant du texte brut.
Si fichier n'est pas précisé, ouvre une boîte de dialogue pour
Si paramètre optionnel affiche = True le contenu du fichier est affiché dans la console
Enregistre sous le nom 'fichier' la liste au format CSV

FONCTIONS SUR NUMPY

vecteur

vecteur(x, y, z='optionnel')->numpy.array():
- créer un vecteur au format numpy à partir des coordonnées x, y et z (optionnel)

norme

norme(v:numpy.array())->float:
- retourne la norme d'un vecteur au format numpy

abscisse

abscisse(v:numpy.array())->float:
- retourne l'abscisse du vecteur v

ordonnee

ordonnee(v:numpy.array())->float:
- retourne ordonnée du vecteur v

cote

cote(v:numpy.array())->float:

- retourne la cote du vecteur v (coordonnée en z du vecteur)

Fonction Graphique

figure

figure(nb_lignes=1, nb_cols=1, **kwargs)
Permet de préparer une ou plusieurs zones qui recevront le graphique
ex :**nb_lignes=3, nb_cols=2**
1 2
3 4
5 6

baton

baton(xi, ni='optionnel', **kwargs)
xi est une liste de valeurs
ni est la liste des effectifs associés, c'est un paramètre optionnel.
couleur donne la couleur du diagramme (optionnel)
Génère le diagramme en bâtons relatif à la liste.

secteur

secteur(valeurs: list, etiquettes: list, **kwargs):
Génère une diagramme secteur en fonction d'une liste de valeur

ligne_brisee

ligne_brisee(xi: list, yi: list, *args: list, **kwargs)
Génère un diagramme lignes brisées
xi liste d'abscisses
yi liste d'ordonnées (args place pour d'autres ordonnées pour avoir plusieurs courbes affichées)

histop

histop(Liste, Classes='optionnel', **kwargs)

Liste est une liste de valeurs
Si seulement Liste est renseigné, les valeurs seront réparties en 10 classes.
Si Classes est un entier, les valeurs seront réparties en ce nombre de classes.
Sinon, vous pouvez choisir vos classes d'amplitudes variées
en indiquant comme Classes la liste ordonnée des bornes.
Génère l'histogramme relatif à la Liste d'aire totale 1.

barre

barre(liste: list, a='optionnel', pas='optionnel', **kwargs):
liste est une liste de valeurs
Si seulement Liste est renseigné, les valeurs seront réparties en 10 classes.
Si Liste et a sont renseignés, les valeurs seront réparties en a classes.
Si les trois paramètres sont renseignés:
a est le centre de la première classe,
et pas est l'amplitude des classes.
Génère le diagramme en barres relatif à la Liste.

colonne

colonne(liste, a='optionnel', pas='optionnel', **kwargs)
idem barre

trace_courbe

trace_courbe(xi: list, yi: list, *args: list, **kwargs):
trace une ou des courbes défini par une liste d'abscisses et
une ou des listes d'ordonnées

nuage

nuage(xi: list, yi: list, *args: list, **kwargs):
trace un ou des nuages de point défini par une liste d'abscisses et
une ou des listes d'ordonnées

repere

repere(xmini, xmaxi, ymini, ymaxi, **kwargs):
défini un repère orthogonal en fonction de xmini, xmaxi, ymini, ymaxi
Est obligatoire pour utiliser **trace_fonction** ou les fonctions de géométrie

segment

segment(x1, y1, x2, y2, **kwargs):
dessine un segment dans la zone courante

point

point(x, y, **kwargs):
dessine un point dans la zone courante

rectangle

rectangle(x, y, largeur, hauteur, **kwargs):
dessine un rectangle dans la zone courante

carre

carre(x, y, largeur, **kwargs):
dessine un carré dans la zone courante

triangle

triangle(x1, y1, x2, y2, x3, y3, **kwargs):
dessine un triangle dans la zone courante

polygone

polygone(xyi: list[list], **kwargs):
dessine un polygone dans la zone courante

cercle

cercle(x, y, rayon, **kwargs):
dessine un cercle dans la zone courante

affiche_graphique

affiche_graphique()
affiche les graphiques précédemment calculés