SNT — Introduction à Python

Polycopié élève · parties 0 à 8 et passerelle Folium

Mode d'emploi

- Le cours alterne des parties théoriques courtes et des exercices. On avance à son rythme.
- Version courte : seuls les exercices essentiels sont reproduits, avec les mêmes numéros que sur le site (où les 85 exercices sont proposés). Le site donne aussi le code à copier et des indices dépliables.
- Difficulté : 🌶 prise en main · 🌶🌶 application · 🌶🌶🌶 plusieurs étapes · 🌶🌶🌶🌶 défi.
- Un fichier par exercice, nommé `NOM_Prenom_ex05.py`, rangé dans ton dossier SNT. Les réponses écrites vont en commentaire `# ...` ou dans le traitement de texte.

Partie 0 — Prise en main d'EduPython

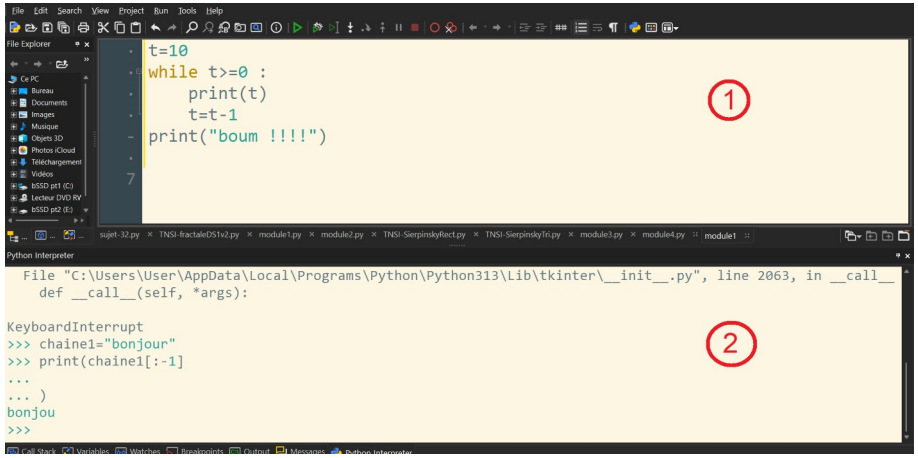
Ouvrir, écrire, exécuter, enregistrer, rendre

Toute l'année, tu vas écrire de vrais programmes en **Python**, le langage utilisé au lycée en SNT, en mathématiques et plus tard en NSI. On programme avec **EduPython**, un éditeur installé sur les ordinateurs du lycée.

Cours · Ouvrir EduPython

1. Sur un ordinateur du lycée, ouvre la suite **MCNL**.
2. Choisis la matière **SNT-NSI**.
3. Clique sur l'icône **EduPython**.

Sur ton ordinateur personnel, la démarche est la même dès que la suite y est installée.



La fenêtre d'EduPython : l'éditeur (1) et la console (2).

Zone	À quoi elle sert
① l'éditeur	On y écrit un **programme** complet. On l'exécute avec la flèche verte ▶. Il s'enregistre dans un fichier `.py`.
② la console	On y tape **une** instruction, on appuie sur Entrée, le résultat s'affiche aussitôt. Rien n'est enregistré.

À retenir

La **console** sert à essayer rapidement un calcul ou une commande.

L'**éditeur** sert à écrire un programme que l'on garde. Dans un programme, seul ce que l'on demande avec `print(...)` s'affiche.

Méthode · Ranger son travail

1. Dans tes *Documents*, crée un dossier nommé `SNT_Python`. Tous tes programmes iront dedans.
2. Chaque exercice fait l'objet d'un fichier nommé `NOM_Prenom_ex05.py` (ici pour l'exercice 5 : le numéro toujours sur deux chiffres).
3. Pour enregistrer : menu *File* → *Save As…* la première fois, puis Ctrl + S.

Quand un exercice pose une question, tu réponds **dans le fichier**, sur une ligne qui commence par # : c'est un **commentaire**, Python ne le lit pas. Tu peux aussi répondre dans ton traitement de texte si le professeur le demande.

DUPONT_Lea_ex05.py

```
# DUPONT Léa - 2nde - exercice 05
# Question 1 : la tortue trace un carré.
print("Bonjour")
```

> **Vigilance**
>
> Pas d'espace ni d'accent dans les noms de fichiers : `DUPONT_Lea_ex05.py` et non `Dupont Léa ex 5.py`.
>
> En binôme (première séance), mets les deux noms : `DUPONT_MARTIN_ex01.py`.

Méthode · Rendre son travail

1. À la fin de la séance, enregistre tous tes fichiers.
2. Dans l'explorateur Windows, fais un clic droit sur le dossier `SNT_Python`, puis *Compresser dans un fichier ZIP* (Windows 11) ou *Envoyer vers → Dossier compressé* (Windows 10).
3. Si le professeur le demande, dépose le fichier `.zip` obtenu sur Pronote, dans le travail à rendre.

> **Vigilance**
>
> Le `.zip` contient une **copie** de tes fichiers au moment où tu l'as créé. Si tu modifies un programme après, il faut refaire le `.zip`.

Exercices · Premiers essais

Exercice 1 — Console ou programme ? 🌶

1. Dans la console, tape `5**7` puis appuie sur Entrée. Quel résultat obtiens-tu ?
2. Dans l'éditeur, écris `5**7`, puis exécute avec ▶. Que se passe-t-il ?
3. Remplace la ligne par `print(5**7)` et exécute à nouveau.
4. Enregistre le fichier sous le nom `NOM_Prenom_ex01.py` dans ton dossier `SNT_Python`. Écris la réponse à la question 2 en commentaire dans le fichier.

Fichier : NOM_Prenom_ex01.py

Exercice 2 — Mon premier programme 🌶

Écris un programme qui affiche, sur trois lignes différentes :

- `Bonjour !`
- ton prénom
- `J'apprends Python.`

Enregistre-le sous `NOM_Prenom_ex02.py`.

Fichier : NOM_Prenom_ex02.py

Partie 1 — La tortue dessine

Instructions, variables et première boucle, en dessinant

Le module **turtle** permet de piloter une tortue qui dessine en se déplaçant. Chaque instruction la fait avancer ou tourner : tu vois immédiatement la conséquence de chacun de tes choix.

Cours · Premières commandes

Au départ, la tortue est au centre de l'écran, au point de coordonnées $(0\,;0)$, et elle regarde vers la droite. Les longueurs sont en **pixels**, les angles en **degrés**.

premiers_pas.py

```
from turtle import *   # on charge les commandes de la tortue

forward(120)   # avance de 120 pixels
left(90)       # tourne à gauche de 90 degrés
forward(50)    # avance de 50 pixels
done()         # garde la fenêtre ouverte à la fin
```

Commande	Effet
`forward(n)`	avance de n pixels
`back(n)`	recule de n pixels
`left(a)`	tourne sur place à gauche de a degrés
`right(a)`	tourne sur place à droite de a degrés
`done()`	termine le dessin et garde la fenêtre ouverte

Vigilance

La première ligne est toujours `from turtle import *` et la dernière `done()`.

Ferme la fenêtre de la tortue avant de relancer ton programme. Si EduPython ne répond plus, réinitialise le moteur Python avec Ctrl + F2.

Exercices · À toi de piloter

Exercice 4 — Le carré

Écris un programme qui dessine un **carré de côté 100 pixels**.

Fichier : NOM_Prenom_ex04.py

Exercice 5 — Le rectangle

Écris un programme qui dessine un **rectangle de 150 pixels sur 80 pixels**.

Fichier : NOM_Prenom_ex05.py

Cours · Les variables

Une **variable** est une boîte qui porte un nom et qui contient une valeur. On l'utilise pour ne pas répéter un même nombre partout : si on change la valeur, tout le dessin suit.

carre_variable.py

```
from turtle import *
cote = 100          # la variable cote reçoit la valeur 100
forward(cote)
left(90)
forward(cote)
left(90)
forward(cote)
left(90)
forward(cote)
left(90)
done()
```

À retenir

`cote = 100` se lit « **cote reçoit 100** » : c'est une **affectation**, pas une égalité mathématique. La valeur de droite est rangée dans la variable de gauche.

Un nom de variable contient des lettres, des chiffres et _, sans espace ni accent, et ne commence pas par un chiffre : `cote`, `largeur_2`, `nb_tours`.

Exercices · Changer une valeur, voir les conséquences

Exercice 7 — Un carré réglable

Copie le programme `carre_variable.py`.

1. Exécute-le avec `cote = 50`, puis `cote = 150`, puis `cote = 300`. Combien de lignes as-tu modifiées à chaque fois ?
2. Essaie `cote = -100`. Que se passe-t-il ? Explique en commentaire.

Fichier : NOM_Prenom_ex07.py

Cours · Répéter : la boucle for

Pour répéter plusieurs fois les mêmes instructions, on utilise une **boucle** `for`.

carre_boucle.py

```
from turtle import *
for i in range(4):      # répète 4 fois le bloc décalé
    forward(100)
    left(90)
done()                  # n'est plus décalé : hors de la boucle
```

À retenir

La ligne `for i in range(4):` se termine par **deux-points** `:`.

Les lignes répétées forment un **bloc** : elles sont **décalées** vers la droite d'une tabulation (touche Tab, soit 4 espaces). La première ligne qui n'est plus décalée est **en dehors** de la boucle.

Vigilance

Oublier les `:` provoque une erreur `SyntaxError`.

Décaler ou ne pas décaler une ligne change complètement le programme : le décalage fait partie du code.

Exercices · Boucles et polygones

Exercice 9 — Prédire, puis tester 🌶

Voici un programme presque identique à `carre_boucle.py` :

carre_rate.py

```
from turtle import *
for i in range(4):
    forward(100)
left(90)
done()
```

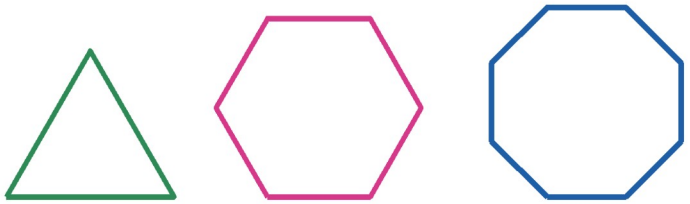
1. **Sans l'exécuter**, prévois en commentaire ce que va dessiner ce programme.
2. Exécute-le et vérifie ta prévision.

Fichier : NOM_Prenom_ex09.py

Exercice 10 — Polygones réguliers 🌶🌶

Pour faire le tour complet d'un polygone régulier à n côtés, la tortue tourne en tout de 360°. À chaque coin, elle tourne donc de $\frac{360}{n}$ degrés.

1. Recopie et complète ce tableau en commentaire : triangle équilatéral (3 côtés), carré (4), hexagone (6), octogone (8) → angle de virage.
2. Avec une boucle `for`, dessine un triangle équilatéral de côté 100.
3. Dessine de même un hexagone régulier, puis un octogone régulier.



Triangle, hexagone et octogone réguliers.Fichier : NOM_Prenom_ex10.py

Cours · Couleurs et déplacements

Commande	Effet
`pencolor('red')`	couleur du trait (en anglais : 'blue', 'green', 'pink', 'orange'…)
`pensize(3)`	épaisseur du trait
`fillcolor('yellow')`	couleur de remplissage
`begin_fill()` … `end_fill()`	remplit la figure tracée entre ces deux commandes
`penup()` / `pendown()`	lève / baisse le crayon : la tortue se déplace sans tracer / en traçant
`goto(x, y)`	va directement au point de coordonnées $(x\,;y)$
`speed(0)`	vitesse maximale
`hideturtle()`	cache la tortue

couleurs.py

```
from turtle import *
pencolor('red')
pensize(4)
fillcolor('yellow')
begin_fill()
for i in range(4):
    forward(100)
    left(90)
end_fill()
```

```
penup()           # on lève le crayon...
goto(150, 0)      # ...pour se déplacer sans tracer
pendown()
done()
```

À retenir · Synthèse

Ce que je sais faire

- Piloter la tortue : `forward`, `left`, `right`, `penup`, `goto`, couleurs.
- Ranger une valeur dans une **variable** et la réutiliser : `cote = 100`.
- Répéter un bloc avec `for i in range(n):` en soignant les `:` et le décalage.
- Choisir l'angle d'un polygone régulier : $\frac{360}{n}$ degrés.

Partie 2 — Calculer avec Python

Opérations, division euclidienne, puissances, racines

Python est une calculatrice très puissante. Dans cette partie, on travaille surtout dans la **console**, puis dans de petits programmes.

Cours · Les opérations

Opération	En Python	Exemple	Résultat
addition	`+`	`7 + 2`	9
soustraction	`-`	`7 - 2`	5
multiplication	`*`	`7 * 2`	14
division	`/`	`7 / 2`	3.5
puissance	`**`	`7 ** 2`	49
quotient euclidien	`//`	`7 // 2`	3
reste euclidien	`%`	`7 % 2`	1

Les priorités de calcul sont les mêmes qu'en mathématiques, et on utilise des parenthèses si besoin : `(2 + 3) * 4` donne 20.

À retenir : la division euclidienne

Pour deux entiers a et b avec $b>0$, il existe un unique quotient q et un unique reste r tels que :

$$a = b \times q + r \text{ avec } 0 \leq r < b$$

En Python : `q = a // b` et `r = a % b`.

Exemple : $17 = 5 \times 3 + 2$, donc `17 // 5` donne 3 et `17 % 5` donne 2.

Vigilance

La puissance s'écrit `**`, **jamais** `^` : `5^2` ne provoque pas d'erreur mais donne 7 !

Les nombres décimaux s'écrivent avec un **point** : `3.5` et non `3,5`.

`/` donne toujours un nombre décimal : `8 / 2` donne `4.0`.

Cours · Très grands et très petits nombres

Python écrit les nombres très grands ou très petits en **notation scientifique** : `4.562e7` signifie $4,562 \times 10^{7}$ et `5.12e-07` signifie $5,12 \times 10^{-7}$. On peut aussi taper les nombres ainsi : `3E8` pour 3×10^{8}.

Pour les racines carrées, on charge le module `math` :

console

```
>>> from math import *
>>> sqrt(57)
7.54983443527075
>>> pi
3.141592653589793
```

Exercice 17 — Division euclidienne

1. Détermine le quotient et le reste de la division euclidienne de 457 895 par 124.
2. Vérifie dans la console que `124 * q + r` redonne bien 457 895.

Fichier : NOM_Prenom_ex17.py

Exercice 18 — Un résultat bizarre 🌶🌶

1. Calcule $18^{18} \times 18^{-18}$ dans la console. Quel résultat devrait-on obtenir mathématiquement ?
2. Quel résultat Python affiche-t-il ? Propose une explication.

Fichier : NOM_Prenom_ex18.py

Exercice 19 — Prévoir l'affichage 🌶🌶

Sans les exécuter, prévois ce qu'affiche chacun de ces trois programmes. Vérifie ensuite.

programme A

```
x = 10
y = 2
print(x + y)
```

programme B

```
x = 10
y = x + 5
print(x + 2 * y)
```

programme C

```
x = 10
y = 2 * x
print(3 * y)
```

Fichier : NOM_Prenom_ex19.py

Exercice 20 — Des secondes aux heures 🌶🌶

Une course d'endurance a duré 55 555 secondes.

1. Avec `//` et `%`, convertis cette durée en heures, minutes et secondes.
2. Écris un programme qui range 55555 dans une variable `duree` puis affiche les trois résultats.

Fichier : NOM_Prenom_ex20.py

Exercice 23 — Les chiffres d'un nombre 🌶🌶🌶

On range `n = 2026`.

1. Que donne `n % 10` ? Et `(n // 10) % 10` ?
2. Trouve les instructions qui donnent le chiffre des centaines, puis celui des milliers.
3. Avec `%`, teste si 2026 est divisible par 2, 3, 5, 7 et 11.

Fichier : NOM_Prenom_ex23.py

À retenir · Synthèse

Ce que je sais faire

- Calculer avec `+ - * / **` en respectant les priorités.
- Obtenir quotient et reste euclidiens avec `//` et `%`.
- Lire et écrire la notation scientifique : `3E8`, `5.12e-07`.
- Utiliser `sqrt` après `from math import *`.
- Savoir qu'un calcul décimal est souvent **approché**.

Partie 3 — Dialoguer avec l'utilisateur

print, input… et un convertisseur de coordonnées GPS

Un programme devient vraiment utile quand il **pose des questions** et **affiche des réponses**. À la fin de cette partie, tu auras programmé un convertisseur de coordonnées GPS.

Cours · Afficher : print

`print` affiche plusieurs éléments séparés par des virgules. Python ajoute un espace entre eux et passe à la ligne à la fin.

afficher.py

```
a = 7
b = 5
print("La somme vaut", a + b)          # La somme vaut 12
```

```
print(f"{a} x {b} = {a * b}")          # 7 x 5 = 35
print(round(22 / 7, 3))                # 3.143
```

À retenir

Un texte s'écrit entre guillemets : `"La somme vaut"`.

Dans une **f-chaîne** `f"..."`, ce qui est entre accolades `{ }` est remplacé par sa valeur.

`round(x, 2)` arrondit x à 2 chiffres après la virgule.

Cours · Demander : input

bonjour.py

```
prenom = input("Quel est ton prénom ? ")
print("Bonjour", prenom)
```

`input` affiche la question, attend que l'utilisateur tape une réponse et appuie sur Entrée, puis **renvoie ce qui a été tapé**.

Vigilance : input renvoie toujours du texte

Si l'utilisateur tape 12, `input` renvoie le **texte** `"12"`, pas le nombre 12. Et `"12" + "3"` donne `"123"` !

Pour obtenir un nombre, on **convertit** : `int(...)` pour un entier, `float(...)` pour un nombre décimal.

À l'exécution, l'utilisateur doit taper les décimaux avec un **point** : `12.5`.

produit.py

```
a = int(input("Premier entier : "))
b = int(input("Second entier : "))
print(f"{a} x {b} = {a * b}")
```

Type	Nom en Python	Exemples	Conversion
entier	`int`	`12`, `-3`	`int("12")` → `12`
décimal	`float`	`12.5`, `3E8`	`float("12.5")` → `12.5`
texte (chaîne)	`str`	`"Bonjour"`, `"12"`	`str(12)` → `"12"`

Vigilance

`int(input("..."))` : **deux** parenthèses fermantes à la fin, une pour `input`, une pour `int`.

Exercices · Des programmes qui dialoguent

Exercice 25 — Bonjour 🌶

Écris un programme qui demande le prénom et l'âge de l'utilisateur, puis affiche par exemple :

console

```
Bonjour Léa, dans 10 ans tu auras 25 ans.
```

Fichier : NOM_Prenom_ex25.py

Exercice 26 — Le produit 🌶

Recopie `produit.py`, teste-le, puis modifie-le pour qu'il affiche aussi la **somme** et la **différence** des deux entiers.

Fichier : NOM_Prenom_ex26.py

Exercice 28 — Moyenne pondérée 🌶🌶

Écris un programme qui demande deux notes et leurs coefficients, puis affiche la moyenne pondérée arrondie au centième.

Test : 12 coefficient 2 et 15 coefficient 1 doivent donner 13.0.

Fichier : NOM_Prenom_ex28.py

Cours · Les coordonnées GPS

Une position sur Terre est repérée par deux angles :

- la **latitude** : position au nord ou au sud de l'équateur, de 0° à 90° ;
- la **longitude** : position à l'est ou à l'ouest du méridien de Greenwich, de 0° à 180°.

Ces angles s'écrivent de deux façons :

Écriture	Exemple : le lycée	Où la trouve-t-on ?
DD : degrés décimaux	44,1646° N ; 4,6103° E	cartes en ligne, applications, Folium
DMS : degrés, minutes, secondes	44° 9′ 52,6″ N ; 4° 36′ 37,2″ E	GPS de randonnée, cartes papier, Wikipédia

À retenir

Comme pour les heures : **1° = 60′** (minutes d'angle) et **1′ = 60″** (secondes d'angle), donc **1° = 3600″**.

Méthode : de DMS vers DD

$$\text{DD} = d + \frac{m}{60} + \frac{s}{3600}$$

Exemple, la tour Eiffel à 48° 51′ 30″ N :

$$\text{DD} = 48 + \frac{51}{60} + \frac{30}{3600} \approx 48,8583$$

Méthode : de DD vers DMS

Exemple avec la latitude du lycée, 44,1646098° :

1. Les degrés sont la **partie entière** : $d = 44$.
2. On multiplie la partie décimale par 60 : $0,1646098 \times 60 = 9,876588$, donc $m = 9$.
3. On multiplie la nouvelle partie décimale par 60 : $0,876588 \times 60 = 52,59528$, donc $s \approx 52,6$.

Résultat : **44° 9′ 52,6″ N**. En Python, la partie entière d'un nombre positif s'obtient avec `int` : `int(44.1646098)` donne 44.

Vigilance

Ces méthodes valent pour des coordonnées **positives** (nord et est). À l'ouest de Greenwich (Bretagne, Bordeaux…), la longitude est **négative** en DD : on verra dans la partie 4 comment gérer N/S/E/O.

Les calculs décimaux sont approchés (exercice 18) : on **arrondit** les secondes, par exemple à 0,1″ près.

Exercice 30 — À la main d'abord 🌶

1. Avec la calculatrice, convertis en DD la latitude de la tour Eiffel : 48° 51′ 30″.
2. Convertis en DMS sa longitude : 2,2945° E.

Fichier : NOM_Prenom_ex30.py

Exercice 31 — De DMS vers DD 🌶🌶

Écris un programme qui demande les degrés, les minutes et les secondes d'un angle, puis affiche sa valeur en degrés décimaux, arrondie à 6 chiffres après la virgule.

Teste-le avec la tour Eiffel (48° 51′ 30″) : tu dois retrouver ta réponse de l'exercice 30.

Fichier : NOM_Prenom_ex31.py

Exercice 32 — De DD vers DMS 🌶🌶🌶

Complète ce programme qui convertit un angle positif de DD vers DMS.

dd_vers_dms.py

```
dd = float(input("Angle en degrés décimaux : "))
d = int(dd)                    # partie entière
minutes_dec = (dd - d) * 60
m = ...
s = ...
print(f"{d}° {m}' {round(s, 1)}\"")
```

Tests : 44.1646098 doit donner `44° 9' 52.6"` et 4.6103368 doit donner `4° 36' 37.2"`.

Fichier : NOM_Prenom_ex32.py

À retenir · Synthèse

Ce que je sais faire

- Afficher des textes et des valeurs avec `print` et les f-chaînes.
- Demander une valeur avec `input` et la convertir avec `int` ou `float`.
- Arrondir avec `round`.
- Convertir des coordonnées GPS entre DD et DMS.

Partie F — Passerelle : mes coordonnées sur une carte

Utiliser Folium comme une application qu'on bidouille

Folium est un module qui fabrique des cartes interactives, comme celles des sites de cartographie. Son fonctionnement interne est complexe : on ne cherche pas à tout comprendre, on l'utilise et on le **modifie** pour obtenir la carte voulue.

Méthode · Installer Folium

À faire une seule fois par ordinateur. Folium est peut-être déjà installé sur les ordinateurs du lycée : essaie d'abord l'exercice 35.

1. Dans EduPython, menu *Outils → Outils → Installation d'un nouveau module*.
2. À la question « Votre choix : », tape 2 puis Entrée.
3. À la question « Nom du module : », tape `folium` puis Entrée.
4. Attends la fin de l'installation, puis appuie sur une touche quand c'est demandé.

Cours · Trois nouveautés

carte_lycee.py

```
import folium

carte = folium.Map(location=[44.1646, 4.6103], zoom_start=16)
folium.Marker([44.1646098, 4.6103368], popup="Lycée Albert
Einstein").add_to(carte)
carte.save("ma_carte.html")
```

Dans le programme	Ce que ça veut dire
`import folium`	charge le module ; ses commandes s'écrivent ensuite `folium.Map`, `folium.Marker`…
`[44.1646, 4.6103]`	une **liste** : plusieurs valeurs rangées dans l'ordre entre crochets. Ici **latitude d'abord**, longitude ensuite, en **DD**.
`zoom_start=16`, `popup="..."`	des **paramètres nommés** : on précise le nom du réglage qu'on modifie.
`carte.save("ma_carte.html")`	enregistre la carte dans un fichier web, dans **le même dossier que le programme**.

Vigilance

La carte ne s'affiche pas dans EduPython : ouvre le fichier `ma_carte.html` en double-cliquant dessus, dans ton dossier SNT_Python.

Le fond de carte se télécharge depuis Internet : il faut être connecté.

Exercices · Ma carte

Exercice 35 — La carte du lycée

1. Copie le programme `carte_lycee.py`, exécute-le, puis ouvre `ma_carte.html`.
2. Clique sur le marqueur. Que se passe-t-il ?
3. Change `zoom_start` : essaie 5, 10 puis 18. Que règle ce paramètre ?

Fichier : NOM_Prenom_ex35.py

Exercice 38 — Le programme complet

Écris un programme qui :

- demande le nom d'un lieu ;
- demande sa latitude puis sa longitude en DMS (degrés, minutes, secondes) ;
- convertit les deux angles en DD ;

- crée une carte centrée sur ce lieu, avec un marqueur portant son nom, et l'enregistre.

Fichier : NOM_Prenom_ex38.py

> **Et ensuite ?**
>
> Tu peux maintenant reprendre le **TP 3 de Localisation** : ses programmes Folium utilisent exactement ces trois nouveautés.

Partie 4 — Décider : if, elif, else

Des programmes qui choisissent ce qu'ils font

Jusqu'ici, tes programmes exécutent toujours les mêmes lignes, dans le même ordre. Dans cette partie, ils vont **choisir** : faire une chose ou une autre selon une condition. À la fin, tu auras complété ton convertisseur GPS avec le nord, le sud, l'est et l'ouest.

Cours · Comparer

Une **comparaison** est une question dont la réponse est `True` (vrai) ou `False` (faux).

Écriture	Signification
`a == b`	a est égal à b
`a != b`	a est différent de b
`a < b` `a <= b`	a est strictement inférieur à b ; inférieur ou égal à b
`a > b` `a >= b`	a est strictement supérieur à b ; supérieur ou égal à b

console

```
>>> 7 > 5
True
>>> 10 % 2 == 0
True
>>> 3 + 4 == 8
False
```

> **Vigilance : = ou == ?**
>
> `a = 5` **range** la valeur 5 dans la variable a. `a == 5` **teste** si a vaut 5.
>
> Pour tester une égalité, il faut toujours **deux** signes égal.

Cours · if, else

parite.py

```
n = int(input("Un entier : "))
if n % 2 == 0:
    print(n, "est pair")
else:
    print(n, "est impair")
print("Fin du programme")
```

- Après `if` et après `else`, on met des **deux-points** `:`.
- Les lignes qui dépendent de la condition sont **décalées** vers la droite (indentation, 4 espaces ou une tabulation).
- La dernière ligne n'est pas décalée : elle est exécutée dans tous les cas.

> **À retenir**
>
> `if condition :` exécute son bloc **seulement si** la condition est vraie.
>
> `else :` exécute son bloc dans **tous les autres cas**.

Cours · Plus de deux cas : elif

eau.py

```
temperature = float(input("Température de l'eau en °C : "))
if temperature <= 0:
    print("Glace")
elif temperature < 100:
    print("Eau liquide")
else:
    print("Vapeur")
```

Python teste les conditions **dans l'ordre**, s'arrête à la première qui est vraie, et ignore les suivantes.

Vigilance : l'ordre des conditions compte

Avec `temperature = -5`, la première condition est vraie : on affiche « Glace » et on ne teste pas la suite.

Si on avait écrit d'abord `if temperature < 100`, « Eau liquide » s'afficherait pour -5 °C !

Cours · Combiner : and, or, not

Mot	Sens	Exemple
and	les deux conditions sont vraies	`n % 3 == 0 and n % 5 == 0`
or	au moins une condition est vraie	`lettre == "S" or lettre == "O"`
not	inverse la condition	`not (n > 10)`

Un `if` peut aussi être placé **dans** un autre `if` : les lignes du bloc intérieur sont décalées une deuxième fois.

tarif.py

```
age = int(input("Ton âge : "))
if age >= 11 and age <= 15:
    print("Tarif collégien")
    if age == 15:
        print("Plus qu'un an avant le lycée !")
```

Cours · Un peu de hasard

de.py

```
from random import *

de = randint(1, 6)      # entier au hasard entre 1 et 6 (inclus)
print("Le dé donne", de)
```

`randint(a, b)` renvoie un entier tiré au hasard entre a et b, **bornes comprises**. Chaque exécution peut donner un résultat différent.

Exercices · Décisions à prendre

Exercice 39 — Vrai ou faux ? 🌶

Sans utiliser l'ordinateur, prévois le résultat (`True` ou `False`) de chaque ligne :

console

```
8 >= 8
5 != 5
17 % 5 == 2
4 + 3 * 2 == 14
2026 % 4 == 0
(3 > 1) and (2 > 5)
(3 > 1) or (2 > 5)
not (3 > 1)
```

Vérifie ensuite chaque ligne dans la console.

Fichier : NOM_Prenom_ex39.py

Exercice 40 — Multiple de 7 ? 🌶

Recopie `parite.py`, teste-le avec plusieurs entiers, puis modifie-le pour qu'il dise si l'entier saisi est **un multiple de 7** ou **n'en est pas un**.

Fichier : NOM_Prenom_ex40.py

Exercice 41 — Multiple de 3, de 5, des deux ? 🌶🌶

Écris un programme qui demande un entier et affiche :

- `multiple de 3 et de 5` si l'entier est multiple des deux ;
- `multiple de 3` s'il l'est de 3 seulement ;
- `multiple de 5` s'il l'est de 5 seulement ;
- `ni de 3 ni de 5` sinon.

Teste avec 30, 9, 10 et 7.

Fichier : NOM_Prenom_ex41.py

Exercice 42 — La mention au bac 🌶🌶

Écris un programme qui demande une note sur 20 (éventuellement décimale) et affiche le résultat d'un candidat au baccalauréat :

Note	Résultat
16 ou plus	Mention très bien
de 14 (inclus) à 16 (exclu)	Mention bien
de 12 (inclus) à 14 (exclu)	Mention assez bien
de 10 (inclus) à 12 (exclu)	Admis sans mention
moins de 10	Non admis

Fichier : NOM_Prenom_ex42.py

Exercice 45 — Année bissextile 🌶🌶🌶

Une année est **bissextile** si elle est divisible par 4 **sauf** si elle est divisible par 100, à moins qu'elle ne soit divisible par 400.

Écris un programme qui demande une année et dit si elle est bissextile.

Teste avec 2024 (oui), 1900 (non), 2000 (oui) et 2026 (non).

Fichier : NOM_Prenom_ex45.py

Exercice 47 — Le convertisseur avec N, S, E, O 🌶🌶🌶

Dans la partie 3, le convertisseur DMS vers DD ne fonctionnait que pour le nord et l'est. En DD, les latitudes **sud** et les longitudes **ouest** sont **négatives**.

Complète ton programme de l'exercice 31 : après les degrés, minutes et secondes, il demande la lettre N, S, E ou O, puis affiche l'angle en DD, arrondi à 6 chiffres, avec le bon signe.

Test : Bordeaux, longitude 0° 34′ 44″ O, doit donner −0.578889 ; sa latitude 44° 50′ 14″ N doit donner 44.837222.

Fichier : NOM_Prenom_ex47.py

À retenir · Synthèse

Ce que je sais faire

- Écrire une condition avec ==, !=, <, <=, >, >=, et la combiner avec `and`, `or`, `not`.
- Choisir entre plusieurs cas avec `if`, `elif`, `else`, en soignant les `:` et l'indentation.
- Tirer un entier au hasard avec `randint`.
- Gérer les signes des coordonnées GPS avec N, S, E, O.

Partie 5 — Répéter : la boucle for

Compter, cumuler, parcourir : calculer sans se fatiguer

Tu connais déjà `for i in range(4):` pour répéter des instructions de la tortue. Ici, la variable `i` va devenir utile : elle **change à chaque tour** et permet de faire des calculs.

Cours · La boucle for et range boucle.py

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

Ce programme affiche `0`, `1`, `2`, `3`, `4`, chacun sur une ligne. La variable `i` prend successivement toutes les valeurs de `range(5)`.

Écriture	Valeurs de la variable
`range(5)`	0, 1, 2, 3, 4
`range(2, 6)`	2, 3, 4, 5
`range(1, 10, 3)`	1, 4, 7 (on avance de 3 en 3)
`range(10, 0, -2)`	10, 8, 6, 4, 2 (on recule de 2 en 2)

Vigilance : la borne de fin est exclue

`range(2, 6)` s'arrête **avant** 6. Pour aller jusqu'à 10 inclus, on écrit `range(1, 11)`.

Sans borne de départ, `range` commence à **0**.

Cours · Cumuler : l'accumulateur

Pour additionner les entiers de 1 à 5, on prépare une variable qui vaut 0, puis on lui ajoute chaque valeur de `i`.

somme.py

```
somme = 0
for i in range(1, 6):
    somme = somme + i
print(somme)
```

Tour	Valeur de `i`	Valeur de `somme` à la fin du tour
avant	—	0
1	1	1
2	2	3
3	3	6
4	4	10
5	5	15

Vigilance

La ligne `somme = 0` est **avant** la boucle : sinon la somme repartirait de 0 à chaque tour.

`print(somme)` est **après** la boucle, donc **non décalé** : on affiche une seule fois, à la fin.

Cours · Compter : le compteur

Pour compter les éléments qui vérifient une propriété, on place un `if` **dans** la boucle.

compteur.py

```
compteur = 0
for n in range(1, 101):
    if n % 3 == 0:
        compteur = compteur + 1
print("Il y a", compteur, "multiples de 3 entre 1 et 100")
```

À retenir

Accumulateur : `total = 0` avant la boucle, puis `total = total + ...` dans la boucle.

Compteur : c'est un accumulateur qui augmente de 1 chaque fois qu'une condition est vraie.

Le nombre de tours est connu à l'avance : on utilise `for`.

Exercices · Boucler pour calculer

Exercice 50 — Prédire ce qui s'affiche

Sans l'ordinateur, écris ce que chaque programme affiche, puis vérifie en les exécutant.

a.py

```
for i in range(4):
    print(i)
```

b.py

```
for i in range(3, 7):
    print(i * 2)
```

c.py

```
for i in range(10, 0, -3):
    print(i)
```

d.py

```
s = 0
for i in range(1, 5):
    s = s + i * i
print(s)
```

Fichier : NOM_Prenom_ex50.py

Exercice 51 — Table de multiplication

Écris un programme qui demande un entier n et affiche sa table de multiplication de 1 à 10, sous la forme `7 x 3 = 21`. Fichier : NOM_Prenom_ex51.py

Exercice 52 — La somme de 1 à n 🌶🌶

1. Écris un programme qui demande un entier n et affiche la somme $1+2+\ldots+n$. Quelle est la somme pour $n=100$?
2. Le mathématicien Gauss a remarqué que cette somme vaut $\frac{n(n+1)}{2}$. Ajoute une ligne qui affiche cette valeur, et compare pour plusieurs n.

Fichier : NOM_Prenom_ex52.py

Exercice 53 — La factorielle 🌶🌶

La **factorielle** de n, notée $n!$, est le produit $1\times2\times3\times\ldots\times n$. Par exemple $5!=120$.

1. Écris un programme qui demande n et affiche $n!$.
2. À partir de quelle valeur de n la factorielle dépasse-t-elle un million ?

Fichier : NOM_Prenom_ex53.py

Exercice 55 — Les diviseurs d'un entier 🌶🌶🌶

Écris un programme qui demande un entier n et affiche tous ses diviseurs, puis leur nombre.

Teste avec 36, puis avec 97.

Fichier : NOM_Prenom_ex55.py

Exercice 56 — Premier ou pas ? 🌶🌶🌶

Un entier est **premier** s'il a exactement **deux diviseurs** : 1 et lui-même.

À partir de l'exercice 55, écris un programme qui dit si l'entier saisi est premier.

Teste avec 97, 91, 2 et 1.

Fichier : NOM_Prenom_ex56.py

Exercice 57 — Un tableau de valeurs 🌶🌶

On considère la fonction $f(x)=x^2-3x+2$.

1. Écris un programme qui affiche, pour tous les entiers x de -3 à 5, la ligne `x = -3 : f(x) = 20`.
2. Pour quelles valeurs de x le programme affiche-t-il 0 ? Vérifie par le calcul.

Fichier : NOM_Prenom_ex57.py

Exercice 58 — Un placement à intérêts composés 🌶🌶🌶

On place 1000 € sur un compte qui rapporte **2 % par an** : chaque année, la somme est multipliée par 1,02.

1. Écris un programme qui demande un nombre d'années et affiche la somme obtenue **chaque année**, arrondie au centime.
2. Quelle somme a-t-on après 10 ans ?
3. Avec un taux de 5 %, que faut-il changer ? Quelle somme a-t-on après 10 ans ?

Fichier : NOM_Prenom_ex58.py

À retenir · Synthèse

Ce que je sais faire

- Écrire `for i in range(...)` avec un, deux ou trois paramètres, et savoir quelles valeurs prend `i`.
- Cumuler une somme ou un produit avec un accumulateur.
- Compter avec un `if` dans une boucle.
- Imbriquer deux boucles.

Partie 6 — Répéter : la boucle while

Répéter tant qu'une condition est vraie

Parfois, on ne sait pas à l'avance combien de fois il faudra répéter : il faut recommencer **tant que** quelque chose n'est pas réalisé. C'est le rôle de la boucle `while`.

Cours · La boucle while

compte_a_rebours.py

```
n = 10
while n > 0:
    print(n)
    n = n - 1
print("Décollage !")
```

Tant que la condition `n > 0` est vraie, Python exécute le bloc décalé, puis revient tester la condition. Dès qu'elle est fausse, il passe à la suite.

Les trois éléments d'un `while`	Dans l'exemple
1. **Initialiser** la variable testée	`n = 10`
2. Une **condition** de poursuite	`while n > 0 :`
3. Une **modification** de la variable dans le bloc	`n = n - 1`

> **Vigilance : la boucle infinie**
>
> Si on oublie la modification (`n = n - 1`), la condition reste vraie pour toujours : le programme ne s'arrête jamais.
>
> Dans EduPython, arrête-le avec **Ctrl + F2** (ou ferme la fenêtre d'exécution).

Cours · Répéter jusqu'à obtenir ce qu'on veut

mot_de_passe.py

```
mot_de_passe = ""
while mot_de_passe != "python":
    mot_de_passe = input("Mot de passe : ")
print("Accès autorisé")
```

On ne sait pas combien d'essais fera l'utilisateur : un `for` ne conviendrait pas.

Je choisis…	quand…	Exemple
`for`	je connais le **nombre de répétitions**	afficher une table de multiplication
`while`	je répète **jusqu'à ce qu'une condition** soit réalisée	attendre une bonne réponse, atteindre un seuil

Exercices · Boucler jusqu'au bout

Exercice 62 — Compte à rebours

Modifie `compte_a_rebours.py` pour qu'il **demande** le nombre de départ, puis affiche le compte à rebours jusqu'à 0 et « Décollage ! ».

Fichier : NOM_Prenom_ex62.py

Exercice 63 — Saisie contrôlée

Écris un programme qui demande l'âge de l'utilisateur et **recommence tant que la réponse est impossible** (négative ou supérieure à 120). Il termine par « Merci, tu as … ans. ».

Fichier : NOM_Prenom_ex63.py

Exercice 64 — Le nombre mystère

L'ordinateur choisit un entier au hasard entre 1 et 100 avec `randint`. Le joueur propose des nombres ; l'ordinateur répond « Trop petit » ou « Trop grand » jusqu'à ce que le joueur trouve, puis affiche le **nombre d'essais**.

1. Programme ce jeu.
2. Existe-t-il une stratégie qui permet de gagner en 7 essais au maximum ? Laquelle ?

Fichier : NOM_Prenom_ex64.py

Exercice 66 — Le seuil du placement

À l'exercice 58, 1000 € placés à 2 % par an.

1. Écris un programme qui calcule au bout de combien d'années la somme **dépasse 2000 €** (elle a doublé).
2. Même question avec un taux de 5 %.

Fichier : NOM_Prenom_ex66.py

Exercice 68 — L'algorithme d'Euclide 🌶🌶🌶

Le **PGCD** de deux entiers est leur plus grand diviseur commun. L'algorithme d'Euclide le calcule ainsi : on remplace le couple $(a\,;b)$ par $(b\,;r)$, où r est le reste de la division de a par b, **jusqu'à ce que le reste soit nul**. Le PGCD est le dernier diviseur non nul.

1. À la main, calcule le PGCD de 1071 et 462.
2. Programme l'algorithme : le programme demande a et b et affiche leur PGCD.
3. Teste avec (84 ; 36), puis (17 ; 5). Que signifie un PGCD égal à 1 ?

Fichier : NOM_Prenom_ex68.py

À retenir · Synthèse

Ce que je sais faire

- Écrire une boucle `while` avec ses trois éléments : initialisation, condition, modification.
- Choisir entre `for` (nombre de tours connu) et `while` (on répète jusqu'à réalisation d'une condition).
- Repérer et arrêter une boucle infinie.
- Programmer des algorithmes classiques : PGCD d'Euclide, suite de Syracuse, conversion en binaire, dichotomie.

Partie 7 — Écrire ses propres fonctions

Donner un nom à un calcul ou à un dessin pour le réutiliser

Tu utilises déjà des fonctions écrites par d'autres : `print`, `round`, `sqrt`, `forward`… Dans cette partie, tu vas **écrire les tiennes**, pour ne plus recopier deux fois le même calcul.

Cours · Définir et appeler une fonction

fonction.py

```
def carre(x):
    return x * x

print(carre(5))          # 25
y = carre(3) + 1         # y vaut 10
```

Élément	Rôle
`def carre(x):`	**définit** la fonction `carre` avec un **paramètre** x (deux-points obligatoires)
le bloc décalé	les instructions de la fonction
`return x * x`	**renvoie** le résultat à l'endroit où la fonction a été appelée
`carre(5)`	**appelle** la fonction en donnant la valeur 5 au paramètre x

Vigilance

Définir une fonction ne l'exécute pas : il faut ensuite l'**appeler**.

La fonction doit être définie **avant** l'endroit où on l'appelle.

Cours · return ou print ?

return_print.py

```
def affiche_carre(x):
    print(x * x)

def renvoie_carre(x):
    return x * x
```

- `affiche_carre(4)` **affiche** 16 mais ne renvoie rien : on ne peut pas s'en servir dans un calcul.
- `renvoie_carre(4)` **renvoie** 16 : on peut écrire `2 * renvoie_carre(4)` et obtenir 32.

À retenir

Une fonction de calcul se termine par `return` : c'est ce qui permet de **réutiliser** le résultat.

Une fonction peut avoir plusieurs paramètres : `def aire(l, h):`. Elle peut aussi renvoyer plusieurs valeurs : `return a, b`.

Les variables créées dans une fonction n'existent **que dans** la fonction.

Cours · Des fonctions qui dessinent

carres.py

```
from turtle import *

def carre(c):
    for i in range(4):
        forward(c)
        left(90)

carre(50)
carre(100)
done()
```

Cette fonction ne renvoie rien : elle **agit** (elle dessine). Comme elle est écrite une seule fois, on peut la modifier à un seul endroit.

Exercices · Écrire des fonctions

Exercice 72 — Lire une fonction

lire.py

```
def f(x):
    return 2 * x + 1

def g(a, b):
    return a * b - 3
```

1. Sans l'ordinateur, calcule `f(4)`, `f(0)`, `g(2, 5)` et `f(g(1, 4))`.
2. Vérifie dans la console après avoir exécuté le programme.

Fichier : NOM_Prenom_ex72.py

Exercice 73 — Mes premières fonctions

1. Écris une fonction `cube(x)` qui renvoie x^3.
2. Écris une fonction `perimetre_rectangle(l, h)` qui renvoie le périmètre d'un rectangle de longueur `l` et de largeur `h`.
3. Écris une fonction `aire_disque(r)` qui renvoie l'aire d'un disque de rayon `r` (utilise `pi` après `from math import *`).

Teste-les dans la console : `cube(3)` doit donner 27.

Fichier : NOM_Prenom_ex73.py

Exercice 74 — print ou return ?

deux_fonctions.py

```
def f_print(x):
    print(x + 1)

def f_return(x):
    return x + 1
```

1. Que fait `2 * f_return(5)` ? Et `2 * f_print(5)` ?
2. Exécute les deux dans la console pour vérifier, puis explique la différence.

Fichier : NOM_Prenom_ex74.py

Exercice 75 — Ma fonction f

On reprend $f(x) = x^2 - 3x + 2$ de l'exercice 57.

1. Écris une fonction Python `f(x)` qui renvoie $f(x)$.
2. Calcule `f(-1)`, `f(0)`, `f(2.5)` et `f(10)`.
3. Refais le tableau de valeurs de -3 à 5 avec une boucle qui **appelle** ta fonction.

Fichier : NOM_Prenom_ex75.py

Exercice 76 — Des fonctions qui répondent par vrai ou faux

1. Écris une fonction `est_pair(n)` qui renvoie `True` si `n` est pair et `False` sinon.
2. Écris une fonction `est_premier(n)` qui renvoie `True` si `n` est premier (voir l'exercice 56).
3. Avec une boucle et `est_premier`, affiche les nombres premiers inférieurs à 50.

Fichier : NOM_Prenom_ex76.py

Exercice 78 — Le convertisseur GPS en fonctions 🌶🌶🌶

Dans l'exercice 38, on écrivait deux fois le même calcul. Range-le dans des fonctions.

1. Écris `dms_vers_dd(d, m, s)` qui renvoie l'angle en degrés décimaux.
2. Écris `dd_vers_dms(dd)` qui renvoie **trois valeurs** : `d, m, s` (avec `return d, m, s`). On récupère les trois valeurs avec `d, m, s = dd_vers_dms(44.1646098)`.
3. Écris un programme principal qui utilise les deux fonctions, et vérifie les valeurs de la partie 3.

Fichier : NOM_Prenom_ex78.py

À retenir · Synthèse

Ce que je sais faire

- Définir une fonction avec `def`, des paramètres et `return`, puis l'appeler.
- Distinguer une fonction qui **renvoie** un résultat (`return`) d'une fonction qui **affiche** ou **dessine**.
- Renvoyer plusieurs valeurs avec `return a, b`.
- Ranger un calcul répété (conversion GPS, distance, PGCD) dans une fonction réutilisable.

Partie 8 — Programm'Art

Projet de fin de partie : créer une œuvre avec du code

Dernière étape : tu vas utiliser **tout** ce que tu as appris (fonctions, boucles, conditions, hasard) pour créer une œuvre graphique à la manière d'un peintre abstrait. Les trois premiers exercices te préparent ; le quatrième est un projet libre.

Cours · De nouveaux outils pour la tortue

Commande	Effet
`setheading(0)`	oriente la tortue dans une direction donnée : 0 vers la droite, 90 vers le haut, 180 vers la gauche, 270 vers le bas
`pensize(5)`	règle l'épaisseur du trait
`circle(r)`	dessine un cercle de rayon `r`
`choice(liste)`	(après `from random import *`) renvoie un élément de la liste choisi au hasard

choix.py

```
from random import *

couleurs = ["red", "blue", "yellow", "white"]     # une liste de couleurs
print(choice(couleurs))                            # une couleur au hasard
```

Une fonction qui dessine un rectangle plein de couleur, à un endroit précis, sera l'outil de base de ton œuvre : rectangle.py

```
from turtle import *

def rectangle(x, y, l, h, couleur):
    penup()
    goto(x, y)          # coin inférieur gauche
    setheading(0)
    pendown()
    fillcolor(couleur)
    begin_fill()
    for i in range(2):
        forward(l)
        left(90)
        forward(h)
        left(90)
    end_fill()
```

Exercice 82 — La fonction rectangle 🌶🌶

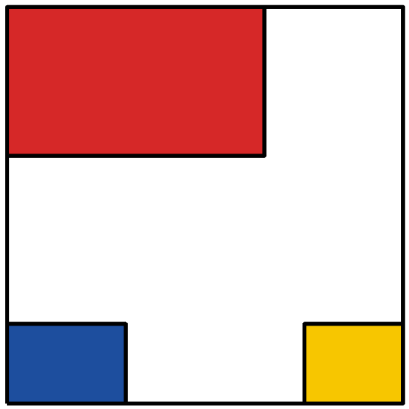
1. Recopie la fonction `rectangle` du cours, puis ajoute à la fin du programme trois appels qui dessinent trois rectangles de couleurs différentes à trois endroits différents.

2. Ajoute `pensize(4)` au début du programme : que change-t-il ? Pourquoi est-ce joli pour une œuvre à la manière de Mondrian ?

Fichier : NOM_Prenom_ex82.py

Exercice 83 — Composition à la manière de Mondrian 🌶🌶🌶

Reproduis cette composition dans une fenêtre de 400 sur 400 : un grand cadre blanc dans lequel se trouvent un rectangle rouge, un rectangle bleu et un rectangle jaune.



Composition à reproduire (coordonnées : cadre de (−200 ; −200) à (200 ; 200)).

Le cadre est le rectangle de coin inférieur gauche $(-200\,;-200)$, de largeur 400 et de hauteur 400.

Fichier : NOM_Prenom_ex83.py

Exercice 85 — Ton œuvre 🌶🌶🌶🌶

Invente ta propre œuvre graphique. Tu peux t'inspirer de Mondrian, mais aussi de Vasarely (grilles et illusions d'optique), de Kandinsky (cercles et formes géométriques), de Sonia Delaunay (disques de couleurs), ou d'un style de ton choix.

Ton programme doit contenir :

- **au moins une fonction** avec des paramètres (par exemple `rectangle`, `cercle_couleur`, ou une nouvelle fonction inventée) ;
- **au moins une boucle** (`for` ou `while`), éventuellement imbriquée ;
- **au moins une condition** (`if`) **ou** du **hasard** (`randint`, `choice`) ;
- **des commentaires** qui expliquent ce que font les parties importantes.

Dépose ton fichier `NOM_Prenom_art.py` sur Pronote, avec une capture d'écran de ton œuvre.

Grille de relecture (à cocher avant de rendre)

- Mon programme s'exécute sans erreur.
- Il contient une fonction avec paramètres, une boucle, et une condition ou du hasard.
- Il est commenté.
- Mon œuvre est soignée : couleurs choisies, composition réfléchie.
- J'ai testé mon programme au moins deux fois et je sais expliquer comment il marche.

Fichier : NOM_Prenom_art.py + capture d'écran

À retenir · Bilan du cours de Python

Ce que tu sais faire maintenant

- Utiliser Python comme une calculatrice, avec des variables, `//`, `%`, `**` et le module `math`.
- Dialoguer avec `print` et `input`, convertir avec `int` et `float`.
- Décider avec `if`, `elif`, `else` et répéter avec `for` et `while`.
- Écrire tes propres **fonctions**, dessiner avec la tortue, cartographier avec Folium.
- Programmer des algorithmes de mathématiques : PGCD, nombres premiers, conversions, dichotomie.