Ada Lovelace (1815 – 1852)

Ada Lovelace, en travaillant sur la machine de Babbage, élabore le premier véritable programme informatique au monde.
↪ Dicomaths p. 351

Alan Turing (1912 – 1954)

Turing donne une définition précise du concept d'algorithme. Dans un article, il présente pour la première fois les termes *programmation* et *programme*.
↪ Dicomaths p. 353

En 1943, l'ENIAC est le premier ordinateur ne comportant plus de pièces mécaniques. Les inventions du transistor (1948), du circuit intégré (1958) et du microprocesseur (1971) entraînent une augmentation de la puissance et une diminution de la taille des ordinateurs. Au XXIe siècle, les supercalculateurs deviennent de plus en plus puissants. Une véritable compétition mondiale de performance se joue.

En 1re générale, je consoliderai mes connaissances sur les variables, les instructions conditionnelles, les boucles et les fonctions. Je découvrirai la notion de liste.

En 1re technologique, j'interpréterai et je traduirai des algorithmes, je modifierai un programme informatique, j'écrirai une fonction simple en langage PYTHON, je décomposerai un programme en fonctions et j'organiserai des feuilles de calcul. Je découvrirai également (sauf en STD2A) la notion de liste.

À quoi ça sert ?

Par exemple :

- En **cryptographie**, à décréter qu'un nombre est premier avec une certaine probabilité (tests de primalité).
- En **géographie**, à déterminer le plus court chemin entre deux lieux (algorithme de Dijkstra).
- En **SES**, à minimiser un coût via une fonction linéaire à plusieurs variables réelles soumises à des contraintes linéaires (algorithme du simplexe).
- En **sciences de l'ingénieur**, à programmer des machines pour des fonctionnalités précises : robotique, automatique.

1

Ada Lovelace (1815-1852), fille du poète britannique Lord Byron, est connue pour avoir conçu le tout premier programme informatique destiné à la machine analytique de Charles Babbage. Cette machine est l'ancêtre du tout premier ordinateur.

Algorithmique et programmation

Je dois être capable de…	Proposition de parcours	
Déterminer le type d'une variable.	1 p. 24	35 36 p. 30
Comprendre une suite d'instructions (affectation, calculs avec ou sans variable(s)).	2 p. 24	37 38 p. 30
Comprendre et écrire une instruction conditionnelle.	3 4 p. 25 TP1 p. 35	43 44 48 49 p. 31
Comprendre et écrire une boucle bornée.	5 6 p. 26	51 52 p. 31 55 56 p. 32
Comprendre et écrire une boucle non bornée.	7 8 p. 27 et 28	58 59 61 62 p. 32
Comprendre et écrire une fonction simple.	9 10 p. 29 TP5 p. 38	63 64 65 66 p. 32

Act 1 activités · 1 exercices résolus · 16 exercices corrigés · 14 exercices non corrigés · TP1 travaux pratiques

Pour prendre un bon départ

Algo & Prog

Exo Parcours différenciés
Lienmini.fr/maths2-01

1. Appliquer un algorithme « débranché »

1. Appliquer plusieurs fois l'algorithme suivant.

```
▸ Écrire 3 nombres sur 3 morceaux de papier et les poser
  de la gauche vers la droite.
▸ Comparer les nombres des deux papiers de gauche : si celui
  le plus à gauche est le plus petit, ne rien faire, sinon,
  les échanger.
▸ Comparer les nombres des deux papiers de droite : si celui
  le plus à gauche est le plus petit, ne rien faire, sinon,
  les échanger.
▸ Comparer les nombres des deux papiers de gauche : si celui
  le plus à gauche est le plus petit, ne rien faire, sinon,
  les échanger.
```

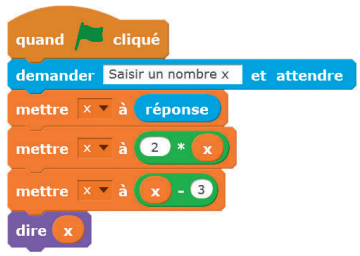
2. Que remarque-t-on à la fin de l'algorithme ?

2. Affecter une valeur à une variable

Soit le programme **SCRATCH** ci-contre.

1. Quelle valeur est affichée dans la bulle quand on saisit 5 lorsqu'un nombre est demandé ? Et pour – 5 ?

2. Recopier et compléter la phrase suivante : « Ce programme permet d'afficher l'… d'un nombre par la fonction $f : x \mapsto$ … »



3. Répéter une instruction un nombre fini de fois

Soit le programme **SCRATCH** ci-contre.

1. Tracer la figure réalisée par la partie du programme qui est à l'intérieur du bloc « répéter 3 fois » (On prendra 1 cm pour 10 pas). Préciser le point de départ et le point d'arrivée.

2. Tracer la figure réalisée par ce programme.

4. Traiter une instruction conditionnelle

Soit le programme **SCRATCH** ci-contre.

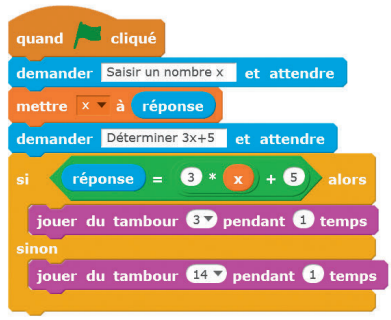
1. Que fait le programme si l'on répond respectivement –1 et 2 aux deux questions ?

2. Que fait le programme si l'on répond respectivement 3 et 4 aux deux questions ?

3. Décrire le fonctionnement de ce programme.

a) Que demande-t-il ? Que fait-il ?

b) Quelle est son utilité concrète ?



Doc Corrigés
Lienmini.fr/maths2-27

ZOOM SUR…

Algo & Prog
dans tout le chapitre

TICE
p. 14, 15, 16, 17, 33, 35, 36, 37, 38

Les autres disciplines
p. 36

Problème ouvert
p. 34

Activités

Algo & Prog TICE

Dans toutes les activités, on utilise le langage PYTHON, on ouvrira un nouvel onglet dans l'éditeur lorsqu'on demande d'écrire un nouveau programme.

1 Afficher et affecter des valeurs

20 min

1. Que veut dire le mot anglais `print` ?

2. a) Écrire et exécuter le **programme 1** et le **programme 2** ci-contre.

b) Expliquer l'affichage du **programme 1**.

c) Pourquoi n'a-t-on pas le même affichage pour le **programme 1** et pour le **programme 2** ? Que permettent les guillemets ?

d) Modifier la dernière ligne du **programme 2** en `print("b=",b)` puis expliquer ce que permet la virgule.

Programme 1
```
a = 2
b = 3*a+5
print(b)
```

Programme 2
```
a = 2
b = 3*a+5
print("b")
```

3. a) **Sans l'écrire**, dire ce que va afficher le **programme 3** ci-contre.

b) L'écrire, l'exécuter et vérifier l'affichage.

Programme 3
```
x = 5
y = 12
x = 3*x+2*y
y = 5*y-12*x
print(y)
```

↪ Cours 2 p. 18

2 Comprendre les variables de type numérique

30 min

1. Écrire et exécuter le **programme 4** et le **programme 5** ci-contre. Quelle différence d'affichage y a-t-il entre ces deux programmes ?

Remarque Dans le langage **PYTHON**, les variables ont des types :
- dans le **programme 4**, l'ordinateur considère que la variable **b** est de type **`int`** pour entier (*integer* en anglais) puisque tous les nombres considérés (**30** ; **2** et **60**) sont des entiers (écrits sans virgule).
- dans le **programme 5**, l'ordinateur considère que la variable **b** est de type **`float`** pour flottant, c'est-à-dire un réel dont il donne une *certaine* écriture décimale, car on a *forcé* **PYTHON** à ne pas considérer **2** comme un entier en écrivant **2.0** au lieu de **2**.

Programme 4
```
a = 30
b = a*2
print(b)
```

Programme 5
```
a = 30
b = a*2.0
print(b)
```

2. a) Écrire et exécuter le **programme 6** ci-contre.

b) De quel type est la variable **a** après la ligne `a=30` ?

c) De quel type est la variable **a** après la ligne `a=a*2` ?

d) De quel type est la variable **a** après la ligne `a=a/8` ?

Remarque Dans le langage **PYTHON**, une variable peut éventuellement changer de type suivant les calculs demandés.

Programme 6
```
a = 30
print(a)
a = a*2
print(a)
a = a/8
print(a)
```

3. On considère le **programme 7** ci-contre.

a) **Sans écrire ce programme**, dire de quels type sont les variables **a** et **b** lors de leur première affectation.

b) Écrire et exécuter ce programme.

c) En observant l'affichage du programme, expliquer ce que fait la ligne `b = float(b)` du programme.

Programme 7
```
a = 3
b = a*2
b = float(b)
print(a)
print(b)
```

4. On considère le **programme 8** ci-contre. Reprendre la question précédente afin d'expliquer ce que fait la ligne `b = int(b)` du **programme 8**.

Programme 8
```
a = 3
b = a*2.5
b = int(b)
print(a)
print(b)
```

↪ Cours 1 p. 18

20 min

3 Comprendre les variables de type textuel

1. a) **Sans l'écrire**, donner l'affichage du **programme 9** ci-contre.

b) Écrire le programme et vérifier la réponse à la question précédente.

2. Modifier le **programme 9**, en remplaçant la première ligne par `a = "bonjour"`.

Remarque Dans le langage **PYTHON**, lorsque l'on écrit une « valeur » entre guillemets " (ou apostrophes ') lors d'une affectation, alors cette variable est de type `str` pour chaîne de caractères (*string* en anglais) c'est-à-dire que sa « valeur » est le texte entre guillemets (même si ce texte est un nombre, comme dans la première version du **programme 9**).

3. a) Écrire et exécuter les **programmes 10, 11 et 12** suivants.

b) Expliquer chacun des affichages.

Programme 9
```
a = "30"
print(a)
a = a*2
print(a)
```

Programme 10
```
a = "3"
a = int(a)
b = a*2
print(b)
```

Programme 11
```
a = 3
a = str(a)
b = a*2
print(b)
```

Programme 12
```
a = "bonjour"
a = int(a)
b = a*2
print(b)
```

↪ Cours 1 p. 18

25 min

4 Programmer des instructions conditionnelles

1. Écrire **le programme 13**.
Attention à bien respecter l'**indentation** (c'est-à-dire l'espace en début de ligne) avant les `print` : pour la réaliser, on utilise les touches TAB ou ⇆ du clavier.

2. Exécuter deux fois le **programme 13** en saisissant d'abord `-3.4` puis `10.8` comme valeur de `x` demandée par le programme.
Expliquer les deux affichages obtenus.

Programme 13
```
x = float(input("Saisir un nombre : "))
if x >= 0:
  print("Le nombre est positif.")
else :
  print("Le nombre est strictement négatif.")
  print("Son opposé est : ")
  print(-x)
```

Coup de pouce Chercher ce que veulent dire `if` et `else`.

3. **Sans l'exécuter avec l'ordinateur**, dire ce que va afficher le programme si on l'exécute et que l'on saisit `-52.568` comme valeur de `x`.
Vérifier avec l'ordinateur.

4. a) Écrire le **programme 14** ci-contre en veillant à bien respecter l'indentation.

b) Le tester avec des valeurs positives et négatives.

c) Expliquer les différents affichages.

Programme 14
```
x = float(input("Saisir un nombre : "))
if x >= 0:
  print("Le nombre est positif.")
else :
  print("Le nombre est strictement négatif.")
  print("Son opposé est : ")
print(-x)
```

↪ Cours 3 p. 19

40 min

5 Programmer une boucle bornée

1. a) Écrire et exécuter le **programme 15** suivant.

```
Programme 15
for i in range(1,6) :
    print("Je répète 6 fois un message")
```

b) Quelle incohérence semble-t-il y avoir entre le **programme 15** et son affichage ?

2. Écrire et exécuter le **programme 16** ci-dessous.

```
Programme 16
for i in range(1,6) :
    print("J'affiche le message numéro",i)
```

3. L'un des deux algorithmes ci-dessous correspond au **programme 15**. Lequel ?

```
Algorithme 1
Pour i allant de 1 à 6
   Afficher "Je répète 6 fois un message"
Fin pour
```

```
Algorithme 2
Pour i allant de 1 à 5
   Afficher "Je répète 6 fois un message"
Fin pour
```

4. Soit `a` et `b` deux entiers, recopier et compléter les pointillés dans le tableau suivant.

En langage naturel	En PYTHON
`Pour i allant de … à …`	`for i in range(a,b)`

5. Dans le **programme 16**, remplacer tous les `i` par des `k` et exécuter de nouveau le programme. Que remarque-t-on ?

6. a) **Sans utiliser l'ordinateur**, dans le **programme 16**, quel serait l'affichage si l'on ajoutait une 3e ligne `print("au revoir")` indentée comme la 2e ligne ? non indentée ?

b) Vérifier avec l'ordinateur.

7. Écrire un programme qui affiche tous les carrés des nombres entiers de **0** à **1 000** (c'est-à-dire **0** ; **1** ; **4** ; **9** ; … ; **1 000 000**).

8. a) Écrire le **programme 17** suivant.

```
Programme 17
for i in range(5):
  print(i)
```

b) Soit `b` un entier. Recopier et compléter les pointillés dans le tableau.

En langage naturel	En PYTHON
`Pour i allant de … à …`	`for i in range(b):`

c) Reprendre la question **7.** avec l'instruction `range(b)` en remplaçant `b` par la valeur adéquate.

↳ Cours 4 p. 20

25 min

6 Programmer une boucle non bornée

Remarque Quelques informations utiles :
- Pour réaliser cette activité, les modules math et random de **PYTHON** sont nécessaires.
- la commande **PYTHON** `random.randint(a,b)` donne un nombre entier au hasard entre **a** inclus et **b** inclus.
- le mot anglais *while* se traduit par « tant que » en français.
- ≠ s'écrit `!=` en langage **PYTHON**.

1. Écrire le **programme 18** ci-contre et l'exécuter.

2. Recopier et compléter le tableau suivant donnant l'évolution des valeurs des différentes variables du programme dans le cas où `nombre_aleatoire` tiré au sort en début de programme est **6** et où l'on adopte la stratégie consistant à tester tous les entiers de **10** à **1** dans l'ordre décroissant jusqu'à trouver le bon.

Programme 18

```
1  import math
2  import random
3
4  nombre_aleatoire = random.randint(1,10)
5  print("Un entier vient d'être tiré au sort entre 1 et 10 inclus.")
6  reponse = 0
7  nombre_essais = 0
8  while reponse != nombre_aleatoire:
9    reponse = int(input("Devinez l'entier tiré au sort:"))
10   nombre_essais = nombre_essais+1
11
12 print("Vous avez trouvé. Nombre d'essais nécessaires:")
13 print(nombre_essais)
```

`nombre_aleatoire`	6	6	6	6	6	6
`reponse`	0	10	9			
`nombre_essais`	0	1	2			
Condition `reponse` ≠ `nombre_aleatoire`	Vérifiée	Vérifiée				

3. Expliquer pourquoi à la 6e ligne du **programme 18**, on a affecté la valeur 0 à la variable `reponse`. D'autres valeurs étaient-elles possibles ?

↪ Cours 5 p. 20

25 min

7 Programmer une fonction

On considère le **programme 19** ci-contre (**ne pas l'écrire tout de suite**) dont on a numéroté les lignes pour plus de commodité.

1. a) Quelle est la première ligne du programme correspondant à un affichage ?

b) Écrire et exécuter le **programme 19** en saisissant la valeur **5** pour la variable `resistance` et la valeur **4** pour la variable `intensite`.
Le premier affichage du programme est-il celui anticipé à la question **1. a)** ?

Programme 19

```
1  def calcul_tension(R,I):
2    U = R*I
3    print("Un calcul de tension vient d'être effectué.")
4    return U
5
6  resistance = float(input("Saisir la résistance(en ohms):"))
7  intensite = float(input("Saisir l'intensité(en ampères):"))
8  tension = calcul_tension(resistance,intensite)
9
10 print("La tension(en volts)est:")
11 print(tension)
```

2. Étant donné les différents affichages observés lors de l'exécution du programme, répondre aux questions suivantes.

a) Quelle est la première ligne « traitée » à l'exécution du programme ?

b) Lorsqu'on écrit `def calcul_tension(R,I) :` on dit que l'on définit la fonction `calcul_tension`. Quelles lignes correspondent au bloc de la fonction `calcul_tension` ?

c) Que se passe-t-il à la 8e ligne ? On précisera notamment les valeurs prises par les variables `R` et `I` de la fonction `calcul_tension`.

↪ Cours 6 p. 21

Cours

1 Types de variables

Définition **Entier, flottant, chaîne de caractères**

Dans un algorithme ou un programme, les variables considérées ont des types qui définissent la nature des valeurs qu'elles peuvent prendre. Les trois types principaux considérés en classe de Seconde sont :
- **les entiers quand les valeurs possibles de la variable sont des nombres entiers.**
- **les flottants quand les valeurs possibles de la variable sont des nombres réels.**
- **les chaînes de caractères quand les valeurs possibles de la variable sont des mots.**

Exemple

On doit écrire un programme effectuant des statistiques sur des équipes sportives. Dans ce programme, il y aura en particulier trois variables : `nom` qui correspond au nom de l'équipe, `effectif` qui correspond à son effectif et `moyenne_age` qui correspond à sa moyenne d'âge. Les valeurs possibles prises par :
- `nom` sont des mots : elle est donc de type chaîne de caractères.
- `effectif` sont des nombres entiers : elle est donc de type entier.
- `moyenne_age` sont des nombres réels, non entier d'une manière générale : elle est donc de type flottant.

Remarques

Dans ce manuel, le langage informatique choisi est **PYTHON** dans lequel :
- le type chaîne de caractères se nomme `str` (pour *string*),
- le type entier se nomme `int` (pour *integer*),
- le type flottant se nomme `float` (pour *floating-point*).

Nous verrons, par la suite un autre type de variable : le type booléen.

↪ Exercice résolu 1 p. 24

2 Affectation

Définition **Affectation d'une valeur à une variable**

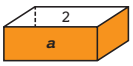
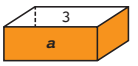
Lorsque l'on a affecté une valeur à une variable, on peut remplacer la variable par cette valeur dans les instructions qui suivent (par exemple dans les opérations arithmétiques).

Remarque On peut visualiser une boîte portant le nom de la variable dans laquelle on stocke une seule valeur.

Exemple

On considère la suite d'instructions ci-contre.
Les différentes étapes correspondantes sont les suivantes

En langage naturel	En PYTHON
a ← 2 a ← a+1	a=2 a=a+1

Étape 1	Étape 2	Étape 3
	a + 1 = 2 + 1 = 3	
Ligne a ← 2 « a reçoit la valeur 2 » ou « 2 est affecté à a » donc on place la valeur 2 dans la boîte a.	**Ligne** a ← a + 1 Le calcul a + 1 est effectué : son résultat est 3 puisque a vaut 2.	**Ligne** a ← a + 1 a reçoit la valeur 3 calculée précédemment donc on remplace 2 par 3 dans la boîte a.

Remarque Plutôt que d'affecter une valeur à une variable, on peut laisser l'utilisateur affecter la valeur de son choix à la variable.

L'algorithme suivant demande la saisie d'un nombre par l'utilisateur puis calcule et affiche le double de ce nombre.

En langage naturel	En PYTHON
a ← Valeur saisie Afficher 2×a	`a = float(input("Saisir une valeur"))` `print(2*a)`

↪ Exercice résolu 2 p. 24

3 Instructions conditionnelles

Définition **Instructions conditionnelles**

Dans un algorithme, on est parfois amené à exécuter une ou plusieurs instructions uniquement si une certaine condition est vérifiée, c'est ce que l'on appelle des instructions conditionnelles.
Si la condition n'est pas vérifiée, on peut soit exécuter un autre bloc d'instructions, soit ne rien faire.
Dans ces deux cas, on exécute ensuite la suite de l'algorithme.

Exemple

L'algorithme ci-contre permet de simuler un jeu dans lequel on lance un dé et où l'on gagne uniquement si le résultat est 6.

- Si la valeur de la variable `x` est 6 alors l'instruction `Afficher "Gagné !"` est exécutée. Sinon, c'est-à-dire si la valeur de la variable `x` n'est pas 6, alors l'instruction `Afficher "Perdu..."` est exécutée.
- Dans les deux cas, l'algorithme se poursuit après `Fin si` et la dernière instruction (qui n'est pas conditionnelle) `Afficher "À bientôt !"` est exécutée.

En langage naturel	En PYTHON

`x ← entier aléatoire entre 1 et 6`
`Si x = 6`
`  Afficher "Gagné !"`
`Sinon`
`  Afficher "Perdu..."`
`Fin si`
`Afficher "À bientôt !"`
	`import random`
	
	`x=random.randint(1,6)`
	`if x == 6:`
	`  print("Gagné!")`
	`else:`
	`  print("Perdu...")`
	`print("À bientôt!")`

Remarques

- La condition qui s'écrit `x = 6` en langage naturel s'écrit `x == 6` en **PYTHON**.

En **PYTHON**, le signe = seul est réservé à l'affectation.

- La condition `Sinon` est facultative.

Exemple

L'algorithme ci-contre est presque le même que le précédent mais n'affiche pas `"Perdu..."` si `x` est différent de 6.

En langage naturel	En PYTHON

`x ← entier aléatoire entre 1 et 6`
`Si x = 6`
`  Afficher "Gagné !"`
`Fin si`
`Afficher "À bientôt!"`
	`import random`
	
	`x=random.randint(1,6)`
	`if x == 6:`
	`  print("Gagné!")`
	`print("À bientôt!")`

Remarques

- La condition peut être de la forme « `Si condition 1 ou condition 2` » ou « `Si condition 1 et condition 2` » (↪ TP1 p. 35).
- Il existe un type de variables appelé **booléen** dont les valeurs ne peuvent être que « Vrai » ou « Faux ». Dans une instruction conditionnelle, le résultat d'une condition est un booléen.

Exemple

Dans l'algorithme en langage naturel précédent, si `x` vaut 5 alors la condition `x = 6` renvoie `FAUX` (`x == 6` renvoie `False` en **PYTHON**) et si `x` vaut 6 alors la condition `x = 6` renvoie `VRAI` (`x == 6` renvoie `True` en **PYTHON**).

↪ Exercices résolus 3 et 4 p. 25

4 Boucles bornées

Définition Boucle Pour

Lorsqu'on veut exécuter un nombre déterminé de fois un même bloc d'instructions, on utilise une boucle bornée, aussi appelée boucle Pour.
Ces boucles sont munies d'une variable compteur que l'on peut utiliser dans les instructions.

Exemple

L'algorithme ci-contre permet de calculer et afficher les 100 premiers nombres pairs strictement positifs puis, quand cela est fait, affiche `Terminé`.

Au premier passage dans la boucle, `i = 1` donc `a = 2`, puis `i = 2` donc `a = 4`,..., puis `i = 100` donc `a = 200`.
Quand ces instructions ont été exécutées, la boucle `Pour` est terminée, on exécute alors la suite de l'algorithme c'est-à-dire l'affichage de `Terminé`.

En langage naturel	En PYTHON

Pour i allant de 1 à 100
  a ← 2×i
  Afficher a
Fin pour
Afficher "Terminé"
	for i in range(1,101):
	  a=2*i
	  print(a)
	print("Terminé")

Remarque Dans la boucle précédente, la variable `i` est le compteur.
On dit qu'à chaque passage dans la boucle, `i` est incrémenté de `1` c'est-à-dire augmente de `1`.

Exercices résolus 5 et 6 p. 26

5 Boucles non bornées

Définition Boucle Tant que

Lorsqu'on veut répéter un même bloc d'instructions tant qu'une certaine condition est vérifiée, on utilise une boucle non bornée, aussi appelée boucle Tant que.

Exemple

L'algorithme ci-contre affiche la plus petite puissance de 2 supérieure strictement à 1 000 000.

En effet, le tableau suivant donne l'évolution des valeurs de la variable `p`.

En langage naturel	En PYTHON

p ← 1
Tant que p ⩽ 1 000 000
  p←p×2
Fin tant que
Afficher p
	p = 1
	while p<=1000000
	  p = p*2
	print(p)

	Initialisation						
p	1	1 x 2 = 2	2 x 2 = 4	4 x 2 = 8	...	524288	1048576
Condition p ⩽ 1 000 000	Vérifiée	Vérifiée	Vérifiée	Vérifiée	...	Vérifiée	Non vérifiée

Lorsque `p = 1048576` on sort de la boucle `Tant que` et on exécute la suite de l'algorithme c'est-à-dire l'affichage de la valeur de la variable `p`, soit : `1048576`.

Remarque Comme la condition de la boucle `Tant que` de l'exemple précédent porte sur la variable *p*, cette variable doit être initialisée préalablement (c'est-à-dire qu'il faut lui donner une valeur au départ) : ici, on l'a initialisée à $1 = 2^0$, la première puissance de 2.

Exercices résolus 7 p. 27 et 8 p. 28

6 Fonctions

Définition **Fonction**

Pour diverses raisons (de lisibilité ou pour éviter des répétitions d'instructions, par exemple), il peut être utile de définir une fonction c'est-à-dire un bloc d'instructions qui ne sera exécuté que s'il est appelé (éventuellement plusieurs fois).
Une fonction possède généralement des paramètres et retourne une valeur de retour (mais pas systématiquement, par exemple si elle réalise un affichage).

Exemple

L'indice de masse corporelle (IMC) d'une personne est donné par la formule $\text{IMC} = \frac{\text{masse}}{\text{taille}^2}$ où la masse est en kilogrammes et la taille en mètres.
On considère l'algorithme ci-dessous dont on a numéroté les lignes.

En langage naturel	En PYTHON
1 fonction calculIMC(masse, taille) 2 IMC←masse/taille² 3 Retourner IMC 4 5 IMCjean←calculIMC(60,1.6) 6 massepaul ← 85 7 taillepaul ← 1.80 8 Afficher calculIMC(massepaul, taillepaul)	1 def calculIMC(masse,taille): 2 IMC=masse/(taille*taille) 3 return IMC 4 5 IMCjean=calculIMC(60,1.6) 6 massepaul=85 7 taillepaul=1.8 8 print(calculIMC(massepaul,taillepaul))

Cet algorithme est constitué :
- d'une fonction appelée **calculIMC** (des lignes **1** à **3**) dont les paramètres sont les variables **masse** et **taille** (ligne **1**) et dont la valeur de retour est la valeur de **IMC** (ligne **3**) ;
- d'un algorithme principal à partir de la ligne **5**.

Dans la ligne **5**, **IMCjean ← calculIMC(60,1.6)** :
- l'algorithme principal appelle la fonction **calculIMC** en spécifiant que la variable masse (de la fonction) doit prendre la valeur **60** et que la variable **taille** (de la fonction) doit prendre la valeur **1.6**.
- le bloc d'instructions correspondant à la fonction **calculIMC** est alors exécuté : la variable **IMC** (de la fonction) reçoit la valeur **60 / 1,6² = 23,4375** (ligne **2**) et la retourne (ligne **3**) ;
- cette valeur **23,4375**, de la variable **IMC**, est retournée dans l'algorithme principal à l'endroit de l'appel de la fonction (ligne **5**), c'est-à-dire que l'algorithme principal affecte la valeur **23,4375** à la variable **IMCjean** (de l'algorithme principal).

De la même manière, à la ligne **8**, l'algorithme principal affiche la valeur retournée par la fonction c'est-à-dire **85 / 1,8²** soit approximativement **26,23**.

Remarques

- Dans le programme **PYTHON** précédent, **taille * taille** peut être remplacé par **taille**2**.
  D'une manière générale en **PYTHON**, **x**n** signifie x^n.
- Si une fonction **PYTHON** n'a pas de paramètre, on écrit tout de même les parenthèses vides dans sa définition et lors des rappels. Par exemple la fonction **de** (qui retourne un entier aléatoire entre 1 et 6) est appelée avec l'instruction **de( )**.

```
def de():
    return random.randint(1,6)
```

↪ Exercices résolus 9 et 10 p. 29

Pour programmer en PYTHON

Instruction et commande PYTHON	Exemple commenté

Affecter une valeur à une variable a de type `float` ou `int`
`a = valeur`
	Commentaire Attention, la deuxième ligne ne veut pas dire que `a` et `a + 1` sont égaux mais bien que `a` va prendre pour valeur la valeur actuelle de `a` augmentée de `1`.
	En langage naturel : `a ← 2` ; `a ← a+1`
	En PYTHON :
	`a = 2`
	`a = a+1`

Affecter une chaîne de caractères à une variable a de type `str` `a = "texte"` ou `a = 'texte'`	**Commentaire** Le mot `bonjour` est affecté à la variable `mot`. **Remarque** `mot = bonjour` renvoie un message d'erreur. En PYTHON : `mot = "bonjour"` ou `mot = 'bonjour'`

Modifier une variable a de type `int` en une variable de type `float` par un calcul où le résultat du calcul n'est pas entier
`a = calcul`
	Commentaire `a` est de type `int` à la première ligne mais est converti en type `float` à la deuxième ligne.
	En PYTHON :
	`a = 2`
	`a = a + 3.3`

Modifier le type d'une variable a en le type flottant (ou un autre type)
`a = float(a)`
	Commentaire `a` est de type `int` à la première ligne mais est converti en type `float` à la deuxième ligne.
	En PYTHON :
	`a = 2`
	`a = float(a)`

Affecter une valeur de type `str` saisie par l'utilisateur à une variable a après l'affichage d'un texte
`a = input("texte")`
	En langage naturel :
	`Afficher "Saisir un mot :"`
	`a ← Valeur saisie`
	`Afficher a`
	En PYTHON :
	`a = input("Saisir un mot :")`
	`print(a)`
	Commentaire Ce programme affiche `Saisir un mot:` attend la saisie d'une chaîne de caractères par l'utilisateur, qu'il affecte à `a`, puis affiche la chaîne de caractères saisie.

Affecter une valeur de type `float` ou `int` saisie par l'utilisateur à une variable a après l'affichage d'un texte
(on remplace `float` par `int` suivant le cas)
`a = input("texte")`
`a = float(a)`
ou
`a = float(input("texte"))`
	En langage naturel :
	`Afficher "a = ?"`
	`a ← Valeur saisie`
	`b ← 2*a`
	`Afficher b`
	En PYTHON :
	`a = input("a = ?")`
	`a = float(a)`
	`b = 2*a`
	`print(b)`
	Commentaire Ce programme affiche `a = ?` attend la saisie d'une valeur par l'utilisateur, qu'il affecte à `a`, puis convertit `a` en flottant puis calcule et affiche le double de `a`.

Afficher un texte
`print("texte")`
ou
`print('texte')`
	En langage naturel : `Afficher "bonjour"`
	En PYTHON : `print("bonjour")` ou `print('bonjour')`

Afficher la valeur d'une variable a
`print(a)`
	Commentaire Ce programme affiche 3.
	En PYTHON :
	`a = 3`
	`print(a)`

Afficher un texte et la valeur d'une variable *a*
`print("texte",a)`
	Commentaire Ce programme affiche `a = 3`.
	En PYTHON :
	`a = 3`
	`print("a=",a)`

Remarques

- Dans toute la suite, certains blocs d'instructions sont précédés d'une ligne finissant par un double point et sont **indentés**, c'est-à-dire que toutes les instructions sont précédées d'un espace (de même taille).
Dans ce cas, la fin de l'indentation marque la fin du bloc.
- Pour les tests de condition (après `if` ou `while`), les symboles =, ≠, ⩽ et ⩾ s'écrivent respectivement `==`, `!=`, `<=` et `>=` en PYTHON.

Instruction et commande PYTHON	Exemple commenté

Écrire des instructions conditionnelles
`if condition:`
`  instruction1`
`  instruction2`
`  etc`
`else:`
`  instruction3`
`  instruction4`
`  etcbis`
`Si condition`
`  instruction1`
`  instruction2`
`  etc`
`Sinon`
`  instruction3`
`  instruction4`
`  etcbis`
`Fin si`
	En PYTHON
	`x = float(input("x=? "))`
	`if x >= 0:`
	`  print("Le nombre est positif.")`
	`else:`
	`  print("x<0")`
	`  print("Son opposé est",-x)`
	`print("Bonne journée.")`
	Commentaire Si l'utilisateur choisit `5` pour valeur de `x`, le programme affichera `Le nombre est positif` puis `Bonne journée.` Si l'utilisateur choisit `– 3,2` pour valeur de `x`, le programme affichera `x < 0` puis `Son opposé est 3,2` puis `Bonne journée.`
	Remarque S'il n'y a pas de Sinon dans l'algorithme alors le bloc allant de `else :` à `etcbis` est retiré.

Écrire une boucle bornée
`for i in range(1,n+1):`
`  instruction1`
`  instruction2`
`  etc`
`Pour i allant de 1 à n`
`  instruction1`
`  instruction2`
`  etc`
`Fin pour`
	En langage naturel
	`Pour i allant de 1 à 100`
	`  a ← 2×i`
	`  Afficher a`
	`Fin pour`
	En PYTHON
	`for i in range(1,101):`
	`  a = 2*i`
	`  print(a)`
	Commentaire Ce programme calcule et affiche tous les nombres pairs entre `2 × 1 = 2` et `2 × 100 = 200` (et non `2 × 101 = 202`) car `range(a,b)` désigne les entiers entre `a` inclus et `b` exclu.

Écrire une boucle non bornée
`Tant que condition`
`  instruction1`
`  instruction2`
`  etc`
`Fin tant que`
`while condition :`
`  instruction1`
`  instruction2`
`  etc`
	En langage naturel
	`p ← 1`
	`Tant que p ⩽ 1 000 000`
	`  p ← 2 × p`
	`Fin tant que`
	`Afficher p`
	En PYTHON
	`p = 1`
	`while p <= 1000000:`
	`  p = 2*p`
	`print(p)`
	Commentaire Ce programme calcule les puissances de 2 qu'il affecte à la variable `p` (`1` ; `2` ; `4` ; `8` ; etc.) jusqu'à ce que `p > 1 000 000` puis affiche cette dernière valeur de `p`.

Écrire une fonction
`fonction nom(p1,p2,…)`
`  …`
`  Retourner a`
`programme principal`
`def nom(p1,p2,...):`
`  ...`
`  return a`
`programme principal`
	En langage naturel
	`fonction difference(x, y)`
	`  dif ← x-y`
	`  retourner dif`
	`Afficher difference(3,5)`
	En PYTHON
	`def difference(x,y):`
	`  dif = x-y`
	`  return dif`
	`print(difference(3,5))`
	Commentaire Ce programme appelle la fonction `difference` avec `3` et `5` pour valeurs de ses variables `x` et `y`. Celle-ci calcule donc `3 – 5 = – 2` (affectée à la variable `dif`) et retourne cette valeur qui est affichée puisque dans un `print`.