Introduction à la Programmation en Python

1 Analyse et Simulation de programmes

Exercice 1. *Conversion en Secondes*

— Simulez pas à pas la fonction `conversion_secondes` avec comme entrées : 12 jours, 12 heures, 12 minutes et 12 secondes.
— Annotez chaque ligne avec une description de son effet.
— Modifiez la fonction pour ajouter une entrée donnant un nombre d'années à convertir.

Exercice 2. *Branchement conditionnel*

Simulez pas à pas la fonction `calculatrice` sur plusieurs entrées différentes, puis annotez chaque ligne avec une description de son effet.
Que se passe-t-il sur l'entrée $a = 12$, $b = 0$ et `operation` = "%" ?

Exercice 3. *Répétition*

Simulez pas à pas la fonction `somme` avec comme entrée 6, puis annotez chaque ligne avec une description de son effet.
Modifier la fonction pour qu'elle calcule la factorielle de l'entrée.

Exercice 4. *Puissance*

Testez la fonction `exp` avec pour entrée :
— `a = 12, b = 3`
— `a = 12, b = -3`
La fonction n'est pas correcte, elle n'est pas cohérente avec sa spécification. En simulant pas à pas, identifiez le problème, puis corrigez le.

Exercice 5. *Arithmétique* COURS

Voici deux algorithmes $\mathcal{E}$ et $\mathcal{P}$.

Algorithme $\mathcal{E}$
ENTRÉES: a et b deux entiers
1. **tant que** $a \neq b$ **faire**
2. **si** $a \geq b$ **alors**
3. $a \leftarrow a - b$
4. **sinon**
5. $b \leftarrow b - a$
6. **fin si**
7. **fin tant que**
8. **renvoyer** a

Algorithme $\mathcal{P}$
ENTRÉES: n un entier
1. $p \leftarrow$ **VRAI**
2. $d \leftarrow 2$
3. **tant que** $d^2 \leq n$ **faire**
4. **si** $n \mod d = 0$ **alors**
5. $p \leftarrow$ **FAUX**
6. **fin si**
7. $d \leftarrow d + 1$
8. **fin pour**
9. **renvoyer** p

— Essayer de deviner leurs spécifications de sortie.
— Implémentez les algorithmes en Python.
— Simulez les algorithmes pas à pas.
— Vérifiez et éventuellement corrigez la spécification de sortie.

2 Premiers programmes

Exercice 6. *Logarithme entier*

On rappelle que pour tout $n \geq 1$ et $b \geq 2$, le logarithme entier en base b de n est l'unique entier l tel que

$$b^l \leq n < b^{l+1}$$

Implémentez en Python la fonction ayant les spécifications suivantes :

Algorithme `logarithme_entier`
ENTRÉES: $n \geq 1$ et $b \geq 2$ deux entiers
SORTIES: un entier. Le logarithme entier en base b de n

Exercice 7. *Méthode de Halley pour le calcul de $\sqrt{2}$*

Implémentez en Python la fonction `halley` qui calcule le $n^{\text{ème}}$ terme de la suite récurrente suivante :

$$H_0 = 1\,,\ H_{n+1} = H_n \cdot \frac{(H_n^2 + 6)}{(3\,H_n^2 + 2)}$$

Exercice 8. *Suite de Fibonacci* COURS

Implémentez en Python la fonction `fibonacci` qui calcule le $n^{\text{ème}}$ terme de la suite récurrente suivante :

$$F_0 = 0\,,\ F_1 = 1\,,\ F_{n+2} = F_{n+1} + F_n$$

Exercice 9. *Nombre parfait*

Un nombre parfait est un nombre entier positif égal à la somme de ses diviseurs stricts (lui-même exclu).
Implémentez une fonction `parfait` qui prend en entrée un entier et renvoie vrai s'il est parfait.

Exercice 10. *Syracuse* COURS

Les mathématiques ne sont pas encore prêtes pour de tels problèmes – Paul Erdõs

Une suite de Syracuse est définie par la relation de récurrence suivante :

$$S_{n+1} = \begin{cases} \dfrac{S_n}{2} & \text{si } S_n \text{ est pair,} \\ 3S_n + 1 & \text{si } S_n \text{ est impair.} \end{cases}$$

La conjecture de Syracuse stipule que, quelque soit $S_0 \in \mathbb{N}^*$, la suite de Syracuse débutant par S_0 atteint 1. On notera qu'une fois la valeur 1 atteinte, la suite boucle sur les valeurs $1, 4, 2, 1, \ldots$.
Implémentez une fonction `syracuse` qui prend en entrée un entier strictement positif s_0 et qui renvoie le premier rang auquel la suite atteint la valeur 1.
Est-ce que la fonction `syracuse` termine sur toutes les entrées ?

3 Génération de chaînes de caractère

c.f. le mémo sur la manipulation de chaînes de caractères

Exercice 11. *Pyramide*

Implémentez une fonction `pyramide` qui prend en entrée un entier $n \geq 0$ et qui renvoie une pyramide d'allumettes à n étages en chaîne de caractères. *e.g.* pour $n = 4$:

```
   |
  |||
 |||||
|||||||
```

Exercice 12. *Table ASCII*

Implémentez une fonction `table_ascii` qui ne prend aucune entrée et renvoie la table ascii dans une chaîne de caractère sous forme de tableau à double entrée :

	0	1	2	3	4	5	6	7	8	9
0										
10										
20										
30				!	"	#	$	%	&	'
40	(	)	*	+	,	-	.	/	0	1
50	2	3	4	5	6	7	8	9	:	;
60	<	=	>	?	@	A	B	C	D	E
70	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y
90	Z	[	\	]	^	_	`	a	b	c
100	d	e	f	g	h	i	j	k	l	m
110	n	o	p	q	r	s	t	u	v	w
120	x	y	z	{			}	~		

Exercice 13. *Table de multiplication*

Implémentez une fonction `table_multiplication` qui prend un entier en entrée et renvoie la table de multiplication de 0 à n dans une chaîne de caractère sous forme de tableau à double entrée. *e.g* pour $n = 4$:

	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	6	8
3	0	3	6	9	12
4	0	4	8	12	16

4 Pour s'entraîner

Exercice 14. *Suites adjacentes*

Implémentez en Python la fonction `adjacentes` qui prend en argument un flottant ϵ et qui calcule le premier rang n termes des suites adjacentes suivantes, à partir duquel $|a_n - b_n| < \epsilon$:

$$b_{n+1} = \frac{a_n + b_n}{2},\ a_{n+1} = \frac{2}{b_{n+1}},\ a_0 = 1, b_0 = 2$$

Exercice 15. *Méthode de Héron*

La suite x^a définie si dessous converge vers $\sqrt{a}$.

$$\forall n \in \mathbb{N} \quad x^a_{n+1} = \frac{x^a_n + \frac{a}{x^a_n}}{2},\ x^a_0 = a$$

Implémentez une fonction `heron` qui prend en argument deux entiers a et n positifs et renvoie x^a_n. Trouvez le rang à partir duquel x^2_n approche $\sqrt{2}$ à 10^{-10} prêt (≈ 1.4142135623).

Exercice 16. *Nombres Chanceux d'Euler*

Un nombre chanceux d'Euler est un nombre $n > 1$ tel que pour tout $0 \leq i \leq n - 2$, $i^2 + i + n$ est premier.
Implémentez une fonction `chanceux_euler` qui prend en entrée un entier n et renvoie vrai si et seulement si n est chanceux.

5 Pour aller plus loin

Exercice 17. *Fonction 91 de McCarthy*

La fonction 91 de McCarthy est définie pour $n \in \mathbb{N}$ par la formule suivante :

$$f(n) = \begin{cases} n - 10 & \text{si } n > 100 \\ f(f(n + 11)) & \text{sinon.} \end{cases}$$

Implémentez une fonction `mccarthy` qui prend en entrée un nombre entier n positif et renvoie $f(n)$. Observez le retour de la fonction pour les valeurs plus petites ou égales à 101.

Syntaxe Python

Objets Python

Type	Syntaxe	Commentaire
Entier	`12`, `-12`	Pas de limite de taille
Flottants	`12.0`, `-12.3`, `12.`, `1.2827e5`, `1.456e-6`	Précision 16 chiffres
Booléens	`True`, `False`	
Chaînes de caractère	`"Nadine"`, `'Nadine'`, `"""Nadine"""`	

Conversions

```
# Numérique vers chaîne de caractère
str(12), str(12.12), str(1.2e1), str(-12)
# Chaîne vers numérique
int("12"), float("-12.12"), float("1.2e-2")
```

Opérations

Type	Syntaxe
Entier	`+`, `-`, `*`, `//`, `%`, `**`
Flottants	`+`, `-`, `*`, `/`, `**`
Booléens	`b1 and b2`, `or`, `not`
Chaînes de caractère	`s1 + s2`, `s1 * 12`

Comparaisons

Numérique

```
x < y, x > y, x <= y, x >= y
```

Égalité entre objets

```
x == y, x != y
```

Fonctions

Définition

```
def fonction(entree1 : type1, entree2 : type2, ...) -> type_sortie :
    ... #Instructions
    return sortie
```

Appel

```
fonction(12, "Nadine", 12.12)
```

Variables

Affectation

```
variable = 12
```

Lecture

```
variable * 21
```


Structures de contrôle

Conditionnelle Générale

```
if condition1 :
    ... # Block d'instruction si la condition 1 est vrai
elif condition2 :
    ... # Block d'instruction si la condition 1 est fausse
    ... # mais la condition 2 est vrai
... # Plusieur, un ou aucun elif
else :
    ... # Block d'instruction si toutes les conditions sont fausse
```

Conditionnelle sans elif

```
if condition :
    ... # Block d'instruction si la condition est vrai
else :
    ... # Block d'instruction si la condition est fausse
```

Conditionnelle if simple

```
if condition :
    ... # Block d'instruction si la condition est vrai
... # Suite du programme
```

Répétition

```
while condition :
    ... # Block d'instruction à répeter tant que la condition est vrai
... # Suite du programme
```

Chaînes de caractère

Table ASCII

	0	1	2	3	4	5	6	7	8	9
0										
10										
20										
30				!	"	#	$	%	&	'
40	(	)	*	+	,	-	.	/	0	1
50	2	3	4	5	6	7	8	9	:	;
60	<	=	>	?	@	A	B	C	D	E
70	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y
90	Z	[	\	]	^	_	`	a	b	c
100	d	e	f	g	h	i	j	k	l	m
110	n	o	p	q	r	s	t	u	v	w
120	x	y	z	{		}	~			

Codage de caractère

```
ord('A'), chr(75)
```

```
Sortie : 65 K
```

Programmes du TP

```python
def conversion_secondes(jours : int, heures : int, minutes : int, secondes : int) -> int :
    """
    Entrées : 4 nombres entiers.
        nombre de jours, d'heures, de minutes et de secondes
    Sortie : un entier.
        Le nombre de secondes équivalents à la durée donnée en paramètre
    """
    total = 0
    total = total + secondes
    total = total + minutes * 60
    total = total + heures * ( 60 * 60 )
    total = total + jours * ( 60 * 60 * 24 )
    return total

def calculatrice(a : int, b : int, operation : str) -> int:
    """
    Entrées :
    - ( a : entier ), (b : entier) ; deux entiers
    - ( operation : chaîne de caractère ) ;
        le symbole de l'opération arithmétique à exécuter
    Sortie : un entier.
        Le résultat de l'opération ayant comme opérandes a et  b
    """
    resultat = 0
    if operation == "+" :
        resultat = a + b
    elif operation == "-" :
        resultat = a - b
    elif operation == "*" or operation == "x" :
        resultat = a * b
    elif operation == "/" and b != 0:
        resultat = a // b
    elif operation == "%" :
        resultat = a % b
    elif operation == "**" :
        resultat = a ** b
    else :
        resultat = None
    return resultat
```



```python
def somme(n : int) -> int :
    """
    Entrées : n un nombre entier positif
    Sortie : s un entier. la somme des entiers de 0 à n
    """
    somme_temporaire = 0
    entier = 0
    while entier <= n :
        somme_temporaire = somme_temporaire + entier
        entier = entier + 1
    return somme_temporaire

def exp(a : int, b : int) -> int :
    """
    Entrées : a et b deux nombre entiers
    Sortie : un entier. a porté à la puissance b
    """
    resultat = 1.0
    compteur_operations = 0
    if b >= 0 :
        while compteur_operations < b :
            resultat = resultat * a
            compteur_operations = compteur_operations + 1
    else :
        while compteur_operations < b :
            resultat = resultat / a
            compteur_operations = compteur_operations + 1
    return resultat
```