

Apprendre à programmer avec Kalam

Le cours complet, de A à Z

Le langage de programmation en arabe tchadien

« Al-barmaja be kalâmna »

Version du 24/11/2026 — tous les programmes de ce livre ont été exécutés automatiquement

Sommaire

Préface

Partie 1 — Premiers pas

1. C'est quoi Kalam ?
2. Installer Kalam
3. Le premier programme

Partie 2 — Les bases

4. Les variables
5. Les types de valeurs
6. Les opérations
7. Dialoguer avec l'utilisateur

Partie 3 — Décider et répéter

8. Les conditions
9. Les boucles

Partie 4 — Organiser les données

10. Les listes
11. Travailler avec les textes
12. Les dictionnaires (dikt)

Partie 5 — Organiser le code

13. Les fonctions (khidime)
14. Gérer les erreurs
15. Fichiers et modules
16. Le temps et le hasard

Partie 6 — Projets

17. Projet : la calculatrice
18. Projet : devine le nombre
19. Projet : la boutique du marché
20. Projet : le carnet de notes de la classe

Annexe A — Aide-mémoire des mots-clés

Annexe B — Toutes les fonctions intégrées

Annexe C — Les messages d'erreur courants

Annexe D — Solutions des exercices

Annexe E — Petit lexique

Préface

À qui s'adresse ce cours ?

Ce cours est pour **toute personne qui veut apprendre à programmer**, même sans jamais avoir écrit une ligne de code. Il suffit de savoir lire le français et de connaître un peu l'arabe tchadien.

Programmer, c'est donner des instructions précises à un ordinateur. D'habitude, on le fait avec des mots anglais : `if`, `while`, `function`… Avec **Kalam**, on utilise les mots de chez nous : `kan`, `tulma`, `khidime`. Les idées sont exactement les mêmes que dans les grands langages (Python, JavaScript…) : ce que vous apprenez ici vous servira partout.

Comment utiliser ce cours

- Lisez les chapitres **dans l'ordre** : chacun utilise ce que les précédents ont expliqué.
- **Tapez les exemples vous-même** au lieu de les copier. C'est en écrivant qu'on apprend.
- Chaque exemple est suivi de son **résultat** : vérifiez que vous obtenez la même chose.
- À la fin de chaque chapitre, faites les **exercices**. Les solutions sont à la fin du livre (Annexe D), mais cherchez d'abord seul !

Dans ce cours, le code Kalam ressemble à ceci :

```
ori("Salam aleykum!")
```

```
RÉSULTAT
Salam aleykum!
```

Et le cadre qui suit montre ce que l'ordinateur affiche quand on exécute le programme.

PARTIE 1

Premiers pas

1. C'est quoi Kalam ?

Un langage de programmation

Un **programme** est une liste d'instructions que l'ordinateur exécute **une par une, de haut en bas**. Un **langage de programmation** est la façon d'écrire ces instructions pour que l'ordinateur les comprenne.

L'ordinateur est très rapide mais il ne devine rien : il fait **exactement** ce qu'on lui dit. Si on se trompe d'un seul caractère, il le signale par un message d'erreur. Ce n'est pas grave : les erreurs font partie du travail de tous les programmeurs, même les plus forts.

Pourquoi en arabe tchadien ?

Kalam (« la parole ») a été créé pour que les Tchadiens puissent apprendre la programmation avec des mots familiers. Les mots-clés sont écrits **en lettres latines**, comme on écrit le tchadien dans les messages de téléphone :

Kalam	Sens en français	Rôle en programmation
`khalli`	laisse	créer une variable
`kan`	si	faire un choix
`kulla`	chaque	répéter pour chaque élément
`khidime`	travail	une fonction (un « travail » qu'on peut réutiliser)
`khalas`	c'est fini	fin d'un bloc

Les **accents sont facultatifs** : `khalâs` et `khalas`, `ôri` et `ori` sont pareils pour Kalam.

À retenir

- Un programme est une suite d'instructions exécutées de haut en bas.
- Kalam utilise des mots tchadiens, mais les idées sont celles de tous les langages.

2. Installer Kalam

Il y a trois façons d'utiliser Kalam. Choisissez celle qui vous convient.

Façon 1 : dans le navigateur (le plus simple)

Ouvrez le **terrain de jeu** sur le site de Kalam : **salah-barkai.github.io/kalam** (page « Essayer »). Vous écrivez votre programme à gauche, vous cliquez sur **Shaghghil** (Exécuter), et le résultat s'affiche à droite. Rien à installer ! C'est idéal pour les premiers chapitres.

Façon 2 : avec VS Code (recommandé ensuite)

VS Code est un éditeur de code gratuit utilisé par les professionnels.

1. Installez **VS Code** depuis *code.visualstudio.com*.
2. Dans VS Code, ouvrez l'onglet **Extensions** (`Ctrl+Shift+X`), cherchez **Kalam** et installez l'extension.
3. Créez un fichier qui finit par **`.klm`**, par exemple `salam.klm`.
4. Écrivez votre programme et appuyez sur **F5** pour l'exécuter.

C'est tout : Kalam est entièrement inclus dans l'extension. Le programme s'exécute dans un terminal en bas de la fenêtre ; quand il pose une question (`asal`), vous tapez la réponse dedans. `Ctrl+C` arrête un programme qui tourne trop longtemps.

L'extension colore aussi votre code, **souligne les erreurs en rouge** pendant que vous tapez, et affiche l'aide d'un mot quand vous passez la souris dessus.

Façon 3 : en ligne de commande avec kalam.exe

Pour utiliser Kalam dans un terminal, téléchargez **`kalam.exe`** sur la page des versions de Kalam (*github.com/salah-barkai/kalam/releases*). C'est un seul fichier, qui contient tout. Une fois, dans le dossier où il se trouve, tapez :

```
kalam.exe rakkib
```

(`rakkib` = « installe ».) Ensuite, dans n'importe quel nouveau terminal, il suffit d'écrire `kalam` suivi du nom du fichier :

```
kalam salam.klm
```

Sans nom de fichier, `kalam` ouvre le **mode interactif** : on tape une ligne, Kalam répond tout de suite. Pratique pour faire des petits calculs. On en sort en tapant `amrug`.

```
kalam
kalam> 2 + 3 * 4
14
kalam> amrug
Ma'a salâma!
```

3. Le premier programme

Afficher avec ori

L'instruction la plus importante pour commencer est `ori` (« montre », « informe »). Elle affiche ce qu'on lui donne entre parenthèses.

```
ori("Salam aleykum!")
ori("Ana gâ'id na'allim al-barmaja.")
```

```
RÉSULTAT
Salam aleykum!
Ana gâ'id na'allim al-barmaja.
```

Le texte est écrit **entre guillemets** `" "`. Sans guillemets, Kalam croirait que c'est un nom de variable (on verra ça au chapitre 4).

On peut donner **plusieurs choses** à `ori`, séparées par des virgules. Elles sont affichées avec un espace entre elles :

```
ori("Tchad indaha", 23, "wilâya")
ori(10 + 5)
ori()
ori("Al-akhir")
```

```
RÉSULTAT
Tchad indaha 23 wilâya
15

Al-akhir
```

Remarquez : `ori()` sans rien affiche une ligne vide. Et `10 + 5` sans guillemets est **calculé** : Kalam affiche `15`.

Les commentaires

Tout ce qui suit un `#` sur une ligne est un **commentaire** : Kalam l'ignore. Les commentaires servent à expliquer le code aux humains (et à vous-même dans trois mois !).

```kalam
# Programme : saluer la classe
ori("Salam ya awlâd!")   # ce texte s'affiche
# ori("ce texte ne s'affiche pas")
```

RÉSULTAT

```
Salam ya awlâd!
```

Votre première erreur

Que se passe-t-il si on oublie le guillemet de fin ?

```kalam
ori("Salam)
```

RÉSULTAT

```
Mushkila fi satir 1: al-kitaba di ma indaha guillemet fôg al-akhir (guillemet fermant manquant)
```

Kalam indique **le numéro de la ligne** (`satir`) et explique le problème en tchadien, avec la traduction en français entre parenthèses. Dans VS Code, l'erreur est soulignée en rouge avant même d'exécuter le programme.

> **Conseil :** quand une erreur apparaît, ne paniquez pas. Lisez le message, allez à la ligne indiquée, et regardez attentivement ce qui ne va pas.

Exercices

Exercice 3.1 — Écrivez un programme qui affiche votre nom, puis votre ville, sur deux lignes.

Exercice 3.2 — Faites afficher le résultat de `365 * 24` avec un texte qui explique ce que c'est (le nombre d'heures dans une année).

PARTIE 2

Les bases

4. Les variables

Ranger une valeur dans une boîte

Une **variable** est comme une boîte avec une étiquette : on y range une valeur pour la réutiliser plus tard. On crée une variable avec `khalli` (« laisse ») :

```
khalli usum = "Halime"
khalli umur = 16
ori(usum, "umurha", umur, "sana")
```

RÉSULTAT

```
Halime umurha 16 sana
```

Lisez `khalli umur = 16` comme : « laisse *umur* valoir 16 ». Le signe `=` ne veut pas dire « égal » comme en maths, mais **« ranger dans »**.

Changer la valeur

Une fois créée, une variable peut changer. On écrit alors simplement le nom, `=`, et la nouvelle valeur (sans `khalli`) :

```
khalli gurûsh = 1000
ori("Indi", gurûsh, "FCFA")
gurûsh = gurûsh - 250
ori("Ba'ad shiri ash-shay:", gurûsh, "FCFA")
```

RÉSULTAT

```
Indi 1000 FCFA
Ba'ad shiri ash-shay: 750 FCFA
```

La ligne `gurûsh = gurûsh - 250` se lit de droite à gauche : Kalam calcule d'abord `gurûsh - 250` (donc 750), puis range le résultat dans `gurûsh`.

Les raccourcis

Changer une variable à partir de sa propre valeur est si fréquent qu'il existe des raccourcis :

Raccourci	Équivalent
`x += 5`	`x = x + 5`
`x -= 5`	`x = x - 5`
`x *= 2`	`x = x * 2`
`x /= 2`	`x = x / 2`

```
khalli nugât = 10
nugât += 5
nugât *= 2
ori(nugât)
```

```
RÉSULTAT
30
```

Bien choisir les noms

- Un nom commence par une **lettre** ou `_`, puis lettres, chiffres ou `_` : `umur`, `taman_al_sukkar`, `talib2`.
- Pas d'espace dans un nom : écrivez `taman_al_sukkar`, pas `taman al sukkar`.
- Choisissez des noms **qui expliquent** : `taman` est mieux que `t`.
- Les accents et apostrophes sont ignorés dans les noms : `jadîd` et `jadid` sont **la même** variable.
- On ne peut pas utiliser un mot-clé comme nom (`kan`, `khalas`, `fi`, `min` …).

Une erreur fréquente

Utiliser une variable qui n'existe pas :

```
khalli taman = 500
ori(tamam)
```

```
RÉSULTAT
Mushkila fi satir 2: 'tamam' da ma ligênah — sawwi 'khalli tamam = ...' awwal
(variable non définie)
```

On a tapé `tamam` au lieu de `taman`. Kalam ne trouve pas cette variable (« *da ma ligênah* ») et propose de la créer avec `khalli`.

Exercices

Exercice 4.1 — Créez une variable `sana_al_wilâda` (année de naissance) et calculez votre âge en 2026.

Exercice 4.2 — Une variable `rasîd` vaut 5000. Retirez 1200 (achat de riz), puis ajoutez 3000 (salaire du jour), en utilisant `-=` et `+=`. Affichez le résultat final.

5. Les types de valeurs

Chaque valeur a un **type** (en Kalam : son `shakal`, sa « forme »). Le type dit ce qu'on peut faire avec la valeur.

Type	Exemples	Description
`nimra`	`12`, `-3`, `2.5`	un nombre
`kitaba`	`"Abéché"`, `'bamya'`	un texte (une écriture)
`sahi/ghalat`	`sahi`, `ghalat`	vrai ou faux
`mafi`	`mafi`	« il n'y a rien »
`liste`	`[1, 2, 3]`	une liste de valeurs (chapitre 10)
`dikt`	`{usum: "Ali"}`	un dictionnaire (chapitre 12)
`khidime`	`ori`	une fonction (chapitre 13)

La fonction `shakal` donne le type d'une valeur :

```
ori(shakal(42))
ori(shakal(3.14))
ori(shakal("salam"))
ori(shakal(sahi))
ori(shakal(mafi))
```

```
RÉSULTAT
nimra
nimra
kitaba
sahi/ghalat
mafi
```

Les nombres

Les nombres s'écrivent sans guillemets. Les nombres à virgule s'écrivent avec un **point** : `2.5` (pas `2,5`, car la virgule sépare les valeurs).

Les textes

Un texte (`kitaba`) s'écrit entre guillemets doubles `"…"` ou simples `'…'`. Quelques caractères spéciaux :

Écriture	Signification
`\n`	retour à la ligne
`\t`	tabulation
`\"`	un guillemet dans le texte

```
ori("Satir wâhid\nSatir tinên")
ori("Hu gâl: \"Âfe!\"")
```

RÉSULTAT

```
Satir wâhid
Satir tinên
Hu gâl: "Âfe!"
```

Vrai, faux et rien

`sahi` (vrai) et `ghalat` (faux) servent aux décisions (chapitre 8). `mafi` représente l'absence de valeur.

Convertir d'un type à l'autre

Fonction	Rôle	Exemple
`nimra(x)`	texte → nombre	`nimra("25")` donne `25`
`kitaba(x)`	n'importe quoi → texte	`kitaba(25)` donne `"25"`

```
khalli a = "12"
khalli b = "30"
ori(a + b)
ori(nimra(a) + nimra(b))
```

RÉSULTAT

```
1230
42
```

Avec des textes, `+` **colle** : `"12" + "30"` donne `"1230"`. Avec des nombres, `+` **additionne**. C'est pour ça qu'il faut convertir.

Si le texte ne ressemble pas à un nombre, `nimra` signale une erreur :

```
ori(nimra("khamsa"))
```

RÉSULTAT

```
Mushkila fi satir 1: 'khamsa' da ma nimra (ce n'est pas un nombre)
```

Exercices

Exercice 5.1 — Quel est le `shakal` de `"42"` ? Et de `42` ? Vérifiez avec un programme.

Exercice 5.2 — Les variables `x = "7"` et `y = "8"` sont des textes. Affichez leur somme comme un nombre (15).

6. Les opérations

Calculs

Opération	Signe	Exemple	Résultat
addition	`+`	`7 + 2`	`9`
soustraction	`-`	`7 - 2`	`5`
multiplication	`*`	`7 * 2`	`14`
division	`/`	`7 / 2`	`3.5`
reste de la division	`%`	`7 % 2`	`1`

```
ori(7 + 2, 7 - 2, 7 * 2, 7 / 2, 7 % 2)
ori(10 / 2)
```

RÉSULTAT

```
9 5 14 3.5 1
5
```

Le **reste** (`%`) est très utile : un nombre est **pair** si `n % 2` vaut `0`.

L'ordre des opérations

Comme en maths, `*`, `/` et `%` passent avant `+` et `-`. Utilisez des **parenthèses** pour changer l'ordre :

```
ori(2 + 3 * 4)
ori((2 + 3) * 4)
```

RÉSULTAT

```
14
20
```

Opérations sur les textes

```
khalli usum = "Abakar"
ori("Salam " + usum + "!")
ori("Umri " + 30)
ori("ha" * 3)
ori("=" * 20)
```

```
RÉSULTAT
Salam Abakar!
Umri 30
hahaha
====================
```

- `+` entre un texte et autre chose **colle** les deux (le nombre est converti en texte).
- `texte * n` **répète** le texte `n` fois. Pratique pour dessiner des lignes !

Diviser par zéro

```
khalli nâs = 0
ori(1000 / nâs)
```

```
RÉSULTAT
Mushkila fi satir 2: ma tagdar tigassim be sifir (division par zéro)
```

Diviser par zéro est impossible, en maths comme en programmation.

Exercices

Exercice 6.1 — Un taxi-moto coûte 500 FCFA par trajet. Combien dépense-t-on pour 2 trajets par jour pendant 22 jours ?

Exercice 6.2 — Quel est le reste de 100 divisé par 7 ? Écrivez le programme.

Exercice 6.3 — Affichez un cadre comme ceci, en utilisant `"*" * …` :

```
**********
* Kalam  *
**********
```

7. Dialoguer avec l'utilisateur

Poser une question avec asal

`asal` (« demande ») affiche une question et attend que l'utilisateur tape une réponse. La réponse est **toujours un texte**.

```
khalli usum = asal("Usumak shunu? ")
ori("Ahlan", usum + "!")
```

RÉSULTAT

```
Usumak shunu? Fatime
Ahlan Fatime!
```

Dans ce livre, les réponses tapées par l'utilisateur sont montrées dans le résultat, après la question.

Demander un nombre

Puisque `asal` renvoie un texte, il faut le convertir avec `nimra` pour faire des calculs :

```
khalli sana = nimra(asal("Inta wildôk be sana kam? "))
ori("Fi 2026 umrak", 2026 - sana, "sana.")
```

RÉSULTAT

```
Inta wildôk be sana kam? 2008
Fi 2026 umrak 18 sana.
```

On peut lire `nimra(asal(...))` de l'intérieur vers l'extérieur : d'abord on pose la question, puis on convertit la réponse en nombre.

Exercices

Exercice 7.1 — Demandez le prix d'un article et la quantité, puis affichez le total.

Exercice 7.2 — Demandez une température en degrés Celsius et affichez-la en Fahrenheit (`F = C * 9 / 5 + 32`).

PARTIE 3

Décider et répéter

8. Les conditions

Comparer

Une **comparaison** donne `sahi` (vrai) ou `ghalat` (faux) :

Signe	Sens	Exemple	Résultat
`==`	est égal à	`5 == 5`	`sahi`
`!=`	est différent de	`5 != 3`	`sahi`
`<`	plus petit que	`2 < 1`	`ghalat`
`>`	plus grand que	`2 > 1`	`sahi`
`<=`	plus petit ou égal	`3 <= 3`	`sahi`
`>=`	plus grand ou égal	`2 >= 5`	`ghalat`

```
ori(5 == 5, 5 != 3, 2 < 1, "bamya" == "bamya")
```

RÉSULTAT

```
sahi sahi ghalat sahi
```

> **Attention :** `=` range une valeur dans une variable, `==` compare deux valeurs. C'est une erreur très courante de les confondre.

kan ... khalas : si

Avec `kan` (« si »), le bloc de code n'est exécuté **que si** la condition est vraie. Le bloc se termine par `khalas`.

```
khalli umur = 20
kan umur >= 18
    ori("Inta kabîr, tagdar tisawwi carte d'identité.")
khalas
ori("Al-barnâmij kammal.")
```

RÉSULTAT

```
Inta kabîr, tagdar tisawwi carte d'identité.
Al-barnâmij kammal.
```

Les lignes à l'intérieur du bloc sont **décalées** (indentées) de 4 espaces. Kalam n'en a pas besoin pour comprendre, mais cela rend le code beaucoup plus facile à lire. VS Code le fait automatiquement.

walla : sinon

`walla` permet de dire quoi faire quand la condition est fausse :

```
khalli nimra_al_imtihân = 8
kan nimra_al_imtihân >= 10
    ori("Najaht! Mabrûk!")
walla
    ori("Ma najaht. Agri shuwayya wa jarrib marra tânye.")
khalas
```

RÉSULTAT

```
Ma najaht. Agri shuwayya wa jarrib marra tânye.
```

walla kan : plusieurs cas

Pour tester plusieurs cas l'un après l'autre, on utilise `walla kan` (« ou bien si »). Kalam prend **le premier** cas vrai et ignore les autres :

```
khalli harâra = 43
kan harâra >= 40
    ori("Al-hamu shadîd! Ashrab almi katîr.")
walla kan harâra >= 30
    ori("Al-hamu.")
walla kan harâra >= 20
    ori("Al-jaww sameh.")
walla
    ori("Al-barid! Albas kabbût.")
khalas
```

RÉSULTAT

```
Al-hamu shadîd! Ashrab almi katîr.
```

Combiner : wa, aw, ma

Mot	Sens	Vrai quand…
`wa`	et	les **deux** conditions sont vraies
`aw`	ou	**au moins une** condition est vraie
`ma`	non / pas	la condition est **fausse**

```
khalli umur = 25
khalli indah_permis = sahi
kan umur >= 18 wa indah_permis
    ori("Yagdar yasûg watîr.")
khalas

khalli yôm = "juma"
kan yôm == "juma" aw yôm == "sabit"
    ori("Al-yôm da ijâza.")
khalas

kan ma indah_permis
    ori("Ma yagdar yasûg.")
walla
    ori("Indah permis.")
khalas
```

RÉSULTAT

```
Yagdar yasûg watîr.
Al-yôm da ijâza.
Indah permis.
```

Ce qui compte comme vrai ou faux

Dans une condition, Kalam considère comme **faux** : `ghalat`, `mafi`, `0`, le texte vide `""`, la liste vide `[]` et le dikt vide `{}`. Tout le reste est **vrai**.

Exercices

Exercice 8.1 — Demandez un nombre et dites s'il est pair ou impair (pensez à `%`).

Exercice 8.2 — Demandez une note sur 20 et affichez la mention : moins de 10 « Ma najah », de 10 à 11 « Sameh », de 12 à 15 « Sameh shadîd », 16 et plus « Mumtâz ».

Exercice 8.3 — Une année est bissextile si elle est divisible par 4, sauf si elle est divisible par 100 et pas par 400. Écrivez le test pour 2024 et 1900.

9. Les boucles

Une **boucle** répète un bloc de code. C'est la grande force de l'ordinateur : il peut répéter mille fois sans se fatiguer.

kulla … fi min … lahaddi : compter

```
kulla i fi min 1 lahaddi 5
    ori("Dôra nimra", i)
khalas
```

RÉSULTAT

```
Dôra nimra 1
Dôra nimra 2
Dôra nimra 3
Dôra nimra 4
Dôra nimra 5
```

Lisez : « pour **chaque** *i* **dans** l'intervalle **de** 1 **jusqu'à** 5 ». La variable `i` prend successivement les valeurs 1, 2, 3, 4, 5 (la fin est **incluse**). On peut aussi compter à l'envers :

```
kulla i fi min 3 lahaddi 1
    ori(i, "...")
khalas
ori("Yalla!")
```

RÉSULTAT

```
3 ...
2 ...
1 ...
Yalla!
```

Exemple classique : la table de multiplication.

```
khalli n = 7
kulla i fi min 1 lahaddi 10
    ori(n, "x", i, "=", n * i)
khalas
```

```
RÉSULTAT
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
7 x 7 = 49
7 x 8 = 56
7 x 9 = 63
7 x 10 = 70
```

Additionner dans une boucle

Une technique très utilisée : une variable qui **accumule** un total.

```
khalli majmû = 0
kulla i fi min 1 lahaddi 100
    majmû += i
khalas
ori("1 + 2 + ... + 100 =", majmû)
```

```
RÉSULTAT
1 + 2 + ... + 100 = 5050
```

kulla … fi : parcourir

`kulla` peut aussi parcourir chaque élément d'une liste, ou chaque lettre d'un texte :

```
kulla madina fi ["Ndjamena", "Moundou", "Abéché", "Sarh"]
    ori("Salam ya", madina)
khalas

kulla harf fi "Tchad"
    ori(harf)
khalas
```

RÉSULTAT

```
Salam ya Ndjamena
Salam ya Moundou
Salam ya Abéché
Salam ya Sarh
T
c
h
a
d
```

tulma : tant que

`tulma` répète **tant que** la condition est vraie. On l'utilise quand on ne sait pas à l'avance combien de fois répéter.

```
khalli gurûsh = 10000
khalli shahar = 0
tulma gurûsh < 20000
    gurûsh = gurûsh * 1.1
    shahar += 1
khalas
ori("Ba'ad", shahar, "shahar, al-gurûsh bigi", garrib(gurûsh), "FCFA")
```

RÉSULTAT

```
Ba'ad 8 shahar, al-gurûsh bigi 21436 FCFA
```

> **Attention à la boucle infinie !** Si la condition ne devient jamais fausse, la boucle ne s'arrête jamais. Vérifiez toujours que quelque chose change dans la boucle. Pour arrêter un programme bloqué : `Ctrl+C` dans le terminal.

agif et fut : arrêter et passer

- `agif` (« arrête-toi ») sort immédiatement de la boucle.
- `fut` (« passe ») saute la fin du tour et passe au tour suivant.

```
kulla i fi min 1 lahaddi 10
    kan i % 2 == 0
        fut
    khalas
    kan i > 7
        agif
    khalas
    ori("fard:", i)
khalas
```

RÉSULTAT
```
fard: 1
fard: 3
fard: 5
fard: 7
```

Une boucle `tulma sahi` (« tant que vrai ») ne s'arrête que grâce à `agif`. C'est pratique pour redemander jusqu'à obtenir une bonne réponse :

```
tulma sahi
    jarrib
        khalli umur = nimra(asal("Umrak kam? "))
        kan umur > 0
            ori("Shukran!", umur, "sana.")
            agif
        khalas
        ori("Al-umur lâzim yabga fôg sifir.")
    mushkila
        ori("Aktib nimra, mâ kalâm.")
    khalas
khalas
```

RÉSULTAT
```
Umrak kam? abc
Aktib nimra, mâ kalâm.
Umrak kam? -5
Al-umur lâzim yabga fôg sifir.
Umrak kam? 17
Shukran! 17 sana.
```

(Les mots `jarrib` et `mushkila` sont expliqués au chapitre 14.)

Exercices

Exercice 9.1 — Affichez les nombres pairs de 2 à 20.

Exercice 9.2 — Calculez 1 × 2 × 3 × … × 10 (c'est « factorielle 10 »).

Exercice 9.3 — Demandez des nombres à l'utilisateur jusqu'à ce qu'il tape 0, puis affichez la somme.

Exercice 9.4 — Affichez un triangle d'étoiles de 5 lignes : `*`, `**`, `***`…

PARTIE 4

Organiser les données

10. Les listes

Créer une liste

Une **liste** range plusieurs valeurs dans l'ordre, entre crochets `[ ]`, séparées par des virgules :

```
khalli khudâr = ["bamya", "tamatim", "basal"]
khalli nugât = [12, 15, 9, 18]
khalli khalît = ["Ali", 20, sahi]
ori(khudâr)
ori(gadur(khudâr), "khudâr")
```

RÉSULTAT

```
['bamya', 'tamatim', 'basal']
3 khudâr
```

`gadur` (« la taille ») donne le nombre d'éléments.

Accéder à un élément

Chaque élément a une **position** (un *index*). Attention : on **commence à compter à 0** !

```
khalli khudâr = ["bamya", "tamatim", "basal"]
ori(khudâr[0])
ori(khudâr[2])
ori(khudâr[-1])
khudâr[1] = "fûl"
ori(khudâr)
```

RÉSULTAT

```
bamya
basal
basal
['bamya', 'fûl', 'basal']
```

L'index `-1` désigne le **dernier** élément, `-2` l'avant-dernier, etc.

```
khalli l = [10, 20, 30]
ori(l[3])
```

RÉSULTAT

```
Mushkila fi satir 2: index 3 marag barra (hors limites, gadur = 3)
```

Ajouter, retirer, chercher

Fonction	Rôle
`zid(liste, x)`	ajoute `x` à la fin (« ajoute »)
`shil(liste)`	retire et renvoie le dernier élément (« enlève »)
`shil(liste, i)`	retire l'élément à la position `i`
`x fi liste`	`sahi` si `x` est dans la liste
`fattish(liste, x)`	position de `x`, ou `-1` s'il n'y est pas (« cherche »)

```
khalli safar = ["Ali", "Halime"]
zid(safar, "Mahamat")
zid(safar, "Zara")
ori(safar)
khalli al_akhîr = shil(safar)
ori(al_akhîr, "ma maragat")
ori("Halime" fi safar, fattish(safar, "Mahamat"), fattish(safar, "Zara"))
```

RÉSULTAT

```
['Ali', 'Halime', 'Mahamat', 'Zara']
Zara ma maragat
sahi 2 -1
```

Parcourir une liste

```
khalli nugât = [12, 15, 9, 18, 14]
khalli majmû = 0
kulla n fi nugât
    majmû += n
khalas
ori("Al-mu'addal:", majmû / gadur(nugât))
```

RÉSULTAT

```
Al-mu'addal: 13.6
```

Kalam a aussi des fonctions toutes faites pour les listes de nombres :

```
khalli nugât = [12, 15, 9, 18, 14]
ori("Al-majmû':", jumla(nugât))
ori("Al-aktar:", aktar(nugât), "- al-agall:", agall(nugât))
ori("Murattabîn:", rattib(nugât))
ori("Magalûbîn:", gallib(nugât))
```

RÉSULTAT

```
Al-majmû': 68
Al-aktar: 18 - al-agall: 9
Murattabîn: [9, 12, 14, 15, 18]
Magalûbîn: [14, 18, 9, 15, 12]
```

Couper et joindre

```
khalli ayyâm = ["itnên", "talâta", "arba'a", "khamîs", "juma"]
ori(gata(ayyâm, 0, 3))
ori(gata(ayyâm, 3))
ori(lumm(ayyâm, ", "))
ori(ayyâm + ["sabit", "ahad"])
```

RÉSULTAT

```
['itnên', 'talâta', "arba'a"]
['khamîs', 'juma']
itnên, talâta, arba'a, khamîs, juma
['itnên', 'talâta', "arba'a", 'khamîs', 'juma', 'sabit', 'ahad']
```

`gata(liste, début, fin)` (« coupe ») prend les éléments de `début` jusqu'à `fin` **non incluse**.

Copier une liste

Attention : `khalli b = a` ne copie pas la liste, les deux noms désignent **la même** liste. Pour une vraie copie, utilisez `nuskha` :

```
khalli a = [1, 2, 3]
khalli b = a
khalli c = nuskha(a)
zid(a, 4)
ori(b)
ori(c)
```

RÉSULTAT

```
[1, 2, 3, 4]
[1, 2, 3]
```

Exercices

Exercice 10.1 — Créez la liste des notes `[11, 7, 15, 13, 9]`. Affichez le nombre de notes supérieures ou égales à 10.

Exercice 10.2 — Sans utiliser `aktar`, trouvez le plus grand nombre d'une liste avec une boucle.

Exercice 10.3 — Demandez 3 noms à l'utilisateur, rangez-les dans une liste, puis affichez-les dans l'ordre alphabétique.

11. Travailler avec les textes

Un texte se comporte un peu comme une liste de lettres : on peut connaître sa longueur, prendre une lettre, le parcourir avec `kulla`.

```
khalli b = "Ndjamena"
ori(gadur(b), b[0], b[-1], gata(b, 0, 3))
```

RÉSULTAT

```
8 N a Ndj
```

Les fonctions utiles

Fonction	Rôle
`kabir(t)` / `saghayyar(t)`	MAJUSCULES / minuscules
`naddif(t)`	enlève les espaces au début et à la fin (« nettoie »)
`ghayyir(t, ancien, nouveau)`	remplace (« change »)
`fattish(t, x)`	position de `x` dans `t`, ou `-1`
`x fi t`	`sahi` si `x` apparaît dans `t`
`bada_be(t, x)` / `kammal_be(t, x)`	commence par / finit par
`gassim(t, séparateur)`	découpe en liste (« partage »)
`lumm(liste, séparateur)`	joint une liste en texte (« rassemble »)
`gallib(t)`	inverse le texte

```
khalli jumla_hana = "  Al-yôm al-jaww sameh  "
ori("[" + naddif(jumla_hana) + "]")
ori(kabir("salam"), saghayyar("TCHAD"))
ori(ghayyir("Ana ba'arif", "ba'arif", "ma ba'arif"))
ori("jaww" fi jumla_hana, fattish("bamya", "m"))
ori(bada_be("0612345", "06"), kammal_be("barnâmij.klm", ".klm"))
```

RÉSULTAT

```
[Al-yôm al-jaww sameh]
SALAM tchad
Ana ma ba'arif
sahi 2
sahi sahi
```

Découper et recoller

```
khalli kalimât = gassim("ana wa inta wa hu")
ori(kalimât)
ori(gadur(kalimât), "kalimât")
ori(lumm(kalimât, "-"))

khalli tarîkh = gassim("23/11/2026", "/")
ori("Al-yôm:", tarîkh[0], "- as-sana:", tarîkh[2])
```

RÉSULTAT

```
['ana', 'wa', 'inta', 'wa', 'hu']
5 kalimât
ana-wa-inta-wa-hu
Al-yôm: 23 - as-sana: 2026
```

Exercice résolu : un palindrome

Un palindrome se lit pareil dans les deux sens (comme « kayak »).

```
khidime palindrome(kilma)
    khalli k = saghayyar(kilma)
    rajja k == gallib(k)
khalas

kulla k fi ["kayak", "Tchad", "Radar", "salam"]
    ori(k, "->", palindrome(k))
khalas
```

RÉSULTAT

```
kayak -> sahi
Tchad -> ghalat
Radar -> sahi
salam -> ghalat
```

Exercices

Exercice 11.1 — Demandez une phrase et affichez combien de mots elle contient.

Exercice 11.2 — Comptez le nombre de `a` dans `"Al-arabi at-tchadi"` (en minuscules).

Exercice 11.3 — Transformez `"mahamat ali"` en `"Mahamat Ali"` (première lettre de chaque mot en majuscule).

12. Les dictionnaires (dikt)

Des étiquettes au lieu des positions

Une liste range les valeurs **par position** (0, 1, 2…). Un **dikt** (dictionnaire) range les valeurs **par nom** (une *clé*). C'est parfait pour décrire une personne, un produit, une ville…

```
khalli talib = {usum: "Halime", umur: 15, fasil: "3ème"}
ori(talib)
ori(talib.usum)
ori(talib["umur"])
```

RÉSULTAT

```
{'usum': 'Halime', 'umur': 15, 'fasil': '3ème'}
Halime
15
```

On lit une valeur avec `.clé` ou `["clé"]`. Les deux sont équivalents. La forme `["clé"]` est nécessaire quand la clé est dans une variable.

Ajouter et modifier

```
khalli taman = {sukkar: 750, shay: 250}
taman.zêt = 1200
taman["sukkar"] = 800
ori(taman)
ori(taman.laban)
ori("shay" fi taman, "laban" fi taman)
```

RÉSULTAT

```
{'sukkar': 800, 'shay': 250, 'zêt': 1200}
mafi
sahi ghalat
```

Une clé qui n'existe pas donne `mafi`. Le test `"clé" fi dikt` dit si la clé existe.

Parcourir un dikt

`kulla` sur un dikt parcourt ses **clés** :

```
khalli taman = {sukkar: 750, shay: 250, zêt: 1200}
kulla haja fi taman
    ori(haja, ":", taman[haja], "FCFA")
khalas
ori("Al-mafâtîh:", mafatih(taman))
ori("Al-giyam:", giyam(taman), "- al-jumla:", jumla(giyam(taman)))
```

```
RÉSULTAT
sukkar : 750 FCFA
shay : 250 FCFA
zêt : 1200 FCFA
Al-mafâtîh: ['sukkar', 'shay', 'zêt']
Al-giyam: [750, 250, 1200] - al-jumla: 2200
```

Listes de dikt

On combine souvent les deux : une liste de dikt, comme un tableau.

```
khalli fasil = [
    {usum: "Ali", nugta: 14},
    {usum: "Zara", nugta: 17},
    {usum: "Moussa", nugta: 9},
]
kulla t fi fasil
    khalli natîja = "najah"
    kan t.nugta < 10
        natîja = "ma najah"
    khalas
    ori(t.usum, "-", t.nugta, "-", natîja)
khalas
```

```
RÉSULTAT
Ali - 14 - najah
Zara - 17 - najah
Moussa - 9 - ma najah
```

Exercices

Exercice 12.1 — Créez un dikt qui décrit votre téléphone (marque, prix, couleur), puis affichez chaque information sur une ligne.

Exercice 12.2 — Comptez combien de fois chaque mot apparaît dans `"shay sukkar shay laban shay sukkar"`. Résultat attendu : `{'shay': 3, 'sukkar': 2, 'laban': 1}`.

PARTIE 5

Organiser le code

13. Les fonctions (khidime)

Pourquoi des fonctions ?

Une **fonction** (`khidime`, « un travail ») est un morceau de code auquel on donne un nom, pour pouvoir le réutiliser. Vous en utilisez déjà : `ori`, `gadur`, `zid`… sont des fonctions. Maintenant, vous allez créer les vôtres.

```
khidime salam()
    ori("Salam aleykum!")
    ori("Kikkêf al-âfe?")
khalas

salam()
salam()
```

RÉSULTAT

```
Salam aleykum!
Kikkêf al-âfe?
Salam aleykum!
Kikkêf al-âfe?
```

La définition (`khidime salam() … khalas`) n'exécute rien : elle **apprend** un nouveau travail à Kalam. Le travail est fait chaque fois qu'on **appelle** la fonction : `salam()`.

Les paramètres

Une fonction peut recevoir des valeurs, appelées **paramètres**. Ils s'écrivent entre les parenthèses, séparés par des virgules :

```
khidime salam(usum)
    ori("Salam", usum + "!")
khalas

khidime ta'arif(usum, umur)
    ori(usum, "umurah", umur, "sana.")
khalas

salam("Ali")
salam("Zara")
ta'arif("Moussa", 12)
```

```
RÉSULTAT
Salam Ali!
Salam Zara!
Moussa umurah 12 sana.
```

À l'intérieur de la fonction, `usum` prend la valeur donnée à l'appel : `"Ali"` la première fois, `"Zara"` la deuxième.

rajja : renvoyer un résultat

Avec `rajja` (« rends »), une fonction **renvoie** une valeur qu'on peut utiliser ailleurs :

```
khidime murabba'(x)
    rajja x * x
khalas

khidime masâha_al_bêt(tûl, ard)
    rajja tûl * ard
khalas

ori(murabba'(7))
khalli m = masâha_al_bêt(12, 8)
ori("Al-masâha:", m, "mitir murabba'")
ori(murabba'(3) + murabba'(4))
```

```
RÉSULTAT
49
Al-masâha: 96 mitir murabba'
25
```

`rajja` termine immédiatement la fonction. Une fonction sans `rajja` renvoie `mafi`.

```
khidime tagdîr(nugta)
    kan nugta >= 16
        rajja "Mumtâz"
    walla kan nugta >= 10
        rajja "Sameh"
    khalas
    rajja "Ma najah"
khalas

kulla n fi [18, 12, 6]
    ori(n, "->", tagdîr(n))
khalas
```

```
RÉSULTAT
18 -> Mumtâz
12 -> Sameh
6 -> Ma najah
```

Variables locales

Une variable créée avec `khalli` **dans** une fonction n'existe que dans cette fonction. Elle est **locale** :

```
khidime hisâb()
    khalli natîja = 42
    rajja natîja
khalas

ori(hisâb())
ori(natîja)
```

```
RÉSULTAT
42
Mushkila fi satir 7: 'natîja' da ma ligênah — sawwi 'khalli natîja = ...' awwal
(variable non définie)
```

C'est une bonne chose : chaque fonction a ses propres « boîtes » et ne dérange pas les autres.

Une fonction qui s'appelle elle-même

Une fonction peut s'appeler elle-même : c'est la **récursion**. Il faut toujours un cas où elle s'arrête.

```
khidime factorielle(n)
    kan n <= 1
        rajja 1
    khalas
    rajja n * factorielle(n - 1)
khalas

ori(factorielle(5))
ori(factorielle(10))
```

```
RÉSULTAT
120
3628800
```

Les fonctions sont des valeurs

Une fonction peut être rangée dans une variable ou donnée à une autre fonction. On peut aussi créer une fonction **sans nom**, sur une seule ligne :

```
khalli marratên = khidime(x) rajja x * 2 khalas
ori(marratên(21))

khidime sawwi_le_kull(liste, f)
    khalli jadîd = []
    kulla x fi liste
        zid(jadîd, f(x))
    khalas
    rajja jadîd
khalas

ori(sawwi_le_kull([1, 2, 3, 4], marratên))
ori(sawwi_le_kull(["ali", "zara"], kabir))
```

```
RÉSULTAT
42
[2, 4, 6, 8]
['ALI', 'ZARA']
```

Exercices

Exercice 13.1 — Écrivez une fonction `zôj(n)` qui renvoie `sahi` si `n` est pair.

Exercice 13.2 — Écrivez une fonction `mu'addal(liste)` qui renvoie la moyenne d'une liste de nombres.

Exercice 13.3 — Écrivez une fonction `aktar_min_tinên(a, b)` qui renvoie le plus grand des deux nombres, sans utiliser `aktar`.

Exercice 13.4 — Écrivez une fonction récursive `fibonacci(n)` : `fibonacci(0) = 0`, `fibonacci(1) = 1`, et ensuite chaque nombre est la somme des deux précédents. Affichez les 10 premiers.

14. Gérer les erreurs

Lire un message d'erreur

Un message d'erreur Kalam a toujours la même forme :

```
Mushkila fi satir 12: <explication en tchadien> (<explication en français>)
```

1. **Où ?** : le numéro de la ligne (`satir`).
2. **Quoi ?** : l'explication. L'annexe C liste les messages les plus fréquents et comment les corriger.

Dans VS Code, la plupart des erreurs sont soulignées en rouge **avant** l'exécution.

jarrib … mushkila : prévoir le problème

Certaines erreurs ne dépendent pas de vous : l'utilisateur tape un mot au lieu d'un nombre, un fichier n'existe pas… Au lieu de laisser le programme s'arrêter, on peut **attraper** l'erreur :

```
jarrib
    ori("Gabul al-mushkila")
    ori(10 / 0)
    ori("Da ma yasal hini")
mushkila m
    ori("Ligêna mushkila:", m)
khalas
ori("Al-barnâmij mâshi ba'ad al-mushkila.")
```

RÉSULTAT

```
Gabul al-mushkila
Ligêna mushkila: ma tagdar tigassim be sifir (division par zéro)
Al-barnâmij mâshi ba'ad al-mushkila.
```

- Kalam **essaie** (`jarrib`) le bloc.
- Si une erreur arrive, il saute directement au bloc **`mushkila`** . La variable après `mushkila` (ici `m`) contient le message.
- S'il n'y a pas d'erreur, le bloc `mushkila` est ignoré.
- Dans les deux cas, le programme **continue** après `khalas` .

zagil : lancer sa propre erreur

`zagil` (« jette ») permet à vos fonctions de signaler un problème :

```
khidime sahab(rasîd, mablagh)
    kan mablagh > rasîd
        zagil "rasîdak ma bikaffi"
    khalas
    rajja rasîd - mablagh
khalas

khalli rasîd = 5000
kulla mablagh fi [2000, 4000, 1000]
    jarrib
        rasîd = sahab(rasîd, mablagh)
        ori("Sahabt", mablagh, "- fadal", rasîd)
    mushkila m
        ori("Ma gidirt nashab", mablagh, ":", m)
    khalas
khalas
```

```
RÉSULTAT
Sahabt 2000 - fadal 3000
Ma gidirt nashab 4000 : rasîdak ma bikaffi
Sahabt 1000 - fadal 2000
```

On peut jeter n'importe quelle valeur, par exemple un dikt avec plus d'informations :

```
jarrib
    zagil {kod: 404, sabab: "ma ligêna al-safha"}
mushkila e
    ori("Mushkila", e.kod, ":", e.sabab)
khalas
```

```
RÉSULTAT
Mushkila 404 : ma ligêna al-safha
```

Exercices

Exercice 14.1 — Écrivez une fonction `gassim_be(a, b)` qui jette une erreur claire si `b` vaut 0, puis appelez-la dans un `jarrib`.

Exercice 14.2 — Demandez un nombre à l'utilisateur ; s'il tape autre chose, affichez « Da ma nimra » au lieu de laisser le programme planter.

15. Fichiers et modules

Écrire et lire un fichier

Les variables disparaissent quand le programme s'arrête. Pour **garder** des informations, on les écrit dans un fichier :

Fonction	Rôle
`aktib(file, t)`	écrit le texte `t` dans le fichier (le remplace)
`aktib(file, t, sahi)`	**ajoute** `t` à la fin du fichier
`agri(file)`	lit tout le contenu du fichier

```
aktib("ma'lûmât.txt", "Usum: Halime\n")
aktib("ma'lûmât.txt", "Hille: Abéché\n", sahi)
khalli muhtawa = agri("ma'lûmât.txt")
ori(muhtawa)
kulla satir fi gassim(naddif(muhtawa), "\n")
    ori("->", satir)
khalas
```

```
RÉSULTAT
Usum: Halime
Hille: Abéché

-> Usum: Halime
-> Hille: Abéché
```

Le fichier est créé dans le même dossier que le programme `.klm`.

Importer un autre fichier avec jib

Quand un programme devient grand, on le **découpe** en plusieurs fichiers. `jib` (« apporte ») exécute un autre fichier `.klm` et rend ses fonctions et variables disponibles.

Fichier `adawat.klm` (« les outils ») :

Fichier `adawat.klm`

```
khidime khatt(n)
    rajja "-" * n
khalas

khidime riyal(mablagh)
    rajja kitaba(mablagh) + " FCFA"
khalas
```

Fichier principal, dans le même dossier :

```
jib "adawat"
ori(khatt(20))
ori("Al-jumla:", riyal(4500))
ori(khatt(20))
```

```
RÉSULTAT
--------------------
Al-jumla: 4500 FCFA
--------------------
```

L'extension `.klm` est facultative dans `jib`. Un fichier n'est importé qu'**une seule fois**, même si on écrit `jib` plusieurs fois.

Exercices

Exercice 15.1 — Écrivez un programme qui ajoute la date du jour (`al_yom()`) à la fin d'un fichier `journal.txt` à chaque exécution, puis affiche tout le fichier.

Exercice 15.2 — Créez un fichier `riyadiyât.klm` contenant les fonctions `murabba'` et `mu'addal`, et utilisez-les depuis un autre fichier.

16. Le temps et le hasard

Fonction	Rôle
`sudfa(a, b)`	un nombre entier au hasard entre `a` et `b` (inclus)
`azil(liste)`	un élément au hasard (« choisis »)
`khalbit(liste)`	mélange la liste
`al_yom()`	la date du jour, `JJ/MM/AAAA`
`as_sa()`	l'heure, `HH:MM`
`wakit()`	l'heure actuelle en secondes (pour mesurer une durée)
`agod(s)`	attend `s` secondes (« assieds-toi »)

```
ori("Al-zahr:", sudfa(1, 6))
ori("Al-yôm nâkulu:", azil(["êsh", "riz", "kisar", "fûl"]))
khalli awrâg = [1, 2, 3, 4, 5]
khalbit(awrâg)
ori(awrâg)
```

EXEMPLE DE RÉSULTAT

```
Al-zahr: 3
Al-yôm nâkulu: riz
[2, 5, 1, 3, 4]
```

```
ori("Al-yôm", al_yom(), "as-sâ'a", as_sa())
khalli bidâya = wakit()
agod(0.5)
ori("Antazart", garrib(wakit() - bidâya, 1), "sânye")
```

EXEMPLE DE RÉSULTAT

```
Al-yôm 24/11/2026 as-sâ'a 11:23
Antazart 0.5 sânye
```

Les résultats changent à chaque exécution : c'est le principe du hasard !

Exercice

Exercice 16.1 — Simulez 1000 lancers de dé et comptez combien de fois on obtient 6. Le résultat devrait être proche de 167.

PARTIE 6

Projets

Il est temps de tout combiner ! Chaque projet est un vrai programme complet. Tapez-le, exécutez-le, puis **modifiez-le** : c'est la meilleure façon de progresser.

17. Projet : la calculatrice

```
ori("=== Al-hâsiba ===")
ori("Aktib 'amrug' le tamrug.")

tulma sahi
    khalli a = asal("Al-nimra al-ûla: ")
    kan a == "amrug"
        agif
    khalas
    khalli amaliya = asal("Al-amaliya (+ - * /): ")
    khalli b = asal("Al-nimra at-tâniye: ")
    jarrib
        a = nimra(a)
        b = nimra(b)
        kan amaliya == "+"
            ori("=", a + b)
        walla kan amaliya == "-"
            ori("=", a - b)
        walla kan amaliya == "*"
            ori("=", a * b)
        walla kan amaliya == "/"
            ori("=", a / b)
        walla
            ori("Al-amaliya di ma na'arfaha.")
        khalas
    mushkila m
        ori("Mushkila:", m)
    khalas
khalas
ori("Ma'a salâma!")
```

RÉSULTAT

```
=== Al-hâsiba ===
Aktib 'amrug' le tamrug.
Al-nimra al-ûla: 12
Al-amaliya (+ - * /): *
Al-nimra at-tâniye: 4
= 48
Al-nimra al-ûla: 100
Al-amaliya (+ - * /): /
Al-nimra at-tâniye: 0
Mushkila: ma tagdar tigassim be sifir (division par zéro)
Al-nimra al-ûla: 5
Al-amaliya (+ - * /): +
Al-nimra at-tâniye: abc
Mushkila: 'abc' da ma nimra (ce n'est pas un nombre)
```

```
Al-nimra al-ûla: amrug
Ma'a salâma!
```

Pour aller plus loin : ajoutez l'opération `%`, et une opération `^` pour la puissance (avec une boucle).

18. Projet : devine le nombre

L'ordinateur choisit un nombre secret ; le joueur doit le deviner. Pour que le résultat soit le même dans ce livre, le nombre secret est fixé à 42. Dans votre version, remplacez `42` par `sudfa(1, 100)`.

```
khalli sirr = 42
khalli marrât = 0
ori("Ana fakkart fi nimra min 1 lahaddi 100. Khammin!")

tulma sahi
    khalli radd = nimra(asal("Nimritak: "))
    marrât += 1
    kan radd < sirr
        ori("Zîd! (plus grand)")
    walla kan radd > sirr
        ori("Nagguss! (plus petit)")
    walla
        ori("Mabrûk! Ligît an-nimra be", marrât, "marrât.")
        agif
    khalas
khalas
```

RÉSULTAT

```
Ana fakkart fi nimra min 1 lahaddi 100. Khammin!
Nimritak: 50
Nagguss! (plus petit)
Nimritak: 25
Zîd! (plus grand)
Nimritak: 37
Zîd! (plus grand)
Nimritak: 43
Nagguss! (plus petit)
Nimritak: 40
Zîd! (plus grand)
Nimritak: 42
Mabrûk! Ligît an-nimra be 6 marrât.
```

Pour aller plus loin : limitez le joueur à 7 essais ; gérez le cas où il tape autre chose qu'un nombre.

19. Projet : la boutique du marché

Un petit logiciel de gestion de stock pour une boutique : afficher les produits, vendre, refuser les ventes impossibles, et calculer la facture.

```
khalli dukkan = {
    sukkar: {taman: 750, kammiye: 20},
    shay: {taman: 250, kammiye: 50},
    dagig: {taman: 600, kammiye: 0},
    zêt: {taman: 1200, kammiye: 8},
}

khidime warri_al_bada'a()
    ori("=" * 34)
    kulla haja fi rattib(mafatih(dukkan))
        khalli b = dukkan[haja]
        kan b.kammiye == 0
            ori(" ", haja, ":", b.taman, "FCFA — kammal!")
        walla
            ori(" ", haja, ":", b.taman, "FCFA —", b.kammiye, "fi l-makhzan")
        khalas
    khalas
    ori("=" * 34)
khalas

khidime bi'(haja, kammiye)
    kan ma haja fi dukkan
        zagil "'" + haja + "' ma indina"
    khalas
    khalli b = dukkan[haja]
    kan kammiye > b.kammiye
        zagil "fi " + b.kammiye + " " + haja + " bas"
    khalas
    b.kammiye -= kammiye
    rajja b.taman * kammiye
khalas

warri_al_bada'a()

khalli talabât = [["sukkar", 2], ["shay", 4], ["dagig", 1], ["laban", 1], ["zêt", 3]]
khalli fâtûra = []
kulla talab fi talabât
    jarrib
        khalli taman = bi'(talab[0], talab[1])
        zid(fâtûra, taman)
        ori("OK  ", talab[1], talab[0], "=", taman, "FCFA")
    mushkila m
        ori("LA  ", talab[0], ":", m)
    khalas
```

```
khalas

khalli jumla_al_fatura = jumla(fâtûra)
ori("Al-jumla:", jumla_al_fatura, "FCFA")
kan jumla_al_fatura >= 5000
    ori("Takhfîd 10% ->", garrib(jumla_al_fatura * 0.9), "FCFA")
khalas
warri_al_bada'a()
```

RÉSULTAT

```
==================================
  dagig : 600 FCFA — kammal!
  shay : 250 FCFA — 50 fi l-makhzan
  sukkar : 750 FCFA — 20 fi l-makhzan
  zêt : 1200 FCFA — 8 fi l-makhzan
==================================
OK   2 sukkar = 1500 FCFA
OK   4 shay = 1000 FCFA
LA   dagig : fi 0 dagig bas
LA   laban : 'laban' ma indina
OK   3 zêt = 3600 FCFA
Al-jumla: 6100 FCFA
Takhfîd 10% -> 5490 FCFA
==================================
  dagig : 600 FCFA — kammal!
  shay : 250 FCFA — 46 fi l-makhzan
  sukkar : 750 FCFA — 18 fi l-makhzan
  zêt : 1200 FCFA — 5 fi l-makhzan
==================================
```

Pour aller plus loin : demandez les commandes à l'utilisateur avec `asal` ; enregistrez chaque vente dans un fichier `mabî'ât.txt` avec la date et l'heure.

20. Projet : le carnet de notes de la classe

Calculer la moyenne de chaque élève, le classement, et enregistrer le bulletin dans un fichier.

```
khalli fasil = [
    {usum: "Abakar", nugât: [12, 14, 9, 16]},
    {usum: "Halime", nugât: [17, 15, 18, 16]},
    {usum: "Moussa", nugât: [8, 11, 7, 10]},
    {usum: "Zara", nugât: [14, 13, 15, 12]},
]

khidime mu'addal(liste)
    rajja garrib(jumla(liste) / gadur(liste), 2)
khalas

khidime tagdîr(m)
    kan m >= 16
        rajja "Mumtâz"
    walla kan m >= 12
        rajja "Sameh shadîd"
    walla kan m >= 10
        rajja "Sameh"
    khalas
    rajja "Ma najah"
khalas

# Hisâb al-mu'addal le kulla talib
kulla t fi fasil
    t.mu'addal = mu'addal(t.nugât)
khalas

# Tartîb : min al-kabîr lê s-saghayyar (tri par sélection)
khalli murattab = nuskha(fasil)
kulla i fi min 0 lahaddi gadur(murattab) - 2
    kulla j fi min i + 1 lahaddi gadur(murattab) - 1
        kan murattab[j]["mu'addal"] > murattab[i]["mu'addal"]
            khalli wasît = murattab[i]
            murattab[i] = murattab[j]
            murattab[j] = wasît
        khalas
    khalas
khalas

khalli satir_hana = []
khalli martaba = 1
kulla t fi murattab
    khalli s = martaba + ". " + t.usum + " — " + t["mu'addal"] + " — " + tagdîr(t["mu'
    ori(s)
```

```
    zid(satir_hana, s)
    martaba += 1
khalas

aktib("bulletin.txt", lumm(satir_hana, "\n"))
ori("Al-bulletin maktûb fi bulletin.txt (" + gadur(satir_hana) + " talamîz)")
```

RÉSULTAT

```
1. Halime — 16.5 — Mumtâz
2. Zara — 13.5 — Sameh shadîd
3. Abakar — 12.75 — Sameh shadîd
4. Moussa — 9 — Ma najah
Al-bulletin maktûb fi bulletin.txt (4 talamîz)
```

Ce programme utilise presque tout le cours : dikt, listes, boucles imbriquées, fonctions, conditions, textes et fichiers. Bravo si vous le comprenez entièrement !

Pour aller plus loin : affichez la moyenne de la classe et le meilleur élève de chaque matière.

Annexe A — Aide-mémoire des mots-clés

Mot	Sens	Utilisation
`khalli`	laisse	`khalli x = 5`
`kan`	si	`kan x > 0 … khalas`
`walla kan`	ou bien si	`walla kan x == 0`
`walla`	sinon	`walla … khalas`
`khalas`	c'est fini	fin de tout bloc
`tulma`	tant que	`tulma x > 0 … khalas`
`kulla … fi`	pour chaque … dans	`kulla x fi liste … khalas`
`min … lahaddi`	de … jusqu'à	`kulla i fi min 1 lahaddi 10`
`agif`	arrête	sortir d'une boucle
`fut`	passe	tour suivant de la boucle
`khidime`	travail	`khidime f(x) … khalas`
`rajja`	rends	`rajja x * 2`
`jarrib`	essaie	`jarrib … mushkila m … khalas`
`mushkila`	problème	bloc exécuté en cas d'erreur
`zagil`	jette	`zagil "message"`
`jib`	apporte	`jib "adawat"`
`sahi` / `ghalat`	vrai / faux	valeurs booléennes
`mafi`	il n'y a rien	absence de valeur
`wa` / `aw` / `ma`	et / ou / non	`kan a wa ma b`
`fi`	dans	`kan x fi liste`

Opérateurs : `+ - * / %`, comparaisons `== != < > <= >=`, raccourcis `+= -= *= /=`.

Annexe B — Toutes les fonctions intégrées

Fonction	Rôle
`ori(a, b, …)`	afficher
`asal(question)`	demander une réponse (texte)
`nimra(x)`	convertir en nombre
`kitaba(x)`	convertir en texte
`shakal(x)`	type d'une valeur
`gadur(x)`	longueur d'un texte, d'une liste ou d'un dikt
`zid(liste, x)`	ajouter à la fin
`shil(liste)` / `shil(liste, i)`	retirer (le dernier / à la position i)
`rattib(liste)`	copie triée
`gallib(x)`	inverser une liste ou un texte
`nuskha(x)`	copier une liste ou un dikt
`azil(liste)`	élément au hasard
`khalbit(liste)`	mélanger
`mafatih(dikt)` / `giyam(dikt)`	clés / valeurs
`fattish(x, y)`	position de y dans x, ou -1
`gata(x, début, fin)`	morceau d'un texte ou d'une liste
`gassim(t, sép)` / `lumm(liste, sép)`	découper / joindre
`ghayyir(t, ancien, nouveau)`	remplacer
`naddif(t)`	enlever les espaces autour
`bada_be(t, x)` / `kammal_be(t, x)`	commence / finit par
`kabir(t)` / `saghayyar(t)`	majuscules / minuscules
`aktar(…)` / `agall(…)`	maximum / minimum
`jumla(liste)`	somme
`garrib(x, chiffres)`	arrondir
`mutlag(x)`	valeur absolue
`jidir(x)`	racine carrée
`PI`	3.14159…
`sudfa(a, b)`	entier au hasard entre a et b
`al_yom()` / `as_sa()`	date / heure

Fonction	Rôle
`wakit()` / `agod(s)`	secondes actuelles / attendre
`agri(file)`	lire un fichier
`aktib(file, t)` / `aktib(file, t, sahi)`	écrire / ajouter

Annexe C — Les messages d'erreur courants

Message	Cause probable	Solution
`'x' da ma ligênah`	variable mal écrite ou pas encore créée	vérifiez l'orthographe ; créez-la avec `khalli`
`'kan' da ma indah 'khalas'`	un bloc n'est pas fermé	ajoutez `khalas` à la fin du bloc
`ma fihimt …`	Kalam ne comprend pas l'écriture	vérifiez parenthèses, virgules, guillemets
`nidôr …, ligît …`	Kalam attendait autre chose à cet endroit	lisez ce qui est attendu (« attendu … »)
`guillemet fermant manquant`	texte non fermé	ajoutez le `"` de fin
`ma tagdar tisawwi '+' bên kitaba wa nimra` (ou `-`, `*`…)	opération entre types incompatibles	convertissez avec `nimra()` ou `kitaba()`
`ma tagdar tigassim be sifir`	division par zéro	vérifiez le diviseur avant de diviser
`index … marag barra`	position qui n'existe pas dans la liste	rappelez-vous : on compte à partir de 0
`… da ma nimra`	`nimra()` sur un texte qui n'est pas un nombre	utilisez `jarrib … mushkila`
`… bidôr 2 argument, inta antêtah 1`	mauvais nombre de valeurs données à une fonction	vérifiez l'appel
`'agif' lâzim yabga juwa dôra`	`agif` ou `fut` hors d'une boucle	placez-le dans `kulla` ou `tulma`
`'rajja' lâzim yabga juwa khidime`	`rajja` hors d'une fonction	placez-le dans une `khidime`

Annexe D — Solutions des exercices

Il y a souvent plusieurs bonnes solutions. Si la vôtre est différente mais donne le bon résultat, elle est juste !

3.1

```
ori("Mahamat")
ori("Ndjamena")
```

RÉSULTAT

```
Mahamat
Ndjamena
```

3.2

```
ori("Fi as-sana indaha", 365 * 24, "sâ'a")
```

RÉSULTAT

```
Fi as-sana indaha 8760 sâ'a
```

4.1

```
khalli sana_al_wilâda = 2008
ori("Umri fi 2026:", 2026 - sana_al_wilâda)
```

RÉSULTAT

```
Umri fi 2026: 18
```

4.2

```
khalli rasîd = 5000
rasîd -= 1200
rasîd += 3000
ori(rasîd)
```

RÉSULTAT

```
6800
```

5.1

```
ori(shakal("42"), shakal(42))
```

RÉSULTAT
```
kitaba nimra
```

5.2

```
khalli x = "7"
khalli y = "8"
ori(nimra(x) + nimra(y))
```

RÉSULTAT
```
15
```

6.1

```
ori(500 * 2 * 22, "FCFA")
```

RÉSULTAT
```
22000 FCFA
```

6.2

```
ori(100 % 7)
```

RÉSULTAT
```
2
```

6.3

```
ori("*" * 10)
ori("* Kalam  *")
ori("*" * 10)
```

RÉSULTAT
```
**********
* Kalam  *
**********
```

7.1

```
khalli taman = nimra(asal("Taman al-haja: "))
khalli kammiye = nimra(asal("Al-kammiye: "))
ori("Al-jumla:", taman * kammiye, "FCFA")
```

RÉSULTAT

```
Taman al-haja: 750
Al-kammiye: 3
Al-jumla: 2250 FCFA
```

7.2

```
khalli c = nimra(asal("Al-harâra (°C): "))
ori(c * 9 / 5 + 32, "°F")
```

RÉSULTAT

```
Al-harâra (°C): 35
95 °F
```

8.1

```
khalli n = nimra(asal("Nimra: "))
kan n % 2 == 0
    ori(n, "zôj (pair)")
walla
    ori(n, "fard (impair)")
khalas
```

RÉSULTAT

```
Nimra: 17
17 fard (impair)
```

8.2

```
khalli nugta = nimra(asal("Al-nugta: "))
kan nugta >= 16
    ori("Mumtâz")
walla kan nugta >= 12
    ori("Sameh shadîd")
walla kan nugta >= 10
    ori("Sameh")
walla
    ori("Ma najah")
khalas
```

RÉSULTAT

```
Al-nugta: 13
Sameh shadîd
```

8.3

```
kulla sana fi [2024, 1900, 2000]
    kan (sana % 4 == 0 wa sana % 100 != 0) aw sana % 400 == 0
        ori(sana, "bissextile")
    walla
        ori(sana, "ma bissextile")
    khalas
khalas
```

RÉSULTAT

```
2024 bissextile
1900 ma bissextile
2000 bissextile
```

9.1

```
kulla i fi min 1 lahaddi 10
    ori(i * 2)
khalas
```

RÉSULTAT

```
2
4
6
8
10
12
14
16
18
20
```

9.2

```
khalli natîja = 1
kulla i fi min 1 lahaddi 10
    natîja *= i
khalas
ori(natîja)
```

RÉSULTAT

```
3628800
```

9.3

```
khalli majmû = 0
tulma sahi
    khalli n = nimra(asal("Nimra (0 le tagîf): "))
    kan n == 0
        agif
    khalas
    majmû += n
khalas
ori("Al-majmû':", majmû)
```

RÉSULTAT

```
Nimra (0 le tagîf): 10
Nimra (0 le tagîf): 25
Nimra (0 le tagîf): 7
Nimra (0 le tagîf): 0
Al-majmû': 42
```

9.4

```
kulla i fi min 1 lahaddi 5
    ori("*" * i)
khalas
```

RÉSULTAT

```
*
**
***
****
*****
```

10.1

```
khalli nugât = [11, 7, 15, 13, 9]
khalli adad = 0
kulla n fi nugât
    kan n >= 10
        adad += 1
    khalas
khalas
ori(adad, "najaho")
```

RÉSULTAT

```
3 najaho
```

10.2

```
khalli l = [4, 19, 7, 23, 11]
khalli kabîr = l[0]
kulla x fi l
    kan x > kabîr
        kabîr = x
    khalas
khalas
ori(kabîr)
```

RÉSULTAT

```
23
```

10.3

```
khalli asâmi = []
kulla i fi min 1 lahaddi 3
    zid(asâmi, asal("Usum " + i + ": "))
khalas
ori(rattib(asâmi))
```

RÉSULTAT

```
Usum 1: Zara
Usum 2: Ali
Usum 3: Mahamat
['Ali', 'Mahamat', 'Zara']
```

11.1

```
khalli jumla_hana = asal("Aktib jumla: ")
ori(gadur(gassim(naddif(jumla_hana))), "kalimât")
```

RÉSULTAT

```
Aktib jumla: al-yôm ana gâ'id na'allim Kalam
5 kalimât
```

11.2

```
khalli adad = 0
kulla h fi saghayyar("Al-arabi at-tchadi")
    kan h == "a"
        adad += 1
    khalas
khalas
ori(adad)
```

RÉSULTAT

```
5
```

11.3

```
khalli natîja = []
kulla kilma fi gassim("mahamat ali")
    zid(natîja, kabir(kilma[0]) + gata(kilma, 1))
khalas
ori(lumm(natîja, " "))
```

RÉSULTAT

```
Mahamat Ali
```

12.1

```
khalli telefôn = {marka: "Tecno", taman: 85000, lôn: "azrag"}
kulla k fi telefôn
    ori(k, ":", telefôn[k])
khalas
```

RÉSULTAT

```
marka : Tecno
taman : 85000
lôn : azrag
```

12.2

```
khalli adad = {}
kulla kilma fi gassim("shay sukkar shay laban shay sukkar")
    kan kilma fi adad
        adad[kilma] += 1
    walla
        adad[kilma] = 1
    khalas
khalas
ori(adad)
```

RÉSULTAT

```
{'shay': 3, 'sukkar': 2, 'laban': 1}
```

13.1

```
khidime zôj(n)
    rajja n % 2 == 0
khalas
ori(zôj(4), zôj(7))
```

RÉSULTAT

```
sahi ghalat
```

13.2

```
khidime mu'addal(liste)
    rajja jumla(liste) / gadur(liste)
khalas
ori(mu'addal([10, 14, 15]))
```

RÉSULTAT
```
13
```

13.3

```
khidime aktar_min_tinên(a, b)
    kan a > b
        rajja a
    khalas
    rajja b
khalas
ori(aktar_min_tinên(3, 9), aktar_min_tinên(12, 5))
```

RÉSULTAT
```
9 12
```

13.4

```
khidime fibonacci(n)
    kan n < 2
        rajja n
    khalas
    rajja fibonacci(n - 1) + fibonacci(n - 2)
khalas

khalli l = []
kulla i fi min 0 lahaddi 9
    zid(l, fibonacci(i))
khalas
ori(l)
```

RÉSULTAT
```
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

14.1

```
khidime gassim_be(a, b)
    kan b == 0
        zagil "ma tagdar tigassim be sifir"
    khalas
    rajja a / b
khalas

jarrib
    ori(gassim_be(10, 4))
    ori(gassim_be(10, 0))
mushkila m
    ori("Mushkila:", m)
khalas
```

RÉSULTAT

```
2.5
Mushkila: ma tagdar tigassim be sifir
```

14.2

```
jarrib
    khalli n = nimra(asal("Nimra: "))
    ori("Shukran:", n)
mushkila
    ori("Da ma nimra")
khalas
```

RÉSULTAT

```
Nimra: ashara
Da ma nimra
```

15.1

```
aktib("journal.txt", al_yom() + "\n", sahi)
ori(agri("journal.txt"))
```

EXEMPLE DE RÉSULTAT

```
24/11/2026
```

15.2 — Fichier `riyadiyât.klm` :

Fichier `riyadiyat.klm`

```
khidime murabba'(x)
    rajja x * x
khalas

khidime mu'addal(liste)
    rajja jumla(liste) / gadur(liste)
khalas
```

Fichier principal :

```
jib "riyadiyat"
ori(murabba'(9), mu'addal([2, 4, 9]))
```

RÉSULTAT

```
81 5
```

16.1

```
khalli sitta = 0
kulla i fi min 1 lahaddi 1000
    kan sudfa(1, 6) == 6
        sitta += 1
    khalas
khalas
ori(sitta, "marra 6 min 1000")
```

EXEMPLE DE RÉSULTAT

```
189 marra 6 min 1000
```

Annexe E — Petit lexique

Les mots tchadiens utilisés dans les exemples de ce cours.

Mot	Sens	Mot	Sens
`usum`	nom	`taman`	prix
`umur`	âge	`gurûsh`	argent
`sana`	année	`rasîd`	solde
`hille`	ville, village	`jumla`	total
`bêt`	maison	`kammiye`	quantité
`talib`	élève	`dukkân`	boutique
`fasil`	classe	`haja`	chose, article
`nugta`	note	`khudâr`	légumes
`radd`	réponse	`bada'a`	marchandise
`natîja`	résultat	`fâtûra`	facture
`marra`	fois	`yôm`	jour
`kalimât`	mots	`shahar`	mois
`satir`	ligne	`harâra`	chaleur, température
`sirr`	secret	`kabîr` / `saghayyar`	grand / petit
`jadîd`	nouveau	`sameh`	bien, bon
`mushkila`	problème	`mabrûk`	félicitations

Kalam est un logiciel libre (licence MIT). Le code source, les exemples et l'extension VS Code sont sur GitHub : github.com/salah-barkai/kalam. Le terrain de jeu en ligne : salah-barkai.github.io/kalam.