

commentaire

# commen- taire	commentaire simple
""" docstring """	commentaire de type docstring, appellable avec help()

string method

capita- lize()	1 lettre en majuscule et reste en minuscule
find(m- odel)	renvoie le plus petit index où model est dans la string
isalnum()	True si string est alphanumérique
isdeci- mal()	True si string est un décimal
isdigit()	True si strind est en digit
islower()	True si string est en minuscule
isupper()	True si toute la string est en majuscule
'.'.join(- itérable)	crée une string à partir d'un itérable et le '.' est le séparateur entre chaque élément de la nouvelle string
lower()	retourne un copie de la string en minuscule
upper()	retourne une copie de la string en majuscule
replac- e(old, new, count=- 1)	retourne unen copie de la string avec les old remplacé par new, count correspond au nombre maximum de remplacement. -1 remplace toutes les occurences
split(sep- =None)	retourne une list des mots de la string, si sep = "", la liste contient tous les caractères

slicing

str[init:- end:step-]	init est le début de la liste, par défaut 0, end est la fin voulue et step est le pas, par default 1.
str[::-1]	renvoie la string à l'envert

f-string

f"{var}"	retourne la valeur de la variable accompagnée du text text"
{:<6}	aligné à gauche, droite ou centré sur 6 caractères, possib-
{:>6}	ilité d'ajouter un caractère pour remplir les emplacements
{:^6}	vides

changement de base

bin()	0b...
oct()	0o...
hex()	0x...
int()	...

vrac & tips

while (var := input()) > ...	permet d'affecter une valeur au sein d'une expression plus large. Le input ainsi que le while est un exemple
\$ pydoc3 -w /chemin_du_script	génère un fichier html avec la pydoc
subprocess.run(- ["commande_ba- sh", argument])	avec import subprocess pour executer une commande bash

fonction

def def (param1, param2, ...): commande commande return val	definition globale d'une fonction
def f(a, b, /, **kwargs):	avant le / le paramètre est exclusivement positi- onnel
var = lambda traitement	la lambda est une fonction anonyme qui ne retour- pas de valeur car le traitement est envoyé dans un variable

regex

import re	module re
re.search(model, string)	renvoie les occurences du model dans la string
re.match(model, str)	cherche si le début de la chaîne correspond
re.fullmatch(model, str)	si l'entièreté de la chaîne correspond



By **B4LBU**
cheatography.com/b4lbu/

Published 14th April, 2022.
Last updated 14th April, 2022.
Page 1 of 4.

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish
Yours!
<https://apollopad.com>

regex (cont)

<code>re.findall(model, str)</code>	ressence toutes les occurrences sous la forme d'une liste
<code>re.finditer(regex, str)</code>	produit le même résultat que <code>findall</code> mais sous forme d'un itérateur
<code>re.split(model, str)</code>	créer une liste à partir d'une chaîne de caractères en utilisant un modèle comme séparateur
<code>re.sub(model, elem, str)</code>	remplace le model de str par elem

sauvegarde de donnée

<code>import csv</code>	module csv qui permet de traiter facilement les csv
<code>writer = csv.writer(file)</code>	writer est un objet qui gère l'écriture
<code>writer.writerow(entete)</code>	entête est une liste
<code>writer.writerows(data)</code>	data est une liste de liste (autant de ligne que de liste dans data)
<code>reader = csv.reader(file)</code>	itérable permettant la lecture d'un fichier csv. <code>next(reader)</code> permet de sauter une ligne du reader

P.O.O

<code>class</code>	definit ma classe
<code>Maclasse:</code>	
<code>class B(A):</code>	la classe B hérite de A
<code>def __init__(self, parm ...):</code>	permet de créer une methode d'initialisation de la classe
<code>@classmethod</code>	methode disponible pour l'ensemble de la classe,
<code>def class(cls):</code>	cls représente la classe
<code>def instance_(self):</code>	methode applicable uniquement à l'instance, self représente l'instance
<code>if __name__ == '__main__':</code>	les commandes de se blocs sont jouées si le nom main est le nom actuel du programme
<code>super().__init__(param, ...)</code>	dans le cadre de l'héritage, <code>super()</code> représente le parent au sein de la classe enfant

P.O.O (cont)

<code>@property</code>	permet d'accéder à la valeur d'un attribut privé
<code>@<nom_du_getter>.setter</code>	modifier la valeur d'un attribut privé
<code>public : nom / protected : _nom / private : __nom</code>	encapsulation : restreindre l'accès des données entres elles

list method

<code>append()</code>	ajoute un élément à la fin de la liste
<code>clear()</code>	retire tous les éléments de la liste
<code>count(value)</code>	retourne le nombre d'occurrence de value
<code>extend(iterable)</code>	agrandit la liste en ajoutant les éléments d'un itérable
<code>insert(index, object)</code>	insert l'objet avant l'index voulu
<code>pop(index=-1)</code>	retire par default le dernier élément de la liste. <code>IndexError</code> si la liste est vide
<code>index(value)</code>	retourne le premier index de value. <code>ValueError</code> si value n'est pas dans la liste
<code>remove(value)</code>	retire la première occurrence de value. <code>ValueError</code> si elle n'est pas présente
<code>sort(reverse=False)</code>	retourn la liste triée (mais ne modifie pas la liste)

dict method

<code>clear()</code>	retire tous les éléments du dictionnaire
<code>get(key)</code>	retourne la valeur associé à key, sinon None
<code>items()</code>	objet donnant une view du dictionnaire
<code>keys()</code>	view de toutes les keys du dictionnaire
<code>values()</code>	objet donnant une view des valeur du dictionnaire

compréhensions

<code>[var for var in itérable]</code>	créer une nouvelle liste
<code>[int(i) for i in range(21) if i%2 == 0]</code>	exemple avec une liste des 20 premier nombre pair
<code>{k: v for k, v in zip(liste1, liste2)}</code>	dictionnaire (usage du zip)



structure de test

if condition :	structure classique
instruction	
elif condition :	
instruction	
else :	
instruction	
valeur = val_vrai if condition	structure ternaire, possibilité de
else val_faux	chaîner les ternaires

structure de boucle

while condition :	si la condition est remplie, la boucle s'arrête
instruction	
for var in sequence :	var prendra successivement toutes les valeurs des éléments de sequence. souvent utilisé avec range()
instruction	
for ... else:	la clause else est jouée si le block s'est exécuté sans
while ...	erreur
else:	

break & continue

break	fin de boucle
continue	fin d'itération (et passe à la suivante)

import

import librairie	importe l'ensemble de la librairie
from librairie import methode	importe uniquement la methode utile (à combiner avec le as)

gestion des erreurs

try:	le code qui suit le bloc try initie potentiellement une erreur
except nameerror:	intercepte l'erreur pour jouer le bloc qui suit
finally:	dans tous les cas joué (erreur ou pas erreur)
else:	joué si aucune erreur levée
raise nameerror('message')	permet de lever une erreur pour un traitement ulterieur

gestion des fichiers

from pathlib import Path	permet d'écrire Path.methode()
Path.cwd()	répertoire courant
Path.home()	renvoie la home
Path(r'/home/cptain/bin')	crée tout de même l'objet de type fichier même si le chemin n'existe pas
Path.home().joinpath('bin', 'bash/')	association de chemin

accès au fichier

with p.open() as f:	avec p un posixPath, cette manière de faire évite de devoir fermer le fichier après le traitement
f.read()	permet la lecture d'un fichier entier
f.readline()	permet de lire une ligne du fichier jusqu'au prochain \n
f.readlines()	créer une liste composée des lignes fi fichier
f.write()	écrire dans un fichier
f.writelines()	écrire à partir d'une liste
f.seek(nb)	positionne le curseur au début si nb=0, à la fin si nb=2
FileNotFoundError	Fichier introuvable (Erreur sur le nom du fichier ou fichier injoignable)
PermissionError	Erreur d'accès (Les droits ugo ou erreur de propriétaire)
IOError	Erreur d'ouverture

méthodes spéciales (les dunders)

__repr__(self)	affiche une information pour le développeur
__str__(self)	fournit un affichage pour l'utilisateur
__ge__(self, other)	>=
__le__(self, other)	<=
__lt__(self, other)	<
__eq__(self, other)	==



méthodes spéciales (les dunders) (cont)

<code>__ne__(self, other)</code>	<code>!=</code>
<code>__add__(self, other)</code>	<code>+</code>
<code>__mul__(self, other)</code>	<code>*</code>
<code>__truediv__(self, other)</code>	<code>/</code>
<code>__del__(self)</code>	détruire une objet <code>del()</code>
<code>__len__(self)</code>	Redéfinition de <code>len()</code>



By **B4LBU**
cheatography.com/b4lbu/

Published 14th April, 2022.
Last updated 14th April, 2022.
Page 4 of 4.

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish
Yours!
<https://apollopad.com>