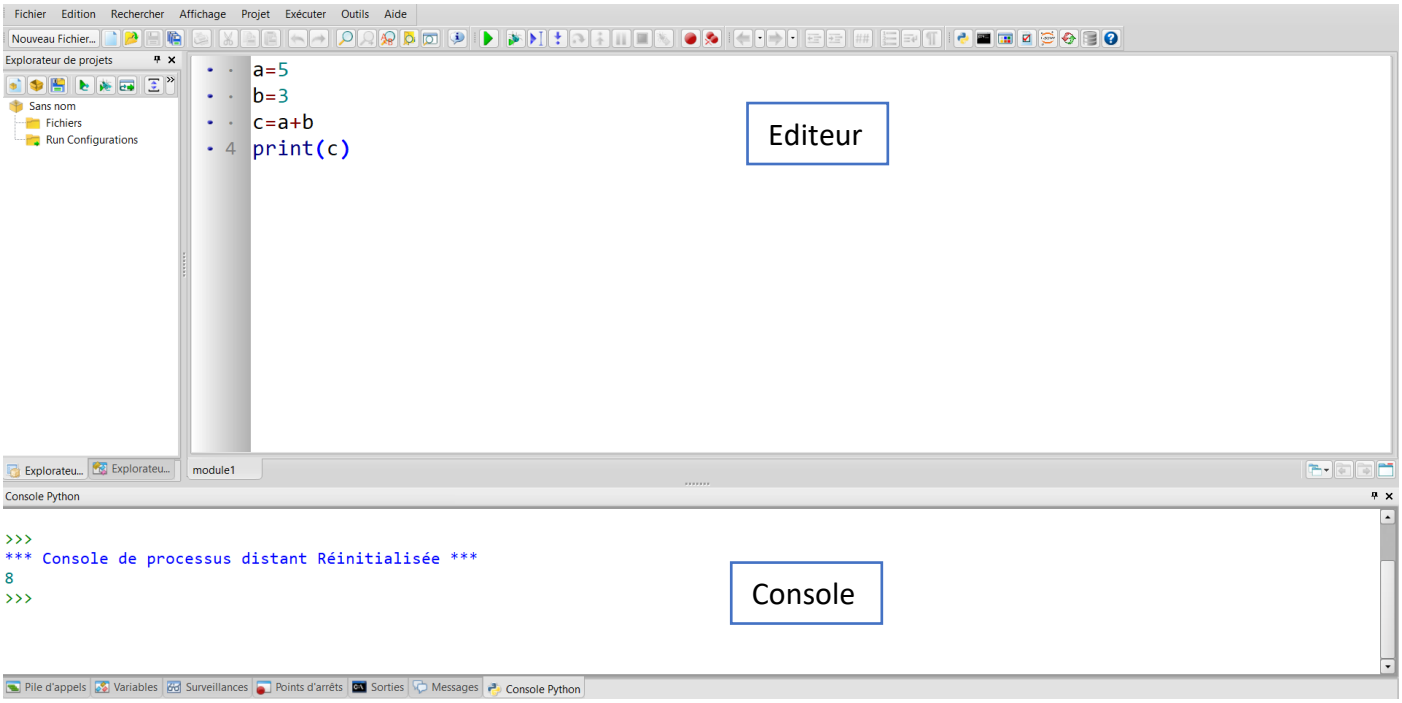
Introduction

LOGICIEL EDUPYTHON

Présentation de l'interface du logiciel EduPython :
Les deux parties importantes de l'interface sont :
- L'éditeur où l'on peut écrire les programmes
- La console où l'on exécute le programme écrit dans l'éditeur, on peut aussi écrire des commandes dans cette fenêtre



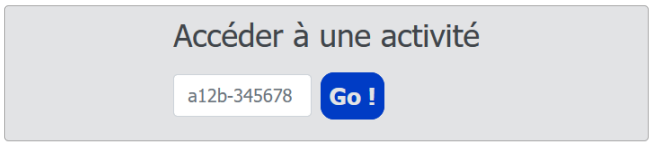
On peut exécuter un programme en cliquant sur l'icône 

Introduction

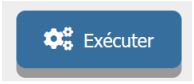
CAPYTALE SUR Lycée Connecté

Connexion et présentation de l'interface de CAPYTALE :

Pour accéder à l'interface Python :
- Se connecter sur l'ENT Lycée Connecté
- Dans les applications, cliquer sur l'icône 

- Saisir le code de l'activité 

Présentation de l'interface :
Les deux parties importantes de l'interface sont :
- Le script où l'on peut écrire les programmes
- La console où l'on exécute le programme écrit dans l'éditeur, on peut aussi écrire des commandes dans cette fenêtre

On peut exécuter un programme en cliquant sur 

On enregistre le travail en cliquant sur 

Introduction

Menu Python sur Calculatrice CASIO graph 35+ EII

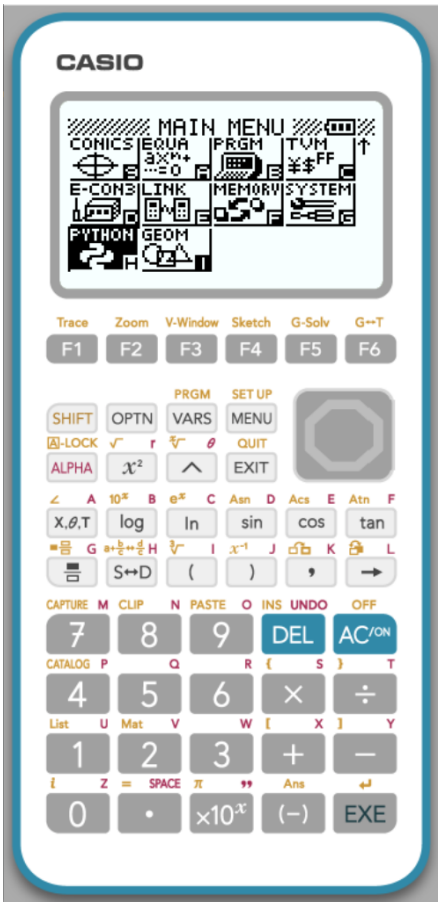
Présentation du menu Python de la calculatrice CASIO graph 35+ EII

- Accès au menu Python

  -Appuyer sur la touche MENU.

  -Sélectionner le menu PYTHON à l'aide des flèches.

  -Valider par la touche EXE.



- Création d'un programme

  -Créer un nouveau programme en sélectionnant F3 **{NEW}**.

  -Entrer le nom du script et valider avec EXE.

- Exécution du programme

  -Une fois le script rédigé, il est possible d'exécuter le programme en se positionnant sur le fichier concerné et en sélectionnant F1 **{RUN}**.

TP 1 : Variable

Activité :

1) a) Ecrire le programme 1 ci-contre dans l'éditeur puis l'exécuter.
   b) Que permet le symbole = ? Que permet l'instruction `print` ?

Programme 1

```
a=5
print(a)
```

2) a) Ecrire le programme 2 ci-contre dans l'éditeur puis l'exécuter.
   b) Quelle est la différence entre ces deux programmes ?

Programme 2

```
a=5
print("a")
```

3) Il existe différents types de variables dans le langage Python.
   On peut déterminer chaque type grâce à l'instruction `print(type(a))`.
   a) Ecrire le programme 3 ci-contre dans l'éditeur puis l'exécuter.
   b) Quels sont les 4 types obtenus ?

Programme 3

```
a=5
b=5.0
c="bonjour"
d=a==5
print(type(a))
print(type(b))
print(type(c))
print(type(d))
```

A retenir :

Qu'est-ce qu'une variable ?

Une **variable** sert à stocker une valeur qui peut être modifiée au cours du temps par l'algorithme.
(C'est comme une sorte de boîte dans laquelle on ne peut stocker qu'une seule valeur à chaque fois)
Chaque variable a un nom.

L'instruction `print(a)` permet d'afficher la variable a dans la console.

En langage Python, on affecte une valeur à une variable en utilisant le signe = .
Exemple : Pour affecter la valeur 2 à la variable a, on utilise l'instruction : a=2

Il existe différents types de variables, en voici quelques exemples :

Type	Notation Python	Exemple
Nombre entier (integer)	Int	a=2
Nombre décimal (flottant)	float	a=3.6
Chaîne de caractères (string)	str	a="bonjour"
Booléen (boolean)	bool	a=True

Pour connaître le type d'une variable a, on peut utiliser l'instruction `type(a)`.

Quelques opérations sur les variables

Opérations	Symboles	Exemples
Somme	+	3+5
Différence	-	6-2
Produit	*	5*4
Puissance	**	2**4
Quotient	/	9/4
Quotient de la division euclidienne	//	9//4
Reste de la division euclidienne	%	9%4

Pour déterminer la longueur d'une chaîne de caractères a, on utilise l'instruction `len(a)` en Python.

Exercice 1

On donne le programme suivant :

```
a=2.6
b=True
c=3
d="Salut!"
```

1) De quels types sont les variables a, b, c et d ?

. . . . . .

. . . . . .

. . . . . .

. . . . . .

2) Quelles instructions peut-on ajouter à la fin du programme pour afficher les types des différentes variables ?

. . . . . .

. . . . . .

3) Tester le programme en affichant le type de chaque variable.

Exercice 2

On écrit les instructions suivantes dans la console :

```
>>> a=3+5
>>> b=6-2
>>> c=5*4
>>> d=2**4
>>> e=9/4
>>> f=9//4
>>> g=9%4
```

1) Donner la valeur de chaque variable.

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

2) Tester ces instructions dans la console pour vérification.

Exercice 3

1) Que vaut a à la fin de l'exécution du programme suivant ?

```
a=2
a=a+a
a=a+a
a=a+a
```

. . . . . .

. . . . . .

2) Tester ces instructions dans la console pour vérification.

Exercice 4

1) Quelle est la valeur des variables c et d dans le programme suivant ?

```
a="Mai"
b="Son"
c=a+b
d=len(c)
```

. . . . . .

. . . . . .

. . . . . .

. . . . . .

2) Tester ces instructions dans la console pour vérification.

Exercice 5

1) Taper les instructions suivantes dans la console Python puis écrire le résultat affiché.

```
>>> a,b=3,5
>>> a+b
```

. . . . . .

2) Que permet la virgule ?

. . . . . .

. . . . . .

TP 2 : Instruction conditionnelle

Activité :

1) Ecrire le programme ci-contre.
2) Exécuter ce programme avec comme valeur de x : `5.3`, `-1.4` et `0`.
   Expliquer les différents affichages.
   Que signifient `if`, `elif` et `else` ?
3) Que signifie le symbole == ?

Programme

```
x=float(input("saisir un nombre:"))
if x < 0:
    print("negatif")
elif x == 0:
    print("zero")
else:
    print("positif")
```

A retenir :

Structures conditionnelles

Une instruction conditionnelle est composée d'un *test* puis d'un *bloc d'instructions*.
L'instruction n'est exécutée que si la condition est réalisée.

Il existe différentes structures conditionnelles en fonction du nombre de conditions que l'on souhaite tester.
Dans tous les cas, la première condition est introduite par le mot `if`.

Structure conditionnelle « if » (Si …

```
if condition:
    instruction
```

Alors) :

Structure conditionnelle « if … else » (Si …

```
if condition:
    instruction1
else:
    instruction2
```

Sinon) :

Structure conditionnelle « if … elif … else » (Si … Sinon Si … Sinon) : (on peut utiliser plusieurs « elif »)

```
if condition1:
    instruction1
elif condition2:
    instruction2
else:
    instruction3
```

Remarques : Il ne faut pas oublier les deux points après la condition et il faut respecter l'indentation (le décalage vers la droite) au niveau des instructions.

Conditions

Pour tester une condition, on peut être amener à utiliser des opérateurs de comparaison ou des opérateurs logiques.

Opérateurs de comparaison (Python)	Signification
==	égal à
!=	différent de
< ou <=	inférieur (ou égal) à
> ou >=	supérieur (ou égal) à

Opérateurs logiques (Python)	Signification
and	Et
or	Ou
not	Non

Exercice 1

On donne le programme suivant :

```
a=2
b=5
if a+b>6:
    a=4
else:
    b=4
```

Donner les valeurs de a et b en fin de programme.

. . . . . .

. . . . . .

Exercice 2

On donne le programme suivant :

```
a=4
if a==5:
    print("gagné!")
else:
    print("perdu!")
```

1) Quelle est la condition dans ce programme ?

. . . . . .

. . . . . .

2) Que va afficher ce programme ?

. . . . . .

. . . . . .

Exercice 3

On donne le programme suivant :

```
x=float(input("saisir un nombre:"))
if x<=2:
    y=2*x
else:
    y=3*x
print(y)
```

1) Qu'obtient-on si $x = 5$?

. . . . . .

2) Qu'obtient-on si $x = 1{,}5$?

. . . . . .

3) Tester ce programme pour vérification.

Exercice 4

On donne le programme suivant :

```
x=float(input("saisir un nombre x:"))
y=float(input("saisir un nombre y:"))
if x*y>0:
    print("le résultat est positif")
elif x*y==0:
    print("le résultat est nul")
else:
    print("le résultat est négatif")
```

1) Qu'obtient-on si $x = 2$ et $y = 5$?

. . . . . .

2) Qu'obtient-on si $x = 3$ et $y = -2$?

. . . . . .

3) Quelles conditions sur x et y permettent d'obtenir « le résultat est positif » ?

. . . . . .

. . . . . .

4) Tester ce programme.

Exercice 5

On donne le programme suivant :

```
n=int(input("saisir un nombre entier:"))
if n%2==0:
    y=n+5
else:
    y=n+10
print(y)
```

1) Que teste l'instruction « `if n%2==0` » ?

. . . . . .

. . . . . .

2) Qu'obtient-on si $n = 6$? si $n = 7$?

. . . . . .

. . . . . .

3) Tester ce programme.

TP 3 : Boucles

Activité :

1) **Boucle bornée**
   a) Que signifie le mot anglais « for » ?
   b) Ecrire le programme 1 ci-contre dans l'éditeur puis l'exécuter. Qu'affiche ce programme ?
   c) Expliquer l'instruction `for i in range(1,4).`
   d) Ecrire le programme 2 ci-contre dans l'éditeur puis l'exécuter. Qu'affiche ce programme ?
   e) Quelle est la différence avec le programme 1 ?

Programme 1

```
for i in range(1,4):
    print(i)
```

Programme 2

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

2) **Boucle non bornée**
   a) Que signifie le mot anglais « while » ?
   b) Ecrire le programme 3 ci-contre dans l'éditeur puis l'exécuter. Qu'affiche ce programme ?
   c) Pourquoi le programme s'arrête-t-il ?

Programme 3

```
a=5
while a < 10:
    a=a+1
    print(a)
```

A retenir :

Une boucle est une suite d'instructions que l'on répète un certain nombre de fois.

Boucle bornée `for`

Lorsque le nombre de répétitions est connu à l'avance, on utilise une boucle bornée `for`.

En langage Python, la syntaxe d'une boucle `for` est la suivante :

```
for variable in range():
    instructions
```

La fonction `range()` permet d'énumérer le nombre de passages dans la boucle bornée.
Elle peut être appelée de plusieurs façons :

`for k in … :`	k prend successivement les valeurs entières
`range(6)`	de 0 à 5 donc 0, 1, 2, 3, 4, 5 (6 répétitions à partir de 0)
`range(1,6)`	de 1 à 5 donc 1, 2, 3, 4, 5
`range(6,15,2)`	de 6 à 14 par pas de 2 donc 6, 8, 10, 12, 14

Boucle non bornée `while`

Lorsque le nombre de répétitions n'est pas connu à l'avance, on utilise une boucle non bornée `while`.
La boucle est parcourue tant que la condition est vérifiée.

En langage Python, la syntaxe d'une boucle `while` est la suivante :

```
while condition:
    instructions
```

Exercice 1

On donne le programme suivant :

```
for i in range(5):
    print(2+i)
```

1) Combien de tours de boucle sont réalisés dans ce programme ?

. . . . . .

2) Quelles valeurs seront affichées après exécution du programme ?

. . . . . .

. . . . . .

Exercice 2

On donne le programme suivant :

```
x=1
while x<10:
    x=x*2
    print(x)
```

1) Quand la boucle `while` va-t-elle s'arrêter ?

. . . . . .

2) Quelles valeurs seront affichées après exécution du programme ?

. . . . . .

. . . . . .

3) Tester le programme pour vérification.

Exercice 3

On donne le programme suivant :

```
a=2
a=a*2
a=a*3
a=a*4
a=a*5
```

1) Ecrire un programme avec une boucle pour remplacer ce programme.

. . . . . .

. . . . . .

. . . . . .

. . . . . .

2) Tester les deux programmes pour vérification.

Exercice 4

On donne le programme suivant :

```
S=5
for k in range(1,10,2):
    S=S+2*k
    print(S)
```

1) Compléter le tableau suivant donnant les valeurs prises par les variables k et S ?

k	1				
S					

2) Tester le programme pour vérification.

Exercice 5

On donne les deux programmes suivants :

Programme 1

```
x=1
while x<5:
    x=x+1
print(x)
```

Programme 2

```
x=1
while x<5:
    x=x+1
    print(x)
```

1) Qu'obtient-on après exécution de chacun des programmes ?

. . . . . .

. . . . . .

2) Tester les deux programmes pour vérification.

Exercice 6

On donne le programme suivant :

```
a=5
b=1
while a<12:
    a=a+b
    b=b+1
print(a,b)
```

1) Que valent a et b après l'exécution du programme ?

. . . . . .

. . . . . .

2) Tester le programme pour vérification.

Exercice 7

Ecrire un programme Python qui calcule et affiche le carré des 20 premiers nombres entiers.

. . . . . .

. . . . . .

. . . . . .

. . . . . .

TP 4 : fonction

Activité :

Le programme ci-dessous contient une fonction.

```
def premierefonction(x):
    a=4*x
    b=a+5
    c=b/2
    return c
```

1) Identifier le nom de la fonction.

. . . . . .

2) Identifier la variable.

. . . . . .

3) Que retourne la fonction ?

. . . . . .

Le test ci-dessous est réalisé dans la console.

```
*** Console de processus distant Réinitialisée ***
>>>
>>> premierefonction(10)
22.5
```

4) Vérifier à la main le résultat affiché.

. . . . . .

. . . . . .

. . . . . .

. . . . . .

A retenir :

Lorsqu'une tâche doit être réalisée plusieurs fois par un programme avec seulement des paramètres différents, on peut l'isoler au sein d'une fonction.

En langage Python, la syntaxe d'une fonction est la suivante :

```
def nom(argument1,argument2,...):
    ...
    return résultat
```

Pour appeler une fonction, il faut écrire son nom avec les paramètres entre parenthèses :

```
>>> nom(paramètre1,paramètre2,...)
```

Exercice 1

On a défini une fonction Python par les instructions ci-dessous :

```
def g(x):
    return 2*x*(x+4)
```

1) Quel est son nom ?

. . . . . .

2) Combien a-t-elle d'arguments (ou paramètres) ?

. . . . . .

3) Que renvoient les appels suivants :
   a. `g(1)` . . . . . .
   b. `g(2)` . . . . . .
   c. `g(4)` . . . . . .
   d. `3*g(1)+4` . . . . . .

Exercice 2

Les instructions ci-dessous définissent une fonction.

```
def puissances(a):
    carre=a**2
    cube=a**3
    return a,carre,cube
```

1) Quel est son nom ?

. . . . . .

2) Combien a-t-elle d'arguments (ou paramètres) ?

. . . . . .

3) Qu'obtient-on en appelant `puissances(3)` ?

. . . . . .

Exercice 3

On veut définir une fonction Python nommée `aire` qui renvoie l'aire d'un triangle de base b et de hauteur h.

1) Combien a-t-elle d'arguments et lesquels ?

. . . . . .

2) Quel résultat doit-elle renvoyer ?

. . . . . .

3) Compléter le script la définissant.

```
def aire(..............):
    return ..............
```

Exercice 4

On veut définir la fonction Python `f` qui a un nombre associe son cube. La définition ci-dessous comporte trois erreurs. Lesquelles ?

```
def f(t)
    x=x**3
return x
```

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Exercice 5

Les frais d'inscription dans une école dépendant du revenu imposable de l'étudiant. Si ce revenu est inférieur à 4 000€, les frais d'inscriptions s'élèvent à 2% du revenu, sinon ils sont égaux à 3% du revenu.

1) Compléter la fonction ci-dessous qui permet de calculer le montant des frais d'un étudiant.

```
def frais(revenu):
    if revenu<4000:
        resultat=...
    else:
        ...
    return ...
```

2) Utiliser la fonction pour déterminer le montant des frais d'un étudiant dont le revenu est égal à 3 500€.

. . . . . .

3) Utiliser la fonction pour déterminer le montant des frais d'un étudiant dont le revenu est égal à 8 250€.

. . . . . .