Exercice : La récursivité

Exercice 1

On donne la fonction suivante :

```python
# Fonction récursive ordre
def ordre(n):
    if n == 0:
        return '0'
    else:
        return ordre(n-1)+str(n)
```

1) Expliquer pourquoi la fonction **ordre** est récursive.

2) Que renvoie l'appel **ordre(4)** ?

3) Que se passe-t-il si on change la dernière instruction par **return str(n)+ordre(n-1)** ?

Exercice 2

On donne la fonction suivante :

```python
# Fonction récursive compte
def compte(n):
    if n == 1:
        return 1
    else:
        return 1 + compte(n-1)
```

1) Expliquer pourquoi la fonction **compte** est récursive.

2) Que renvoie l'appel **compte(5)** ?

3) Que se passe-t-il si on change la dernière instruction par **return 1+compte(n-2)** ?

Exercice 3

On donne le script suivant et la table de caractères ASCII :

```python
# Version itérative
mot=''
for i in range(65,91):
    mot += chr(i)
print(mot)
```

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
000	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
001	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
002	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
003	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
004	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
005	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
006	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
007	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

On rappelle que la fonction Python **chr** prend en argument un nombre entier N en écriture décimale et renvoie le caractère ASCII associé.

1) Quelle est la valeur renvoyée par la commande **chr(65)** ? Justifier.

2) Quel est l'affichage à l'issu du script ?

3) Écrire la fonction récursive **alphabet_recursif** renvoyant le même résultat que le script ci-dessus.

Exercice 4

Lorsqu'on effectue une remise de 10% sur un prix, cela revient à multiplier ce prix par la valeur $1 - 10/100$. On veut calculer des baisses successives de 10% sur une valeur, le nombre de remises étant défini à l'avance.

1) Calculer trois remises successives de 10% sur un prix de 100 €.

2) Montrer en détaillant le calcul qu'un algorithme récursif peut s'appliquer.

3) Écrire un script itératif qui calcule n remises successives de 10% sur un prix. On utilisera les variables **prix** et **n**. La variable **prix** contiendra la valeur finale.

4) Vérifier votre script avec un prix de 100 pour $n = 3$ remises.

5) Écrire la fonction récursive **remise_successive** qui calcule n remise de 10% sur un prix défini à l'avance.

Exercice 5

En mathématiques, pour trouver le plus grand commun diviseur de 2 nombres entiers, on applique l'algorithme d'Euclide, donné ci-dessous en python :

```
1 def pgcd(a,b):
2     if a<b:
3         print(a,b)
4         a,b=b,a
5         print(a,b)
6     if b==0:
7         return a
8     else:
9         return pgcd(b,a%b)
```

1) S'agit-il d'une fonction récursive ? Pourquoi ?

2) **a)** Que calcule l'opération $a\%b$ dans la dernière ligne de la fonction ?

b) Quelle est la valeur de 12%7 ?

c) Que se passe-t-il si a est strictement inférieur à b ?

3) Quelle est la signification des instructions aux lignes 2 et 3 de la fonction ?

4) Donner les différentes phases d'exécution de l'appel pgcd(28, 42)

Exercice 6

1) Calculer : **a)** 1×2 **b)** $1 \times 2 \times 3$ **c)** $1 \times 2 \times 3 \times 4$ **d)** $1 \times 2 \times 3 \times 4 \times 5$

2) Les produits précédents s'appellent **factorielles** et se notent en mathématiques avec un point d'exclamation. Par exemple : factorielle(4) $= 1 \times 2 \times 3 \times 4 = 4!$

a) Quelle est la valeur de factorielle(1) ?

b) Quelle égalité peut-on écrire entre factorielle(4) et factorielle(5) ?

c) Pour tout nombre entier n, exprimer factorielle(n) en fonction de $n - 1$.

3) Écrire la fonction récursive **factorielle(n)** qui prend en paramètre un nombre entier n et renvoie la valeur de sa factorielle. On donne ci-dessous le squelette de la fonction.

```
def factorielle(n):
    if ...:
        return ...
    else:
        return ...
```

Exercice 7

1) Écrire une fonction récursive **puissance** qui prend en paramètres un flottant x non nul et un entier naturel n et qui renvoie x^n. On se base sur la définition mathématique : $x^0 = 1$ et $x^n = x \times x^{n-1}$.

2) On propose une seconde méthode de calcul de la puissance.
Pour cela, on note $x^0 = 1$ et on remarque que si $n = 2k$ (n pair), alors $x^n = (x^2)^k$ et si $n = 2k+1$ (n impair) alors $x^n = x^{2k+1} = x(x^2)^k$.
Écrire une fonction récursive utilisant cette méthode de calcul.

Exercice 8

1) Expliquer quel est le résultat renvoyé par le code suivant :

```python
def mystere(n):
    if n<2:
        return str(n)
    else:
        return mystere(n//2)+str(n%2)
```

2) Écrire une fonction binaire qui prend en paramètres un entier relatif r et un entier naturel n strictement positif, et qui renvoie la représentation en machine de r sur n bits. La méthode utilisée est celle du complément à 2.
Déterminer l'écriture binaire sur n bits d'un nombre négatif r revient à déterminer l'écriture binaire du nombre positif $r + 2^n$.
Exemple de l'écriture binaire du nombre -35 sur 7 bits : $-35 + 2^7 = 93 = 1011101_2$.

Exercice 9

1) Dans un idle (pyzo, thonny, python) saisir le programme ci-dessous et le tester :

```python
from turtle import *

couleurs=['blue','green','yellow','orange','red','purple']
bgcolor('black')

def dessin():
    for i in range(180):
        color(couleurs[i%6])
        forward(i)
        right(59)

dessin()
```

2) En donner une version récursive.

Exercice 10

La fonction **fibonacci(n)**, qui doit son nom au mathématicien Leonardo Fibonacci, est définie récursivement, pout tout entier n, de la manière suivante :

$$\text{fibonnacci}(n) = \begin{cases} 0 & \text{si } n = 0, \\ 1 & \text{si } n = 1, \\ \text{fibonnacci}(n-2) + \text{fibonnacci}(n-1) & \text{si } n > 1. \end{cases}$$

1) Calculer fibonacci(5).

2) Écrire en python cette fonction fibonacci.