Séance 3: TESTS DU PROGRAMMEUR

Université Paris-Diderot

Objectifs:

- Tester des fonctions selon un contrat
- Utiliser les conditionnelles et les boucles
- Manipuler le type `int` des entiers
- Manipuler le type `str` des chaînes de caractères

L'une des grandes problématiques de la programmation informatique est le "bug", c'est-à-dire, un programme qui ne termine pas ou qui ne renvoie pas le résultat attendu. Une méthode pour s'assurer que les programmes correspondent le plus possible à leur description est le *test* : on définit un "contrat" associant à un paramètre la valeur d'une fonction et on vérifie que la fonction satisfait ce contrat.

Après avoir testé plusieurs fonctions sur des exemples à l'aide de conditionnelles, nous allons utiliser des boucles pour automatiser les tests.

En bonus, il faudra donner la description de fonctions mystères après les avoir testées intensivement.

Rappel : Tous les fichiers que vous sauvegarderez ou créerez pour ce TP devront être mis dans le répertoire `TP3` sous-répertoire du répertoire `IP1-Python`. Mettez-y également **tous** les fichiers fournis pour ce TP.

Exercice 1 (Valeur Absolue, $\star$)
Le fichier `exoValeurAbsolue.py` *propose deux fonctions* `myAbs` *et* `newAbs` *qui prennent en paramètre un entier et retournent sa valeur absolue. Le but de cet exercice est de les tester. Les tests, écrits à la suite des fonctions, seront exécutés à l'aide de la commande* `python3 exoValeurAbsolue.py` *lancée depuis un terminal.*

1. *À l'aide de la procédure* `print`*, afficher dans le terminal le type de retour de la fonction* `myAbs` *évaluée sur l'entier* 0.
2. *Vérifier que les fonctions* `myAbs` *et* `newAbs` *vérifient le contrat suivant :*

   Contrat:
   À chaque paramètre `x` *est associé la valeur de retour de la fonction valeur absolue :*

$x=0$	$\rightarrow$	0	$x=2$	$\rightarrow$	2
$x=1$	$\rightarrow$	1	$x=10$	$\rightarrow$	10
$x=-1$	$\rightarrow$	1	$x=-100$	$\rightarrow$	100

   Par exemple, utiliser la suite d'instructions :

```
print("Entrée :")
print(0)
print("Sortie myAbs :")
print(myAbs(0))

```

   pour afficher le résultat du premier test de manière lisible :

```
Entrée :
0
Sortie myAbs :
0

```

3. *Pour les paramètres compris entre -3 et 2 tester la fonction* `myAbs` *selon le contrat suivant :*

   Contrat:

   Pour tout entier `i>=0` *passé en paramètre, la valeur de retour de la valeur absolue est* `i`.

   Pour tout entier `i<0` *passé en paramètre, la valeur de retour de la valeur absolue est* `-i`.

   Pour chaque valeur, utiliser une conditionnelle afin d'afficher la valeur du paramètre et `"ok"` *si le test est passé et* `"ko"` *sinon.*

4. *À l'aide d'une boucle, reprendre la question précédente pour vérifier le contrat pour les entiers compris entre -10 et 0 d'une part et pour les entiers compris entre 1 et 10 d'autre part.*

5. *À l'aide d'une boucle, afficher la valeur de retour de la fonction* `newAbs` *pour les entiers compris entre -100 et 99 inclus.*

6. *À l'aide d'une boucle et d'une conditionnelle, tester si la fonction* `newAbs` *vérifie le contrat suivant sur les paramètres compris entre -10 et 100, en affichant* `"ok"` *si le test est passé et un message d'erreur indiquant le paramètre qui ne passe pas sinon.*

   Contrat:

   Si la fonction prend en paramètre un entier, alors elle renvoie un entier positif.

7. *À l'aide d'une boucle et d'une conditionnelle, comparer les fonctions* `abs` *(valeur absolue de* Python*)* *et* `newAbs` *sur les entiers compris entre -200 et 199 inclus, en affichant* `"ok"` *si les deux fonctions retournent la même valeur et* `"ko"` *sinon. À partir de quelle valeur, les deux fonctions ne coïncident plus ?*

□

Exercice 2 (Miroir, $\star\star$)

Le fichier `exoMiroir.py` *contient deux fonctions* `reverse` *et* `myReverse` *permettant d'inverser une chaîne de caractères. Le but de cet exercice est de tester leur validité. Vous écrirez vos tests à la fin du fichier* `exoMiroir.py` *que vous compilerez à l'aide de la commande* `python3 exoMiroir.py`.

1. *À l'aide de la procédure* `print`*, afficher dans un terminal la valeur de retour des fonctions* `reverse` *et* `myReverse`*. Regarder si elles vérifient le contrat suivant :*

   Contrat:

   À chaque paramètre `s` *est associé la valeur de retour attendue d'une fonction encodant l'inversion d'une chaîne de caractères :*

s = "Miroir"	$\rightarrow$	*"rioriM"*
s = "a"	$\rightarrow$	*"a"*
s = ""	$\rightarrow$	*""*
s = "avec des espaces"	$\rightarrow$	*"secapse sed ceva"*
s = "elle"	$\rightarrow$	*"elle"*

2. *Vérifier le contrat suivant sur trois chaînes de caractères de votre choix.*

   Contrat:

   Inverser deux fois une chaîne de caractères revient à ne rien faire.

   ```
   reverse(reverse(s)) == s
   myReverse(myReverse(s)) == s
   ```

   Utiliser une conditionnelle afin d'afficher "ok" si le test est vérifié et sinon, d'afficher la chaîne de caractères `s` *passée en paramètre.*

3. *Pour tester les fonctions sur plus d'exemples, on va utiliser le dictionnaire de la langue française. La fonction* `wordFromDict` *prend en paramètre un entier i compris entre 0 et 336530 et renvoie la chaîne de caractères correspondant au $i^{ème}$ mot du dictionnaire. À l'aide d'une boucle sur une dizaine d'entiers*

compris entre 0 *et* 336531 *et d'une conditionnelle, afficher les valeurs de retour des fonctions* `reverse` *et* `myReverse` *sur quelques exemples.*

4. *Tester les fonctions* `reverse` *et* `myReverse` *sur des paramètres aléatoires en utilisant la fonction* `wordFromDict(i)` *et la fonction* `randrange(a, b)` *(du module* `random`*), qui renvoie aléatoirement un entier* `n` *compris entre les paramètres* `a` *et* `b` *:* `a <= n < b`.
5. *Plutôt que de vérifier à la main que la fonction renvoie le bon résultat, utiliser une boucle pour vérifier si les deux fonctions vérifient le contrat suivant sur un paramètre quelconque.*

   Contrat:
   Si sur le paramètre `s` *la fonction inverse retourne la chaîne* `r`*, alors les deux chaînes ont même longueur* `l` *et pour* `0 <= i < l, on a  s[i]` $==$ `r[l-i]`.
6. *À l'aide de la question précédente et de boucles, tester la fonction* `myReverse` *sur dix mots du dictionnaire choisis aléatoirement.*

□

Exercice 3 (Mystère, $\star\star\star$**)**
Le fichier `exoMyster.py` *contient* 7 *fonctions nommées* `mystere0`, `mystere1`,..., `mystere6` *qui prennent en paramètre soit un entier, soit une chaîne de caractères et renvoient, soit un entier, soit une chaîne de caractères. À l'aide de tests, déterminer leur type, leur comportement décrit sous forme d'un contrat le plus précis possible, puis vérifier votre hypothèse.*

□