

	Réc.	Réc.	Interf.			Créer	Dict.
	analyse	ecine	impiem.	API	implem.	module	
Act. 1	/5	/5					
Act. 2	/5	/5					
Act. 3					/5	/5	
Act. 4			/5	/5			
Act. 5	/5						
Act. 6						/5	
Act. 7	/5	/5					
Act. 8			/5				
Act. 9						/5	
Act. 10							/5
Total	/20	/15	/10	/5	/5	/15	/5

#### ACTIVITÉ 1

Une grenouille décide de monter un escalier. Quand elle saute, elle monte de 1 ou de 2 marches.

- 1. **Détermine** le **nombre de chemins différents possibles** si l'escalier est composé d'une seule marche.
- 2. Même question avec 2 marches, 3 marches et 4 marches.
- 3. **Proposer** une formulation récursive d'une fonction  $ch_p(m_r)$  calculant le nombre de chemins possibles en fonction du nombre de marches restantes  $(m_r)$ .
- 4. **Proposer** une implémentation en Python d'une telle fonction récursive.



#### ACTIVITÉ 2

John McCarthy a inventé la fonction  $f_{91}(n)$  définie par :

$$f_{91}(n) = \begin{cases} n - 10 & \text{si } n > 100, \\ f_{91}(f_{91}(n+11)) & \text{si } n \le 100. \end{cases}$$

- 1. **Calcule**  $f_{91}(101)$ ,  $f_{91}(100)$ ,  $f_{91}(98)$  et  $f_{91}(91)$ .
- 2. **Propose** une implémentation de cette fonction.

#### ACTIVITÉ 3

Voici l'interface minimale pour une structure de tableau redimensionnable tab\_redim:

fonction	description			
cree()	crée et renvoie un tableau vide (équivalent à [])			
lit(tr,i)	renvoie l'élément de tr à l'indice i (équivalent à tr[i])			
ecrit(tr,i,x)	place la valeur $x$ dans la case			
	d'indice i du tableau tr (équivalent à tr[i] = x)			
ajoute(tr,x)	ajoute le nouvel élément $x$ au tableau $tx$ , après ses éléments			
	actuels (équivalent à			
	tr.append(x))			



On décide de représenter un tableau redimensionnable tr de n éléments par un dictionnaire contenant (1) d'une part le nombre 'n' appelé taille et (2) d'autre part un tableau 't' de longueur supérieure ou égale à n appelée capacité.

Les n éléments sont stockés dans les cases d'indices 0 à n-1. Les autres cases de t contiennent None.

Propose une implémentation du module tab\_redim.

## ACTIVITÉ 4

Dans ton programme projet.py, tu souhaites utiliser les fonctions affiche et est\_entier de ton module personnel. Tout le code de ton module personnel est dans un fichier nommé perso.py qui est situé dans le même répertoire que ton programme.

- 1. **Indique** la/les instructions permettant de te donner accès aux deux fonctions.
- 2. Tu souhaites initialiser la variable tirage avec un nombre aléatoire pris entre 100 et 1000 inclus. **Indique** la/les instructions permettant de réaliser cela.
- 3. Tu souhaites notifier l'utilisation d'une erreur de nom de variable inexistante et affichant le texte "Désolé, la variable a été effacée". **Indique** la/les instructions permettant de réaliser cela.

## ACTIVITÉ 5

Indique les constituants d'une fonction récursive.



# ACTIVITÉ 6

En quoi consiste la factorisation de code? Donne en trois avantages.



**Donne** les définitions récursives et **propose** les implémentations en Python de la fonction mathématique somme(n) qui associe à n la somme des n premiers nombres entiers et de la fonction puissance(x,n) qui aux nombres entiers x et n associe le nombre  $x^n$ .

#### ACTIVITÉ 8

On dit que dans un module on distingue *implémentation* et *interface*. **Donne** une définition/explication de ces deux termes.

## ACTIVITÉ 9

**Donne** cinq exemples d'exceptions en Python.

**Indique** à quoi servent les mots clés try et except et **donne** un exemple d'utilisation et précise ce que fait ton code.

### ACTIVITÉ 10

**Indique** l'intérêt d'une table de hachage et **précise** en quelques lignes le fonctionnement général d'une table de hachage.