wraps ne renvoie pas qq chose de décoré

parad 1

ligne 488 f = traceur(g)

f.__name__ va renvoyer wrapped on a perdu le nom de la fonction passée en param

@wraps(func) ne modifie pas func
mais dit que quand on demande son nom ça renvoie func et non wrapped

retour de f_memo renvoie | f_memo et aussi 1 dico
chaq f_memo a son propre dic

ici approche fonctionnelle avec les closure

- ĉ closure
- ĉ attribut de la f° renvoyée
- ĉ attribut de la f° de décoration

3 approches possibles proposées dans correction visualize

Un effet de bord = une modif faite par une fc° à l'ext des variables qu'elle manipule

exemple : variable globale, écrire sur l'écran
ttemutation ailleurs q ds les variables locales à la fonction

maybe : décorateur très utilisé
renvoie une valeur par défaut lorsq undefined

dans la fc° interne f et v sont pris dans la closure

↗ pr les ptes fc° ĉ 1 map

pb au débuggin carelles n'ont pas de nom

typiq prog fonctionnelle pas pour les élèves

pas recommandé au niveau génie logicielle car les gens ne comprennent pas

lambda x : expr
↑ return implicite

parag 2

for, while λ pas adapté

1936 Alonso Church
en même tps q machine de Turing

mfcb = lambda f,v : lambda value :
f(x) if value else v
à tester

λ calcul aussi puissant q machine de Turing mais oublié
≃60 retour du λ calcul pr les app° info q et le lien ac la logiq

if f(x) is None else f(x)
prenons un print dans f
on aura 2 affichages
pb: s'il y a un compteur c'est incrémenté 2 fois

lambda x: (lambda a: v if e is None else e)(f(x))

exple =
~~lambda f,v : lambda x : f(x) or v~~
ex = exple (positive_or_none, 42)
ex(0) affiche 42 marche tq f(x) renvoie Falsy

closure à l'intérieur de cette expression
e a la valeur de f(x)

counter_obj_inst

me doit remplacer q les cas indéfinis et pas 0

⚠ décorateur ne doit pas modifier comportemt

sol 1 compteur d'invocation global au décorateur
il n'est pas attribué du décoré
il est partagé par toutes les instances
on cumule le nb d'appel de ttes les instances

sol 3 : l'invocation est dans la closure on ne peut pas l'appeler
sol 2 : c'est 1 attribut de la fonction

ds notre exple le compteur revient à 0 avec l'appel de local ou global ça ne change rien

fournir le décorateur aux élèves pour la visualisation des appels

paradgm 3

python # objet, 1 fct est 1 obj, plutôt sol 2

---

* args

python pour gérer les arguments multiples / variables

f(1,2,3) 3 positions où on met des valeurs

si on veut la liste des arguments pr avoir leur nb par exple

max(1, 2, 3, ..., 99)

ce n'est pas une liste

* args transforme une séquence d'arguments / de paramètres / en ~~liste~~ / tuple ??

* args débale le tuple → tuple unpacking

** kwargs pour les arguments nommés avec valeurs par défaut

séquence d'args nommés transformé en dictionnaire

on peut renommer * séquence des args * patates

** " des mots clés ** carottes

---

python a des règles pour convertir / caster / les valeurs en booléan

false : entier 0 true : les autres

"falsy" chaîne vide

None

c'est pythonique return a is None

plutôt return a

map, filter et réduce renvoient des itérateurs paresseux
↳ très python

un itérateur est un objet qui renvoie la valeur courante et passe au suivant
renvoie une erreur particulière lorsqu'à la fin

design pattern itérateur fin des 90s.

for x in blabla fait des appels implicites à next tq l'exception n'est pas capturée et à ce moment-là on s'arrête

x = map(print, range(1, 10)) créé l'itérateur mais ne commence pas à itérer
rien ne s'affiche ds la console

list(map(print, range(1,10))) liste de 9 None

for i in x : pass
on fait 10 appels à next
les 9 None sont affichés maintenant

itérateurs paresseux qui ne feront le calcul q lors du parcours

tq pas dernier elt
yield (value) ← plus pythoniq
remplace DoublyLinkedListIterator

1m avec la mémoization fibo_rec 2x plus lente q fibo_iter
expl pathologique 2m additions
expl d'1ff qui se comporte mal
m additions linéaire taille des entrées

structure d'arbre : parcours très naturel en récursif
mémoization est une des techniqs pr rattraper fibo_rec

technique de l'accumulateur :

```
def fibo_rec(n, f=[0,1]):
    if n <= 1 : return f[n]
    return fib_iter(n-1, [f[1], f[0]+f[1]])
```

récursive : empiler / dépiler on x ces chgts de contexte et c'est pour ça q c'est plus long et cette pile à une taille max

?

on peut / doit déclarer l'attribut de wrapped dans sa fonction englobante juste avant return wrapped

assert : une fct booléenne qui doit être vérifiée

pytest sur un tableau donne la position de l'erreur

prog fonctionnelle : on ne peut pas attribuer une valeur à une case mémoire, on peut seulement produire des fonctions

" choisir paradigme selon besoins " c'est dans d'accès

faire du code pythonic : inaccessible si on mang de pratiq ne pas tomber ds apprendre python pour python

dans décorateur visualize

il faut encadrer l'appel de la fct entre visualize.level += 1 et visualize.level -= 1

on peut enlever l' * ?