Python sur calculatrice

La console Python (ou interpréteur) permet l'exécution de quelques calculs ou d'instructions simples.

Pour accéder à la console Python dans la Numworks :

- dans la page d'accueil, à l'aide des flèches et du bouton OK, lancez l'application Python ;
- descendez jusqu'à "Console d'exécution" puis OK

Exercice 1 : utilisation de la console Python

1°) Essayez de comprendre les réponses données par Python à ces calculs :

Calculs

```
>>> 7 ** 2
>>> 22 // 3
>>> 22 % 3
>>> 1 + 10**(-20)
>>> "top" + "!"
```

```
>>> "top" * 3
>>> "top" + 3
>>> sqrt(8)
>>> cos(pi)
>>> randint(1, 100)
```

Catalogue dans (paste)

appuyez deux fois sur alpha pour passer en mode texte puis encore une fois pour en sortir

alpha (paste)

shift .

En vérité, certaines fonctions ne font pas partie du langage Python de base mais sont définies dans des bibliothèques qu'il faut importer. C'est le cas des fonctions `sqrt`, `cos`, `randint`, etc. Certaines sont importées automatiquement par la Numworks mais d'autres non. Pour importer la fonction `randint`, qui fait partie de la bibliothèque `random`, il faut taper :

```
>>> from random import randint
```

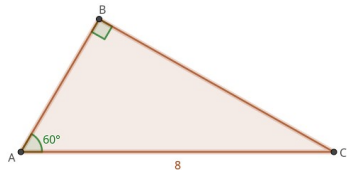
(vous pouvez utiliser le bouton « boîte à outils (paste) »).

2°) Après l'avoir importée, exécutez plusieurs la fonction `randint`. Quel est sa utilité ?

3°) En utilisant la console Python, trouvez une valeur approchée de l'hypoténuse d'un triangle rectangle dont les côtés de l'angle droit mesurent 10 cm et 4 cm.

4°) En utilisant la console Python et la trigonométrie, calculez la longueur AC.

Indication : utilisez le sinus de 60°.



Exercice 2 : affectations, variables

1°) Essayez de prévoir la valeur finale de `a` à la fin du programme suivant puis tapez les instructions dans la console pour vérifier votre réponse :

Instructions	Commentaires
`>>> a = 5`	affecter 5 à a
`>>> b = 3` (shift π pour le =)	affecter 3 à b
`>>> a = a - b`	affecter la valeur de a – b à a
`>>> b = a ** 2`	affecter la valeur de a^2 à b
`>>> a`	afficher la valeur de a

Dans la console il suffit de taper le nom d'une variable pour voir sa valeur.

2°) Sans Python, déterminez la valeur de `x` à la fin de chacun des suites d'instructions suivantes :

```
>>> a = 2
>>> b = 6
>>> b = a * b
>>> a = a ** 2
>>> x = b // a
```

```
>>> a = 5
>>> x = a + 4
>>> x = x * a
>>> y = a + 1
>>> z = a + 3
>>> y = y * z
>>> x = x - y
```

```
>>> d = 4
>>> x = d - 3
>>> f = x - 3
>>> x = x ** 2
>>> x = x - d*f
```

3°) Vérifiez vos réponses avec la console Python.

4°) Changez la valeur initiale de `a` dans le 2ième algorithme. Que remarquez-vous ? Comment expliquer cela ?

5°) Changez la valeur initiale de `d` dans le 3ième algorithme. Que remarquez-vous ? Comment expliquer cela ?

Exercice 3 : types de données

1°) Tapez dans la console et observez bien les réponses de Python :

```
>>> type(30)
>>> type("test")
```

La commande type(…) demande le type de donnée d'une valeur.
En Python, les entiers sont de type *int* (integer = entier) et les textes de type *str* (string = chaîne).

2°) Comment Python appelle-t-il les nombres à virgules ?

3°) Tapez dans la console et observez bien les réponses de Python :

```
>>> nb = 10
>>> txt = "test"
>>> txt + nb
```

On ne peut pas ajouter un texte et un nombre. Si je veux obtenir le texte "test10", il faut convertir le nombre 10 en le texte "10". Ceci se fait à l'aide d'une fonction de trans-typage, ici str(…). Il existe également les commandes int(…) et float(…) pour convertir en entier ou en réel.

3°) Tapez donc dans la console :

```
>>> chn+str(nb)
```

4°) Tapez et complétez ci-dessous les commandes manquantes.
Les nombres 38 et 7 ne doivent pas apparaître dans les lignes cachées.

Commandes	Résultats attendus	Commentaires
`>>> a = "38"` `>>> b = 7.0` `>>>` ..... `>>>` ..... `>>>`	 266 5.428571428571429 5	 produit de 38 et 7 quotient de 38 par 7 il y a 5 fois 7 dans 38

Remarquez que la conversion du type entier vers le type réel est automatique (exemple : 4 + 7.02).

Utilisation de l'éditeur Python

L'éditeur Python permet la sauvegarde et l'exécution de programmes plus complexes. Pour accéder à l'éditeur Python dans la Numworks :

- dans la page d'accueil, à l'aide des flèches et du bouton OK, lancez l'application Python ;
- descendez jusqu'à "Ajouter un script" puis OK, donnez un nom au nouveau script, par exemple "algo", en utilisant les touches de la calculatrice (les lettres y apparaissent en gris) puis OK pour valider le nom et OK pour **éditer** le programme ;
- vous taperez plus tard un programme ;
- utilisez (⮌) pour sortir de l'éditeur ;

pour exécuter un programme : avec la flèche droite, allez sur les trois points puis OK et choisissez « Exécuter le script » ;

Exercice 4 : utilisation de l'éditeur Python

Voici un exemple de traduction d'un algorithme en programme :

Algorithme en langage naturel	Algorithme en pseudo-code	Programme en Python
• étant donné un nombre ; • lui ajouter 4 et multiplier le résultat par le nombre initial ; • soustraire le carré du nombre initial ; • afficher le résultat.	y ← x + 4 z ← y*x a ← z – x^2 Afficher a	`y = x + 4` `z = y*x` `a = z - x**2` `print(a)`

taper pr puis appuyer sur la flèche droite

1°) Entrez ce programme dans l'éditeur Python. Exécutez-le.

2°) Il y a une erreur : lisez-là, comprenez-là puis corrigez-là…

3°) Exécutez votre programme pour chacun des nombres suivants :

Nombre choisi	0	1	3	10
Résultat du programme				

4°) Que remarquez-vous ? Expliquez.

La fonction print (→ sorties)

En mode éditeur, pour faire afficher quelque chose il faut utiliser la fonction : `print(…)`

Par exemple, pour afficher le résultat du calcul 8 × 7, on taperait :

```
print(8*7)
```

Par exemple, pour afficher la valeur d'une variable nommée *a*, on taperait :

```
print(a)
```

Pour afficher un texte, on utilise des guillemets :

```
print("Bonjour !")
```

Pour mixer les deux, on sépare avec des virgules :

```
print("2 fois 3 =", 2*3)
```

La fonction input (→ entrées de l'utilisateur)

Pour demander à un utilisateur de rentrer une valeur utilisée par un programme (pensez par exemple au code pin de votre téléphone), nous pouvons utiliser la fonction `input(…)`

Par exemple, les deux instructions :

```
print("Quel est votre code ?")
votre_code = input()
```

ou, en plus court l'instruction :

```
votre_code = input("Quel est votre code ?")
```

affichent le texte « Quel est votre code ? », attendent la réponse de l'utilisateur et placent la réponse dans la variable `votre_code` (qu'on pourra utiliser plus tard, pour vérification d'un code pin par exemple).

Exercice 5 : un problème avec input

Testez le programme suivant et essayez de le réparer…

```
nb = input("nombre ?")
print("2*", nb, "=", 2*nb)
```

La fonction `input(…)` renvoie toujours du texte ! Il sera parfois nécessaire de convertir ce texte en entier ou en réel, par exemple :

```
nb = int(input("nombre ?"))
```

Exercice 6 : quelques programmes en Python

Créez les programmes suivants dans Python (mode éditeur) et lancez-les pour vérifier qu'ils fonctionnent. Imaginez que ces programmes sont destinés à un autre utilisateur, qui n'a pas accès à vos programmes. Faîtes vérifier (valider) chaque programme par votre professeur.

Programme 1 :

La machine demande à un utilisateur son année de naissance (exemple : l'utilisateur entre 2008) et l'année actuelle (exemple : 2023) et affiche le message :

Tu as … ans (exemple : Tu as 15 ans)

Programme 2 :

La machine demande à un utilisateur la longueur du côté d'un carré et affiche le périmètre et l'aire de ce carré.

Exercice 7 : Le magicien

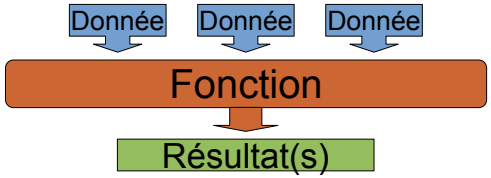
Un magicien demande à un spectateur :

- de penser à un nombre entier ;
- de le multiplier par 5 ;
- d'ajouter 7 au résultat ;
- de multiplier par 4 le résultat ;
- d'ajouter 6 au résultat ;
- de multiplier par 5 le résultat ;
- d'annoncer le résultat final obtenu.

1°) Le spectateur pense au nombre 4, quel nombre annonce-t-il à la fin ? 2°) Le magicien trouve à chaque fois le nombre choisi au départ par le spectateur ! Soit il est très fort en calcul mental, soit il a un truc de magicien… Écrivez un programme Python qui : – demande un nombre entier ; – effectue les opérations demandées par le magicien ; – affiche le résultat final (celui que le spectateur annonce).	3°) Entrez ce programme dans l'éditeur Python puis vérifiez la réponse à la question 1°). Relancez plusieurs fois votre programme, choisissez d'autres valeurs de départ et cherchez un lien entre le nombre choisi par le spectateur et celui qu'il annonce. 4°) On appelle x le nombre choisi par le spectateur. Écrivez en fonction de x le résultat qu'il annonce. Prouvez alors la remarque faîte au 3°).

Les fonctions

Une fonction en Python est en fait… un programme !
Elle reçoit donc des données en entrée et produit un résultat (affichage, dessin, nouvelles données) en sortie.



Entrée

Sortie

Pour définir une fonction en Python, il faut utiliser le mot clé def.

Mot clé pour définir une fonction

Nom choisi pour la fonction

Paramètre(s) de la fonction : données à fournir à la fonction

```
def mafonction(donnees):
    instruction1
    instruction2
    ...
    return nouvelles_données
```

Les instructions de la fonction doivent être décalées vers la droite (indentation)

N'oubliez pas ce :

Instruction à utiliser si la fonction renvoie des nouvelles données

Nouvelles données produites par la fonction

Un programme bien écrit est découpé en plusieurs fonctions (plusieurs sous-programmes).

Ne confondez pas définition et appel d'une fonction :

- définir une fonction (avec `def mafonction(…):`) veut dire l'enregistrer dans la mémoire de la calculatrice ;
- appeler une fonction (avec `mafonction(…)`) veut dire l'exécuter en lui fournissant les données dont elle a besoin.

Exercice 8 : définition et appels

1°) Voici un exemple de définition puis d'appel d'une fonction qui calcule le périmètre d'un rectangle.
Les textes qui suivent # sont des commentaires : ils ne sont pas vus par Python.
Tapez (sans les commentaires…) et exécutez ce programme.

```
# définition d'une fonction
def peri(a, b):
    c = 2*a + 2*b
    return c

# appel de la fonction :
# dans l'éditeur :
print(peri(4, 7))
```

touche ×

2°) Dans la console Python, vous pouvez relancer la fonction en tapant :

```
>>> peri(8, 3)
```

vous pouvez également stocker dans une variable une valeur renvoyée par la fonction :

```
>>> a = peri(10, 5)
>>> a
```

3°) Dans un cinéma, les places adultes coûtent 11 € et les places enfants coûtent 6 €. Écrivez une fonction en Python qui calcule le coût d'un film en fonction du nombre x d'adultes et du nombre y d'enfants.
Testez votre fonction avec 5 adultes et 8 enfants.

4°) La formule de Héron d'Alexandrie donne l'aire d'un triangle quelconque de côtés a, b, c : Aire = $\sqrt{p(p-a)(p-b)(p-c)}$ où p est le demi-périmètre.

a) Écrivez une fonction qui calcule p en fonction de a, b et c.

```
def demiperi(a, b, c):
    p = ……
    return p
```

b) Écrivez une fonction qui calcule l'aire en fonction de a, b et c et qui utilisera la fonction précédente.

```
def aire(……):
    p = demiperi(a, b, c)
    ……
    return ……
```

c) Testez cette fonction pour trouver l'aire des triangles suivants :
– côtés : 4 cm ; 7 cm ; 10 cm (réponse : environ 10,93 cm²).
– côtés : 8 cm ; 5 cm ; 11 cm

Exercice 9 : utilisation en géométrie

1°) Nous voulons écrire une fonction donnant la longueur de l'hypoténuse d'un triangle rectangle. Combien de nombres y a-t-il en entrée de la fonction ? Et en sortie ?

2°) Complétez ce programme(*) :

```
def hyp(.....):
    res = .....
    return .....
```

3°) Entrez ce programme dans la calculatrice et testez la fonction avec les valeurs suivantes :

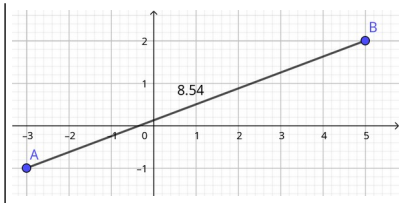
Côtés de l'angle droit	1 et 1	3 et 4	$\sqrt{5}$ et 2	33 et 56	16 et 63
Hypoténuse approximative					

4°) Pour calculer la distance entre deux points A et B (dans un repère orthonormé) nous avons besoin de leurs coordonnées. Nous pouvons alors calculer les deux coordonnées X et Y du vecteur $\overrightarrow{AB}$ puis sa norme.

a) Complétez cette fonction (utilisez la fonction hyp) :

```
def dist(xA, xB, yA, yB):
    X = .....
    Y = .....
    d = .....
    return .....
```

b) Entrez cette fonction dans la calculatrice, dans le même script que la fonction hyp, et testez-la avec les points ci-contre.



5°) Ajoutez une fonction qui renvoie les coordonnées du milieu d'un segment puis testez-là avec ces deux points.

Exercice 10 : pour les plus rapides

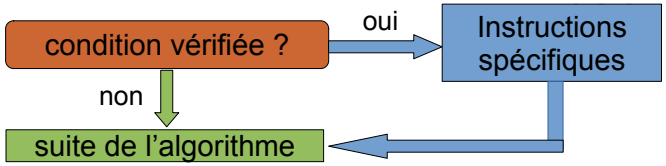
Soient trois points A, B, C dont les coordonnées sont connues. Écrivez un programme qui, à partir des coordonnées de ces trois points, donne les coordonnées du point D tel que $ABCD$ soit un parallélogramme.

Testez-le avec les points suivants : A (– 1 ; 3), B (2 ; – 3), C (4 ; 2).

Tests (instruction « Si … alors … »)

Un programme doit parfois pouvoir s'adapter aux circonstances.
Cela se fait grâce à des tests (ou instructions conditionnelles).

Si *condition vérifiée* **alors** *instructions à effectuer*



La condition est une comparaison, telles que, par exemple :
temperature = 0 ou hauteur < 3

Voici un exemple volontairement simpliste :
Si météo=*pluie* **alors** *prendreparapluie=oui*

Exercice 11 : tests en pseudo-code

Indiquez la valeur affichée par chacun des algorithmes suivants :

Algorithme 1 :

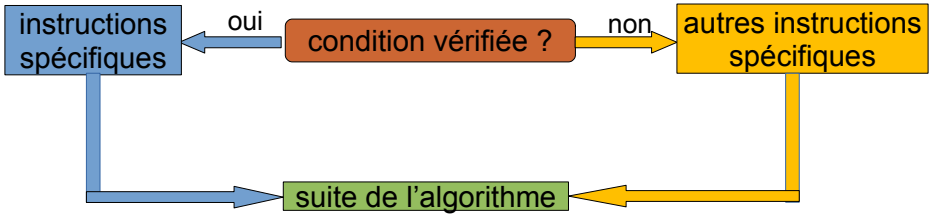
```
i ← 5
p ← 25
Si 2*i < 10
alors p ← 32
Afficher p
```

Algorithme 2 :

```
f ← 5
b ← 3
Si b + f  >  20
alors b ← b + f
Afficher s
```

Tests (instruction « Si … alors … sinon … »)

Si *condition vérifiée* **alors** *instructions à effectuer* **sinon** *autres instructions à effectuer*



Exemple :
Si j'ai assez d'argent **alors** je m'achète la TX3000 **sinon** j'achète la Z20
qu'on pourrait traduire ainsi (en supposant que la TX3000 coûte 500 €) :
Si *argent ≥ 500* **alors** *achat= « TX3000»* ***sinon*** *achat = « Z20 »*

Exercice 12 : tests en pseudo-code

Indiquez la valeur affichée par chacun des algorithmes suivants :

Algorithme 1 :

```
x ← 4
y ← 3
Si x**y < 50
alors p ← 32
sinon p ← x + 2*y
Afficher p
```

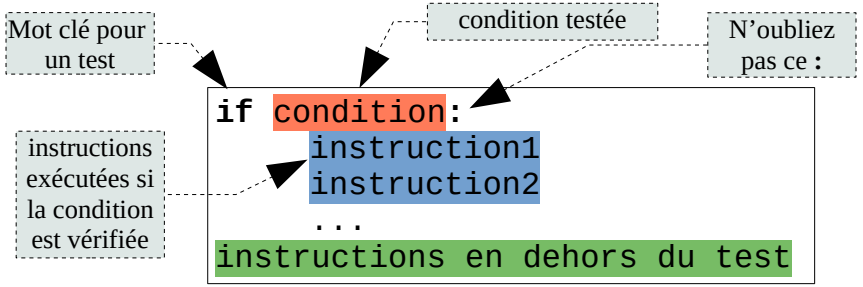
Algorithme 2 :

```
f ← 5
b ← 3
s ← 20
Si b + f  >  s
alors s ← b + f
sinon s ← s + 1
Afficher s
```

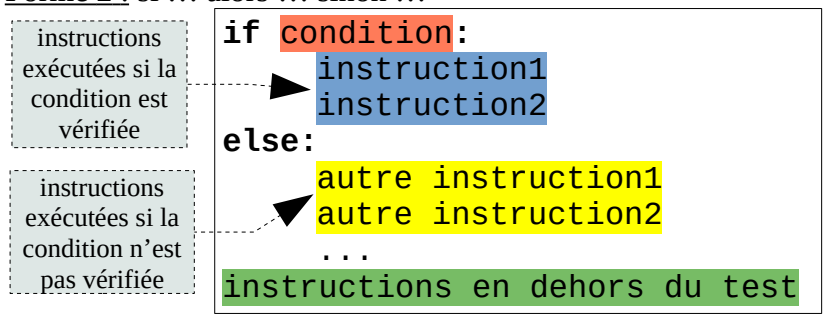
Tests en Python

Pour programmer un test en Python, il faut utiliser le mot clé if.

Forme 1 : si … alors …



Forme 2 : si … alors … sinon …



Remarque : il existe une troisième forme en Python (`elif` = *sinon si*) :

```
if condition1:
      instructions1
elif condition2:
      instructions2
else:
      instructions3
```

Exercice 13 : tests en Python

1°) Indiquez ce qu'affiche le programme ci-contre suivant les valeurs de x quand :
$x = 2$ $x = 8$ $x = -1$ $x = -3$

```
if -3*x+19 < 0:
    print("négatif")
else:
    print("positif")
```

2°) Trouvez toutes les valeurs de x pour lesquelles le programme affiche « positif ».
3°) Modifiez le programme pour qu'il gère un troisième cas oublié…

Exercice 14 : tests dans une fonction Python

La valeur absolue d'un nombre x, notée $|x|$, est égale à x s'il est positif et à $-x$ s'il est négatif. Par exemple, $|4| = 4$ et $|-5| = 5$.
Écrivez une fonction `abso` qui prend en paramètre un nombre x et qui renvoie sa valeur absolue. Testez-là ensuite avec votre calculatrice :

```
>>> abso(4)
4
>>> abso(-5)
5
```

Exercice 15 : tests en géométrie

1°) Dans un repère orthonormé, soit (C) le cercle de centre A (– 2 ; 3) et de rayon 5. Le point B (– 3 ; – 2) appartient-il à (C) ?
2°) Écrivez une fonction qui prend en paramètres les coordonnées d'un point quelconque M et détermine si ce point est sur le cercle (C) ou pas.

Exercice 16 : petit jeu

Écrivez un programme en Python du jeu suivant :
– la machine choisit un nombre mystère entre 1 et 100 ;
– un premier joueur entre un nombre dans la machine ;
– un second joueur entre un autre nombre dans la machine ;
– la machine dit lequel des deux a été le plus proche du nombre mystère.

Il est possible d'utiliser les opérateurs ou (or) et et (and) quand les conditions deviennent plus complexes.

Exercice 17 : or, and

1°) Indiquez ce qu'affiche le programme ci-contre suivant les valeurs de x quand :
$x = 2$ $x = 13$ $x = -1$ $x = 7$

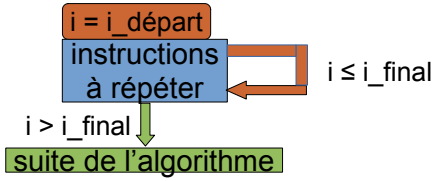
```
if x < 4 or x > 10:
    print("trop loin")
else:
    print("près")
```

2°) Quel est le sens de l'affichage « trop loin » ?
3°) Bonus : ré-écrivez le programme sans utiliser de `or`…

Boucle « Pour »

Les machines sont très utiles pour les tâches répétitives !
Dans une boucle Pour, nous demandons à ce que des instructions soient répétées pour chaque valeur entière d'un compteur entre deux bornes.

Pour i **allant de** i_depart **à** i_final
    instructions à répéter
FinPour



Le nom du compteur, ici i, n'a aucune importance (les lettres i, j, k, … désignent habituellement des nombres entiers en informatique).
Le compteur peut être utilisé dans les instructions ou ne pas l'être.
Les répétitions sont aussi appelées *itérations*.

Exemple 1 :

Pour i **allant de** 0 **à** 99
Afficher « Bonjour ! »
FinPour

Ceci affiche « Bonjour ! » **100** fois.

Exemple 2 :

Pour k **allant de** 1 **à** 5
Afficher k^2
FinPour

Ceci affiche les carrés de tous les nombres de 1 à 5.

Exemple 3 :

s ← 2
Pour n **allant de** 0 **à** 9
s ← s²
FinPour
Afficher s

Ceci met 2 au carré puis le résultat (4) au carré puis le résultat au carré (donc 16) et ceci 10 fois de suite. Affiche seulement le résultat final.

Exercice 18 : boucle Pour en pseudo-code

1°) Complétez le tableau ci-dessous. Qu'affiche l'algorithme ?

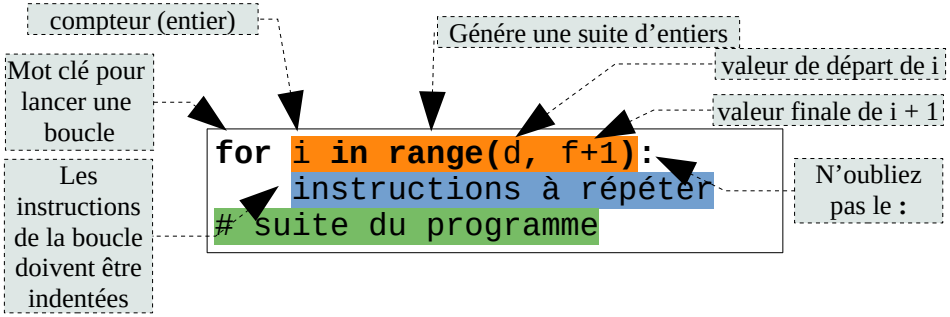
Algorithme
s ← 20 **Pour** i **allant de** 1 **à** 4 s ← s + 4 **FinPour** **Afficher** s

	i	s
Avant la boucle		
Fin 1ière itération		
Fin 2ième itération		
Fin 3ième itération		
Fin 4ième itération		

2°) Que se passe-t-il si je remplace « **Pour** i **allant de** 1 **à** 4 » par « **Pour** j **allant de** 0 **à** 3 » ?

Boucle « Pour » en Python (instruction for)

Pour créer une boucle Pour en Python, utlisez les mots clés for … in ….



Traduction en Python des trois exemples précédents.

Exemple 1	Exemple 2	Exemple 3
`for i in range(100):` `  print("Bonjour")`	`for k in range(1, 6):` `  print(k**2)`	`s = 2` `for n in range(10):` `  s = s**2` `print(s)`

Attention : remarquez que `range(100)` donne les nombres entiers de 0 à 99, pas de 0 à 100 (ni de 1 à 100). De même pour `range(1, 6)` qui donne les nombres entiers de 1 à 5.

Exercice 19 : boucle Pour en Python (for)

1°) Traduisez les algorithmes en langage Python :

Algorithme 1	**Python**	**Algorithme 2**	**Python**
s ← 20 **Pour** i **allant de** 0 **à** 9 s ← 2 * s + 1 **FinPour** **Afficher** s		s ← 0 **Pour** i **allant de** 1 **à** 9 s ← s + i **FinPour** **Afficher** s	

2°) Écrivez une fonction de paramètre *n* qui calcule et affiche la somme des entiers de 1 à *n*. Entrez cette fonction dans la calculatrice et testez-là.
3°) Essayez de trouver une formule : 1 + 2 + 3 + … + *n* = ……

Exercice 20 : factorielles

La factorielle d'un entier *n* positif, notée *n* ! est le produit de tous les entiers positifs non nuls inférieurs ou égaux à *n* : $n ! = 1 \times 2 \times 3 \times \ldots \times n$
C'est le nombre d'anagrammes d'un mot de *n* lettres ne comportant pas de lettre répétée (exemple : MATH a 4 ! = 24 anagrammes).
Écrivez une fonction `facto` qui calcule la factorielle d'un entier *n* positif.

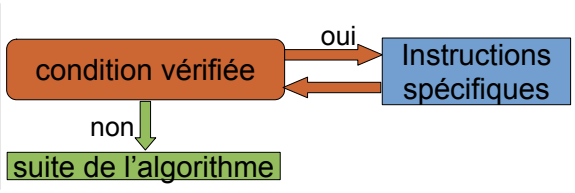
Exercice 21 : tables de multiplication

Écrivez un programme qui affiche les tables de multiplication :
0 * 0 = 0
0 * 1 = 0
…
9 * 9 = 81
Indication : une boucle dans une boucle…

Boucle « Tant que »

Une boucle Tant que ressemble à un test sauf que les instructions seront répétées *tant que* la condition sera vérifiée.

Tant que *condition*
Faire *instructions*
FinTantQue



Attention :

- il faut que la condition devienne fausse à un moment, sinon il y aura une boucle *infinie* ;
- contrairement à un compteur de boucle `for`, une variable utilisée dans la condition doit avoir été définie avant la boucle Tant que.

Exercice 22 : tant que en pseudo-code

Trouvez *x* pour chaque algorithme.

```
i ← 5
Tant que i < 10
Faire
    x ← 3 * i
    i ← i + 1
FinTantQue
```

```
i ← 5
Tant que condition
Faire instructions
FinTantQue
```

```
i ← 5
Tant que condition
Faire instructions
FinTantQue
```

Exercice 23 : rebonds

Une balle est lâchée de 2 mètres de hauteur et perd 10 % de sa hauteur maximale en rebondissant au sol.

Complétez l'algorithme ci-contre pour qu'il donne le nombre de rebonds *n* nécessaires pour que la balle ait perdu 90 % de sa hauteur maximale.

```
h ← 2
n ← …
Tant que ……
Faire
……
……
FinTantQue
Afficher ……
```

Boucle « Tant que » en Python (while)

Pour créer une boucle Pour en Python, utlisez les mots clés for … in ….

Mot clé pour lancer une boucle tant que

N'oubliez pas le :

```
while condition vérifiée:
    instructions à répéter
# suite du programme
```

Les instructions de la boucle doivent être indentées

Exercice 23bis : rebonds

Traduisez l'algorithme en Python et trouvez le nombre de rebonds avec la calculatrice.

Exercice 24 : nombre mystère

Écrivez en Python un programme de jeu suivant ces règles :

- la machine choisit un nombre entier au hasard entre 1 et 100 ;
- le joueur propose un nombre, la machine lui dit si la proposition est trop grande ou trop petite ;
- le joueur propose un autre nombre tant qu'il n'a pas trouvé ;
- la machine affiche à la fin le nombre d'essais nécessaires.

Testez ce programme avec votre calculatrice.