Institut Qalâmi



INFORMATIQUE · PYTHON

Tes 5 premiers programmes en Python

Cinq programmes à taper toi-même, du plus simple au plus complet, expliqués ligne par ligne, avec un défi pour chacun. À partir de 12 ans.

Avant de commencer

Programmer, c'est écrire des instructions que l'ordinateur exécute dans l'ordre, une ligne après l'autre. Dans ce guide, tu écris cinq vrais programmes : un qui parle, une calculatrice, un jeu, un quiz et un dessin.

Ouvrir Python sur ton ordinateur

Il te faut un ordinateur Windows ou Mac (sous Linux, Python est en général déjà installé). Fais l'installation avec un parent, et télécharge uniquement depuis l'un de ces deux sites.

Solution 1 : Thonny, la plus simple

1. Sur **thonny.org**, télécharge Thonny, un éditeur gratuit pensé pour les débutants.
2. Installe-le : Python est déjà inclus.
3. Écris ton code dans la grande zone du haut et enregistre-le en `.py`.
4. Clique sur le bouton vert ▶ ou **F5**. Le résultat s'affiche dans la zone *Shell*, en bas : c'est là que tu tapes tes réponses.

Solution 2 : Python et IDLE

1. Sur **python.org**, menu *Downloads*, télécharge la dernière version de Python 3 et installe-la.
2. Ouvre **IDLE**, l'éditeur installé avec Python (cherche « IDLE » dans tes applications), puis *File* › *New File*.
3. Écris ton code, puis *File* › *Save* avec un nom qui finit par `.py`.
4. Lance-le avec *Run* › *Run Module* ou **F5**. Le résultat s'affiche dans la fenêtre *IDLE Shell*.

Les règles à connaître avant la première ligne

1. **Majuscules et minuscules comptent.** `print` fonctionne, `Print` provoque une erreur.
2. **Un texte se met entre guillemets droits** `"comme ceci"`, ouverts et fermés. Les guillemets français (« et ») ne marchent pas.
3. **Les parenthèses vont toujours par deux** : chaque `(` a sa `)`.
4. **Après `if`, `elif`, `else`, `while`, `for` et `def`, on met deux-points**, et les lignes qui suivent sont décalées de 4 espaces. IDLE et Thonny font ce décalage tout seuls quand tu appuies sur Entrée.
5. **Les nombres à virgule s'écrivent avec un point** : `2.5` et non `2,5`.
6. **Un nom de variable** s'écrit sans espace et ne commence pas par un chiffre : `score`, `score_final`, `joueur2`.
7. **Ne donne pas à ton fichier le nom d'un module de Python**, comme `random.py` ou `turtle.py` : Python chargerait ton fichier à la place du vrai module.

COMMENT LIRE CE GUIDE

Le code à taper est sur fond noir ; ce qui s'affiche est sur fond clair, et **ce que tu tapes au clavier y est souligné en or**. Recopie le code toi-même : c'est en tapant qu'on retient, et un copier-coller depuis un PDF abîme souvent les espaces.

01 L'ordinateur te dit salam

Nouveau · print() · input() · les variables · int()

Ton premier programme pose deux questions et répond avec ce que tu as tapé. C'est le principe de la plupart des jeux : le programme écoute, puis il réagit.

```
# Mon premier programme
print("Salam ! Je suis ton ordinateur.")
prenom = input("Comment tu t'appelles ? ")
print("Enchanté,", prenom, "!")
age = int(input("Quel âge as-tu ? "))
print("Dans 10 ans, tu auras", age + 10, "ans.")
```

LIGNE PAR LIGNE

`# Mon premier programme`	Un commentaire : tout ce qui suit `#` est ignoré par Python.
`print("Salam ! ...")`	`print` affiche ce qu'il y a entre les parenthèses. Un texte entre guillemets s'appelle une **chaîne de caractères**.
`prenom = input("...")`	`input` affiche la question et attend ta réponse (validée par Entrée). Elle est rangée dans la **variable** `prenom` : ici, `=` veut dire « range dans ».
`print("Enchanté,", prenom, "!")`	`print` met une espace entre les éléments séparés par des virgules. Sans guillemets, `prenom` est remplacé par son contenu.
`age = int(input("..."))`	`input` renvoie toujours du texte, même si tu tapes 12. `int()` le transforme en nombre entier : Python peut alors calculer `age + 10`.

À L'ÉCRAN

```
Salam ! Je suis ton ordinateur.
Comment tu t'appelles ? Adam
Enchanté, Adam !
Quel âge as-tu ? 12
Dans 10 ans, tu auras 22 ans.
```

DÉFI

1. Demande aussi la ville, puis affiche par exemple : `Salam à toute la ville de Lyon !`
2. Demande l'année de naissance et affiche l'année de tes 18 ans.
3. Réponds **douze** en lettres à la question de l'âge. Quelle erreur s'affiche, et pourquoi ? (Voir la page des erreurs.)

ASTUCE : LA CONSOLE

Dans la zone *Shell* de Thonny (ou la fenêtre *IDLE Shell*), tape une instruction après `>>>` et appuie sur Entrée : Python répond tout de suite. C'est pratique pour tester une idée avant de l'écrire dans ton programme.

```
>>> 7 * 8
56
>>> "Salam " * 3
'Salam Salam Salam '
>>> len("Python")
6
```

02 Une calculatrice

Nouveau | float() | if, elif, else | == | les opérateurs

Le programme demande deux nombres et une opération, puis il choisit quoi faire selon ta réponse. Choisir selon une condition, c'est ce que font presque tous les programmes.

```
# Ma calculatrice
print("=== Calculatrice ===")
a = float(input("Premier nombre : "))
operation = input("Opération (+ - * /) : ")
b = float(input("Deuxième nombre : "))

if operation == "+":
    print("Résultat :", a + b)
elif operation == "-":
    print("Résultat :", a - b)
elif operation == "*":
    print("Résultat :", a * b)
elif operation == "/":
    if b == 0:
        print("Impossible : on ne divise pas par zéro.")
    else:
        print("Résultat :", a / b)
else:
    print("Je ne connais pas cette opération.")
```

LIGNE PAR LIGNE

`a = float(input("..."))`	`float()` transforme le texte en nombre à virgule : c'est pour cela que 7 * 6 affiche `42.0`.
`if operation == "+":`	« Si l'opération est + ». `==` compare deux valeurs (est-ce égal ?), alors que `=` range une valeur. La ligne finit par deux-points.
`    print("Résultat :", a + b)`	Décalée de 4 espaces : elle ne s'exécute que si la condition au-dessus est vraie.
`elif operation == "-":` `else:`	`elif` : « sinon, si », testé seulement si les conditions d'avant étaient fausses. `else` : « dans tous les autres cas ».
`    if b == 0:`	Un `if` dans un autre. Sans ce test, diviser par zéro arrêterait le programme (`ZeroDivisionError`).

À L'ÉCRAN

```
=== Calculatrice ===
Premier nombre : 7
Opération (+ - * /) : *
Deuxième nombre : 6
Résultat : 42.0
```

AVEC UNE DIVISION

```
=== Calculatrice ===
Premier nombre : 10
Opération (+ - * /) : /
Deuxième nombre : 4
Résultat : 2.5
```

LES OPÉRATEURS DE PYTHON

Opérateur	`+`	`-`	`*`	`/`	`//`	`%`	`**`
Rôle	addition	soustraction	multiplication	division	division entière	reste	puissance
Exemple	`7 + 2`	`7 - 2`	`7 * 2`	`7 / 2`	`7 // 2`	`7 % 2`	`7 ** 2`
Résultat	9	5	14	3.5	3	1	49

LE SAVAIS-TU ?

Calcule 0.1 + 0.2 : Python affiche `0.30000000000000004`. L'ordinateur range les nombres à virgule en binaire, et 0.1 ne peut pas s'y écrire exactement. Ce n'est pas une erreur de ton programme. `round(0.1 + 0.2, 2)` arrondit à `0.3`.

DÉFI

1. Ajoute la puissance `**` et le reste `%` avec deux nouveaux `elif`.
2. Accepte aussi la lettre x pour multiplier. Indice : `elif operation == "*" or operation == "x":`

---

03 Le nombre mystère

Nouveau · import · random.randint() · while · != · un compteur

L'ordinateur choisit un nombre secret entre 1 et 100, et tu dois le trouver. À chaque essai, il te dit si c'est plus grand ou plus petit. C'est ton premier vrai jeu.

```
# Le nombre mystère
import random

mystere = random.randint(1, 100)
essais = 0
proposition = 0

print("Devine mon nombre, entre 1 et 100 !")

while proposition != mystere:
    proposition = int(input("Ta proposition : "))
    essais = essais + 1
    if proposition < mystere:
        print("C'est plus grand.")
    elif proposition > mystere:
        print("C'est plus petit.")

print("Bravo ! Trouvé en", essais, "essais.")
```

LIGNE PAR LIGNE

`import random`	Charge `random`, la boîte à outils de Python pour le hasard.
`random.randint(1, 100)`	Tire un nombre entier au hasard entre 1 et 100, 1 et 100 compris.
`proposition = 0`	Préparée avant la boucle. Le nombre secret n'est jamais 0 : la boucle peut donc démarrer.
`while proposition != mystere:`	« Tant que la proposition est différente (`!=`) du nombre secret ». Les lignes décalées se répètent tant que c'est vrai : c'est une **boucle**.
`essais = essais + 1`	Un **compteur** : la valeur actuelle plus 1. Raccourci : `essais += 1`.
`if proposition < mystere:`	`<` : « plus petit que » ; `>` : « plus grand que ».
`print("Bravo ! ...")`	Pas décalée : en dehors de la boucle. Elle ne s'exécute qu'une fois, quand tu as trouvé.

À L'ÉCRAN (LE NOMBRE CHANGE À CHAQUE PARTIE)

```
Devine mon nombre, entre 1 et 100 !
Ta proposition : 50
C'est plus petit.
Ta proposition : 25
C'est plus grand.
Ta proposition : 37
Bravo ! Trouvé en 3 essais.
```

LA MÉTHODE DU MILIEU

Propose toujours le milieu de ce qui reste possible : 50, puis 25 ou 75, et ainsi de suite. Chaque essai coupe les possibilités en deux. Avec cette méthode, tu trouves n'importe quel nombre entre 1 et 100 **en 7 essais au maximum**. On l'appelle la **recherche dichotomique**.

DÉFI

1. Limite la partie à 7 essais. Indice : `while proposition != mystere and essais < 7:`, puis, après la boucle, un `if proposition == mystere:` qui dit si tu as gagné.
2. Fais un niveau difficile, avec un nombre entre 1 et 1000. Combien d'essais faut-il au maximum avec la méthode du milieu ? (Réponse à la page « Pause ».)

---

04 Un quiz avec score

Nouveau · les listes · for · range() · len() · strip() et lower()

Le programme pose des questions l'une après l'autre, vérifie chaque réponse et compte les points. Pour changer de quiz, il suffit de changer les deux listes.

```
# Mon quiz
questions = [
    "Combien font 7 x 8 ?",
    "Combien de côtés a un hexagone ?",
    "Combien de minutes dans une heure ?",
    "Quelle est la capitale du Maroc ?",
]
reponses = ["56", "6", "60", "rabat"]

score = 0
for i in range(len(questions)):
    reponse = input(questions[i] + " ")
    if reponse.strip().lower() == reponses[i]:
        print("Juste !")
        score = score + 1
    else:
        print("Raté. La bonne réponse était :", reponses[i])

print("Score :", score, "sur", len(questions))
if score == len(questions):
    print("Sans faute, bravo !")
```

LIGNE PAR LIGNE

`questions = [ ... ]`	Une **liste** : des valeurs entre crochets, séparées par des virgules.
`reponses = ["56", ...]`	Les bonnes réponses, dans le même ordre, en minuscules.
`for i in range(len(questions)):`	`len` donne la longueur de la liste (4) et `range(4)` produit 0, 1, 2, 3. La boucle `for` répète les lignes décalées pour chaque valeur, rangée dans `i`.
`questions[i]`	L'élément numéro `i`. On compte à partir de 0 : `questions[0]` est la première question.
`questions[i] + " "`	Entre deux textes, `+` les colle (ici, une espace après la question).
`reponse.strip().lower()`	`strip()` enlève les espaces autour, `lower()` met en minuscules : « Rabat » et « RABAT » sont acceptés.

À L'ÉCRAN

```
Combien font 7 x 8 ? 56
Juste !
Combien de côtés a un hexagone ? 8
Raté. La bonne réponse était : 6
Combien de minutes dans une heure ? 60
Juste !
Quelle est la capitale du Maroc ? Rabat
Juste !
Score : 3 sur 4
```

LA TABLE DE 7 EN DEUX LIGNES

```
for i in range(1, 11):
    print(7, "x", i, "=", 7 * i)
```

Affiche `7 x 1 = 7`, `7 x 2 = 14`… jusqu'à `7 x 10 = 70`. `range(1, 11)` va de 1 à 10 : le second nombre n'est jamais compris.

DÉFI

1. Écris ton propre quiz. Garde le même nombre d'éléments dans les deux listes, sinon Python peut s'arrêter avec `IndexError`.
2. Transforme le quiz en entraînement aux tables : 10 multiplications tirées au hasard. Indice : `a = random.randint(2, 9)` (après `import random`), pareil pour `b`, puis `input(str(a) + " x " + str(b) + " = ")` : `str()` change un nombre en texte. Pour vérifier, compare `int(reponse)` à `a * b`.

Pause : prédis ce qui s'affiche

Avant le dernier projet, vérifie que tu as compris. Écris ta réponse sur la ligne, puis lance le code pour vérifier.

1.

```
x = 5
x = x * 2
print(x + 1)
```

2.

```
for i in range(3):
    print(i)
```

3.

```
mot = "Python"
print(mot[0], len(mot))
```

4.

```
n = 1
while n < 20:
    n = n * 2
print(n)
```

5.

```
print(7 // 2, 7 % 2)
```

6.

```
print("3" + "4")
print(3 + 4)
```

Réponses de la pause. 1 : `11` . 2 : `0` , `1` et `2` , chacun sur sa ligne. 3 : `P 6` (les lettres d'un texte se numérotent comme une liste, à partir de 0). 4 : `32` . 5 : `3 1` . 6 : `34` puis `7` (deux textes se collent, deux nombres s'additionnent). Défi du projet 3 : 10 essais au maximum entre 1 et 1000.

05 Une rosace avec la tortue

Nouveau · turtle · def : les fonctions · un paramètre · %

La tortue est un crayon que tu pilotes : avance, tourne, change de couleur. En répétant une forme simple et en tournant un peu à chaque fois, on obtient une rosace.

```
# La rosace de la tortue
import turtle

t = turtle.Turtle()
t.speed(0)
t.pensize(2)

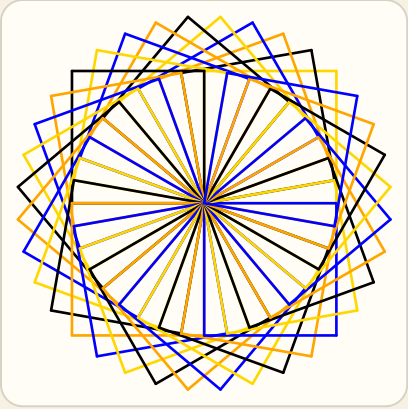
def carre(cote):
    for i in range(4):
        t.forward(cote)
        t.left(90)

couleurs = ["gold", "black", "orange", "blue"]
for n in range(36):
    t.color(couleurs[n % 4])
    carre(100)
    t.left(10)

turtle.done()
```

LIGNE PAR LIGNE

`t = turtle.Turtle()`	Crée une tortue `t` dans une fenêtre. Elle part du centre, tournée vers la droite.
`t.speed(0)` · **`t.pensize(2)`**	Vitesse la plus rapide (0 = sans animation), trait de 2 pixels.
`def carre(cote):`	Définit une **fonction**, une recette nommée `carre` , avec un **paramètre** `cote` . Rien n'est encore dessiné.
`t.forward(cote)` **`t.left(90)`**	Avance de `cote` pixels, tourne de 90 degrés à gauche. Quatre fois : un carré.
`couleurs[n % 4]`	`n % 4` , le reste de la division par 4, vaut 0, 1, 2, 3, 0, 1… : les couleurs reviennent à tour de rôle.
`carre(100)`	Appelle la fonction : le carré est dessiné maintenant, avec un côté de 100.
`t.left(10)`	La tortue tourne de 10 degrés après chaque carré : 36 × 10 = 360 degrés, un tour complet.
`turtle.done()`	Garde la fenêtre ouverte jusqu'à ce que tu la fermes.



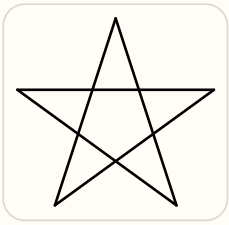
À L'ÉCRAN

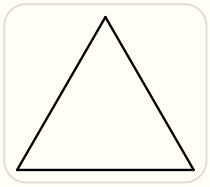
Les 36 carrés tournent autour du centre et forment une rosace en or, noir, orange et bleu.

Remplace `t.speed(0)` par `t.speed(1)` pour regarder la tortue dessiner chaque carré, lentement.

DÉFI

1. Dessine une étoile à 5 branches : répète 5 fois `t.forward(150)` puis `t.right(144)`.
2. Écris une fonction `triangle(cote)` : 3 côtés, et un virage de 120 degrés à chaque coin.
3. Fais une rosace de 72 carrés en tournant de 5 degrés à chaque fois.





Quand Python affiche une erreur

Une erreur n'est pas un échec : Python te dit ce qui ne va pas. Regarde la fin du message : le nom de l'erreur et le numéro de la ligne t'indiquent où chercher.

ERREUR	CAUSE LA PLUS FRÉQUENTE	EXEMPLE FAUTIF	CORRECTION
SyntaxError	Guillemet ou parenthèse non fermé, deux-points oubliés, `=` au lieu de `==`	`print("Salam)` `if x = 3:`	`print("Salam")` `if x == 3:`
IndentationError	Ligne pas décalée après les deux-points	`if x == 3:` `print(x)`	Décaler `print(x)` de 4 espaces
NameError	Nom mal écrit, majuscule en trop, variable pas encore créée	`Print("Salam")` `print(prenon)`	`print("Salam")` `print(prenom)`
ValueError	`int()` reçoit un texte qui n'est pas un nombre	`int("douze")`	Taper `12` en chiffres
TypeError	Texte et nombre mélangés avec `+`	`"Tu as " + 12`	`"Tu as " + str(12)`
IndexError	Numéro trop grand dans une liste	Liste de 2 éléments, puis `liste[2]`	Les numéros vont de 0 à 1

Et ensuite ? Cinq projets avec ce que tu sais déjà

- **Le dé** : un nombre de 1 à 6 à chaque appui sur Entrée (`random.randint`, `while`).
- **Pierre, feuille, ciseaux** : l'ordinateur joue au hasard, le programme dit qui gagne (`random.choice`, `if`, `and`).
- **Le compte à rebours** : 10, 9, 8… 1, puis « Décollage ! » (`range(10, 0, -1)`).
- **Le convertisseur** : 135 minutes, c'est 2 h 15 (`//` et `%`).
- **Le dessin libre** : une maison, un soleil, avec la tortue (`t.penup()`, `t.goto(x, y)`).

Le mémo des notions

Tout ce que tu as vu dans ces cinq projets, avec quelques outils en plus, sur une page. Garde-la à côté de ton clavier.

POUR…	ON ÉCRIT	À RETENIR
Afficher	`print("Salam", prenom)`	Les virgules ajoutent une espace.
Demander	`nom = input("Ton nom ? ")`	Renvoie toujours du texte.
Ranger	`score = 0` · `# une note`	`=` range une valeur dans une variable ; `#` commence un commentaire.
Convertir	`int("12")` · `float("2.5")` · `str(12)`	Texte vers entier, vers nombre à virgule ; nombre vers texte.
Calculer	`+ - * / // % **`	`//` division entière, `%` reste, `**` puissance.
Comparer	`== != < > <= >=` · `and` · `or`	Égal, différent, plus petit, plus grand… `and` : les deux vraies ; `or` : au moins une.
Choisir	`if ... :` · `elif ... :` · `else:`	Deux-points, puis lignes décalées de 4 espaces.
Répéter tant que	`while essais < 7:`	Tourne tant que la condition est vraie.
Répéter n fois	`for i in range(5):` · `range(1, 11)` · `range(10, 0, -1)`	`i` vaut 0, 1, 2, 3, 4 ; de 1 à 10 ; de 10 à 1.
Les listes	`fruits = ["datte", "figue"]` · `fruits.append("pomme")`	`fruits[0]` vaut `"datte"` ; `append` ajoute à la fin ; `len(fruits)` donne la longueur.
Le hasard	`random.randint(1, 6)` · `random.choice(fruits)`	Après `import random` : un entier de 1 à 6 compris ; un élément de la liste au hasard.
Une fonction	`def carre(cote):` · `carre(100)`	On la définit une fois, on l'appelle autant qu'on veut.
Le texte	`mot.lower()` · `mot.strip()` · `"Sa" + "lam"`	En minuscules ; sans espaces autour ; textes collés.
Texte et variables	`print(f"Salam {prenom} !")`	Le `f` remplace `{prenom}` par sa valeur.

Mon carnet de codeur

Coche chaque étape quand elle est faite et note en une phrase ce que tu as appris.

PROJET	TERMINÉ LE	IL MARCHE	DÉFI RÉUSSI	CE QUE J'AI APPRIS
01 Salam		☐	☐	
02 Calculatrice		☐	☐	
03 Nombre mystère		☐	☐	
04 Quiz		☐	☐	
05 Rosace		☐	☐	

Votre fils a pris goût au code ?

S'il souhaite aller plus loin, en petit groupe et avec un enseignant, nous serons heureux d'en parler avec vous.

WhatsApp : +971 52 151 9314

contact.qalami@gmail.com

institutqalami.com/informatique