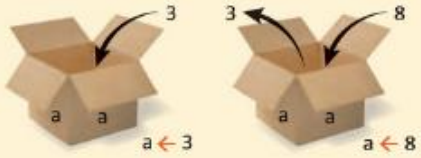
Variables et premières opérations

A Notion de variable

- Définir une variable, c'est associer un nom à une valeur qu'on peut modifier au cours du temps. On dit qu'on affecte une valeur à une variable.

Exemple : on peut associer la lettre *a* à la valeur 3.

En langage naturel, on note a ← 3. On dit que *a* prend la valeur 3 ou de manière équivalente qu'on affecte 3 à *a*.
Ainsi, dans la suite d'affectations :
a ← 3
a ← 8
la variable *a* vaut 3, puis 8.
À la fin de cette suite d'instructions, la variable *a* vaut 8.



En langage Python, on affecte une valeur à une variable grâce au signe = .
L'instruction `a=3` permet donc d'affecter la valeur 3 à la variable nommée *a*.

- La valeur peut être de nature différente : nombre entier, nombre décimal, texte (chaîne de caractères avec des guillemets). On parle alors de type.

Nom du type	Mot-clé en Python	Exemple
entier relatif (integer)	int	`a=4`
nombre décimal (flottant)	float	`a=2.3`
texte (string)	str	`a="bonjour"`

B Opérations sur les variables

- L'action que l'on peut réaliser sur une variable dépend fortement de son type. Par exemple, on peut se demander quelle est la longueur d'un texte alors qu'on ne se demandera pas quelle est la longueur de −2.

Remarque : pour obtenir la longueur d'une chaîne de caractères, on utilise `len` en Python.

Exemple : dans la console Python :

```
>>> a=4
>>> b=2.4
>>> a+b
6.4
>>> a-b
1.6
```
Somme et différence d'un entier et d'un flottant

```
>>> a=3
>>> b=2
>>> a*b
6
>>> a/b
1.5
```
Produit et quotient de deux entiers

```
>>> a=3
>>> b=2
>>> a**b
9
```
Puissance d'entiers

```
>>> a="bon"
>>> b="jour"
>>> a+b
"bonjour"
```
Somme (concaténation) de chaînes de caractères

```
>>> a="bonjour"
>>> len(a)
7
```
Longueur d'une chaîne de caractères

Réponses du qcm : 1 B 2 C 3 C 4 B

Réponses Automatismes : 1) nb 2) 3 3) flottant 4) age, nom, prenom 5) b, c, e 6) 18

(p.4 à p.7 du cahier d'algo)

QCM

Cocher la bonne réponse

1 Combien y a-t-il de variables dans l'algorithme ci-dessous ?

```
a ← 2
b ← 3
a ← 4
c ← b
```

a. ☐ 2 b. ☐ 3 c. ☐ 4

2 Que valent les variables *a* et *b* à la fin de l'exécution de l'algorithme suivant ?

```
a ← 1
b ← 2
a ← b
```

a. ☐ *a* = 1 et *b* = 2
b. ☐ *a* = 2 et *b* = 1
c. ☐ *a* = 2 et *b* = 2

3 Que vaut la variable *a* à la fin de l'exécution du programme suivant ? www.lien

```
1 a=1
2 b=4
3 a=b
4 a=b+a
```

a. ☐ 2 b. ☐ 5 c. ☐ 8

4 Quel est le type de la variable *a* dans l'instruction ci-dessous ?

```
a=2.01
```

a. ☐ string b. ☐ float c. ☐ integer

Automatismes à compléter

1
```
nb=7
```
Nom de la variable :

2
```
x=3
```
Valeur de la variable :

3
```
val=12.4
```
Type de la variable val :

4
```
1 age=16
2 nom="Moulin"
3 prenom="Paul"
4 age=age+3
```
Variables :

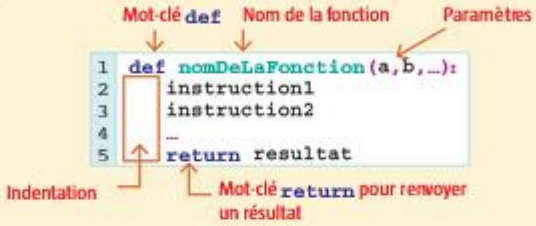
5
```
1 a=3
2 b="Sarah"
3 c="Arthur"
4 d=3+4
5 e=b+c
```
Variables qui sont des chaînes de caractères :

6
```
1 a=4
2 a=a+2
3 a=a*3
```
Valeur de *a* :

Fonctions

A Définition d'une fonction

- Une **fonction**, en programmation, est un sous-programme qu'on peut utiliser à volonté dans le programme principal.
- Écrire des fonctions permet d'**organiser** et de simplifier les programmes.
- Une fonction a un **nom**, peut prendre des valeurs en entrée données sous la forme de variables appelées **paramètres**, et peut renvoyer un ou plusieurs résultats. En langage Python, elle est structurée de la manière suivante :

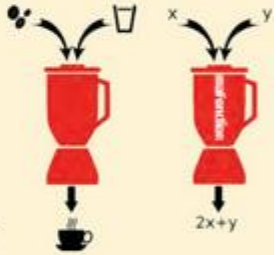


- Les **instructions** contenues dans la fonction sont décalées vers la droite. Ce décalage, appelé **indentation**, peut être réalisé en créant 2 ou 4 espaces ou en utilisant la touche tabulation ⇥.

Exemple :

On considère la fonction suivante :

```
1 def maFonction(x,y):
2     resultat=2*x+y
3     return resultat
```



Le nom de la fonction est `maFonction`, elle a deux paramètres en entrée x et y, et elle renvoie le nombre $2 \times x + y$.

Remarques :

– il faut ajouter deux points « : » au bout de la ligne de définition de la fonction ;

– pour renvoyer la valeur de la variable resultat, on peut écrire `return` resultat ou `return` (resultat) ;

– il est également possible de renvoyer plusieurs valeurs en les séparant par des virgules.

```
1 def doubleTriple(x):
2     return 2*x,3*x
```

B Appel d'une fonction

- Pour appeler (c'est-à-dire utiliser) **une fonction**, il faut écrire son nom avec les **valeurs d'entrée**, appelées **paramètres**, entre parenthèses et dans le bon ordre. Ainsi, pour appeler la fonction `maFonction` pour les valeurs d'entrée (paramètres) $x = 2$ et $y = 3$, il faut écrire l'instruction suivante sous la définition de la fonction :

```
a=maFonction(2,3)
```

- Utiliser des fonctions permet également de modifier facilement un programme. Ainsi, si le calcul n'est plus $2 \times x + y$ mais $5 \times x + y$, il suffira de le changer une seule fois dans la fonction. La ligne 2 de `maFonction` sera alors :

```
2 resultat=5*x+y
```

QCM

Cocher la bonne réponse

1 Quel mot-clé définit une fonction ?

a. ☐ func
b. ☐ procedure
c. ☐ def

2 Quel est le nom de la fonction ?

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

a. ☐ *x*
b. ☐ carre
c. ☐ def

3 Quels sont les noms des paramètres de la fonction ?

```
1 def somme(a,b):
2     s=a+b
3     return s
```

a. ☐ *a* et *s*
b. ☐ *a*, *b* et *s*
c. ☐ *a* et *b*

4 Comment utilise-t-on la fonction `f` avec le paramètre 3 ?

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

a. ☐ *f*[3]
b. ☐ *f*(3)
c. ☐ *f*{3}

Automatismes à compléter

1

```
1 .....f(x):
2     return x*3+1
```

Mot-clé manquant : ☐

2

```
1 def cube(x):
2     ..... x**3
```

Mot-clé manquant : ☐

3

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

Nombre de paramètres : ☐

4

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

Nom des paramètres : ☐

5

```
1 def g(a):
2     return a-5
```

Valeur renvoyée par l'instruction *g*(7) : ☐

6

```
1 def h(a,b):
2     return a+b-1
```

Valeur renvoyée par l'instruction *h*(4,3) : ☐

Réponses du qcm : 1 A 2 B 3 C 4 B

Réponses Automatismes : 1) def 2) return 3) 2 4) x, y 5) 7 – 5 = 2 6) 4 + 3 – 1 = 6

(p.8 à p.11 du cahier d'algo)

Instructions conditionnelles

A Les conditions dans un test

Une **condition** est une expression dont le résultat est soit « vrai » soit « faux ».
Une condition peut être construite à l'aide :

- d'**opérateurs de comparaison** :

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

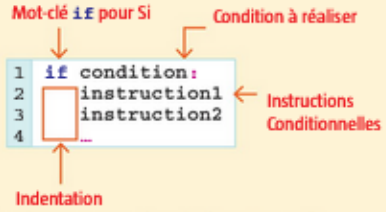
- d'**opérateurs logiques** :

Opérateur logique (Python)	and	or	not
Signification	et	ou	non

Exemple :
```
condition=(heure>9)and(jour=="lundi")
```
La condition est vraie si l'heure est supérieure à 9 et le jour est égal à "lundi".

B Structure conditionnelle « if » (Si... alors)

- Une **instruction conditionnelle** est une instruction qui n'est exécutée que si une condition est réalisée. Autrement dit, si une condition est réalisée **alors** l'instruction est réalisée.
- La condition est suivie de **deux points**.
- Les instructions liées à la condition doivent être **indentées** (décalées vers la droite). On peut dire que c'est l'indentation qui remplace le mot « alors » qui n'existe pas en langage Python.



Exemple :
```
1 x=2
2 a=4
3 if 6*a-4>1:
4     x=3
5 print(x)
```
Ligne 3, il faut que $6*a-4$ soit supérieur à 1 pour que, dans la ligne 4, on ait $x = 3$. Dans le cas contraire, x garde sa valeur.

C Structure conditionnelle « if... else » (Si... Sinon)

La structure **if... else** (si... sinon) permet d'exécuter des instructions si une condition est réalisée et d'autres instructions **sinon**.

```
1 a=4
2 if 6*a-4>1:
3     x=3
4 else:
5     x=4
```
Condition
Si la condition est réalisée, x=3
Si la condition n'est pas réalisée, x=4

D Structure conditionnelle « if... elif... else » (Si... Sinon Si... Sinon)

La structure **if... elif... else** permet d'utiliser plusieurs conditions qui s'excluent l'une l'autre.

Le mot-clé **elif** est la contraction de « else if » qui signifie « **sinon si** ».

Il est possible d'utiliser autant de « elif » qu'on le souhaite.

```
1 a=4
2 if 6*a-4>1:
3     x=3
4 elif:3>6*a-4
5     x=2
6 else:
7     x=4
```
Condition 1
Si la condition 1 est réalisée, x=3
Condition 2
Si la condition 2 est réalisée mais pas la condition 1, x=2
Si aucune condition n'est réalisée, x=4

Réponses du qcm : 1 B 2 A 3 C 4 A

Réponses Automatismes :

1) sinon si 2) Si x strictement supérieur à 2 et y égale à 5 3) 3 ou 4

4) 4 5) Si a > 2 6) elif (p.16 à p.19 du cahier d'algo)

QCM

Cocher la bonne réponse

1 Après la condition dans une instruction conditionnelle, on place :
a. ☐ le mot then
b. ☐ deux points
c. ☐ une virgule

2 Dans l'instruction suivante `if a>3:`
$a > 3$ est une :
a. ☐ condition b. ☐ instruction c. ☐ structure

3 Pour ajouter une alternative dans une instruction conditionnelle, on utilise le mot-clé :
a. ☐ other b. ☐ else c. ☐ or

4 Qu'affiche le programme suivant ?

```
1 a=3
2 if a<=3:
3     print(a)
4 else:
5     print(a-2)
```
a. ☐ 3
b. ☐ 1
c. ☐ 31

Automatismes à compléter

1
`elif`
Signification de ce mot-clé :

2
`if x>2 and y==5`
Traduction en langage naturel :

3
```
1 x=2
2 if a>1:
3     x=3
4 elif a<=1:
5     x=4
6 else:
7     x=5
```
Valeurs possibles pour x :

4
```
1 a=2
2 b=3
3 if a>1:
4     b=4
```
Valeur de b :

5
```
1 x=3
2 if a>2:
3     x=2
```
Condition pour que x soit égal à 2 à la fin du programme :

6
```
if x<3:
    print("gagné")
else:
    if x<5:
        print("perdu")
```
Structure conditionnelle mieux adaptée ici :

Boucles bornées (p.20 à p.23 du cahier d'algo)

A Définition d'une boucle bornée

Une **boucle** permet de répéter des instructions. Lorsqu'on connaît à l'avance le nombre de répétitions, on parle de **boucle bornée**. L'instruction **for** (pour) permet de créer une boucle bornée.

B Boucle « for » élémentaire

Pour répéter *n* fois un ensemble d'instructions, la syntaxe est la suivante :

```
for i in range(n):
    instruction1
    instruction2
    ...
```

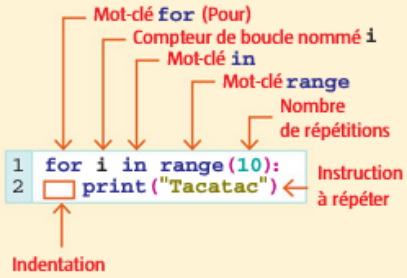
Exemple :

Ligne 1 : les mots-clés `for`, `in` et `range()` créent une boucle **for** en Python. Une variable **entière**, nommée *i* dans cet exemple, indique le numéro de la boucle en cours. Dans ce cas, elle vaut successivement 0, 1, 2, 3, 4,... , 9. Elle commence avec la valeur 0 et est donc égale à 9 (et non 10 comme on pourrait s'y attendre) au bout des 10 tours de boucle.

Cette variable *i* n'est pas nécessairement utilisée dans l'instruction à répéter, on l'appelle **compteur**.

Ligne 2 : l'instruction à répéter est indentée (décalée vers la droite).

Ce programme affiche 10 fois le mot « Tacatac ».



C Utilisation du compteur

Il est possible d'utiliser le **compteur de boucle** dans les instructions à répéter.

Exemple : les deux programmes suivants sont équivalents.

Programme 1

```
1 for i in range(3):
2     x=2*i
```

Programme 2

```
1 x=2*0
2 x=2*1
3 x=2*2
```

Dans le programme 1, le compteur vaut successivement 0, 1 et 2. Il est utilisé pour le calcul de *x* à chaque étape.

Remarque : il est possible de faire varier le compteur entre deux bornes en donnant deux paramètres au `range` comme dans `for in range (2,5)`.

Attention ! Dans ce dernier cas, le compteur prend la valeur 2 mais ne prend pas la valeur 5.

QCM Cocher la bonne réponse

1 Quelle est la première valeur affichée par le programme suivant ?

```
1 for i in range(100):
2     print(4*i)
```

a. ☐ 1 b. ☐ 0 c. ☐ 4

2 Quelles sont les valeurs successives de *i* dans le programme suivant ?

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

a. ☐ 0, 1, 2 b. ☐ 1, 2, 3 c. ☐ 0, 1, 2, 3

4 Quelles valeurs de *a* et *b* faut-il choisir pour que la variable *i* soit successivement égale à : 5, 6, 7, 8 et 9 ?

```
for i in range(a,b):
```

a. ☐ $a = 5$ et $b = 9$ b. ☐ $a = 4$ et $b = 9$ c. ☐ $a = 5$ et $b = 10$

5 Que vaut la variable *a* à la fin de l'exécution du programme suivant ?

```
1 a=1
2 for i in range(3):
3     a=a*2
```

a. ☐

6 Que vaut la variable *s* à la fin de l'exécution du programme suivant ?

```
1 s=0
2 for i in range(4):
3     s=s+i
```

a. ☐ 4 b. ☐ 6 c. ☐ 10

Automatismes à compléter

1

```
..... i in range(4):
```

Mot-clé manquant de la boucle :

2

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

Nom du compteur de la boucle :

3

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

Type de la variable *i* :

4

```
for i in range(100):
```

Première valeur du compteur :

5

```
for i in range(2,8):
```

Boucle pour *i* allant de :

6

```
1 for i in range(4):
2     print(2*i)
```

Affichage :

Réponses du qcm : 1 B 2 A 4 C 5 C 6 B

Réponses Automatismes : 1) for 2) i 3) entier (int)
4) 0 5) 2 à 7 6) 0 ; 2 ; 4 ;6

Boucles non bornées (p.24 à p.27 du cahier d'algo)

A Définition d'une boucle non bornée

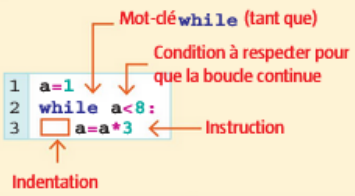
- Dans un certain nombre de cas, il est indispensable de répéter des instructions sans savoir à l'avance combien de fois on les répète. La boucle **for** est alors inefficace. On utilise un autre type de boucle : **les boucles non bornées**. Les instructions sont alors répétées **tant qu'**une condition est vérifiée. On parle ainsi de **boucle tant que** ou de **boucle while**.
- La boucle s'arrête quand la condition n'est plus vérifiée.

B Boucle « while »

- Pour répéter un ensemble d'instructions tant qu'une condition est vérifiée, la syntaxe est la suivante :

```
while condition:
    instruction1
    instruction2
    ...
```

Exemple :




```
1 a=1
2 while a<8:
3     a=a*3
```

Ligne 2 : le mot-clé **while** indique la création d'une boucle **while** et précède la condition : tant que $a < 8$.

Ligne 3 : l'instruction à répéter, a=a*3, est **indentée** (décalée vers la droite).

Ce programme multiplie la valeur de la variable a par 3 tant qu'elle est inférieure à 8. La variable a vaut donc successivement 1, 3 et 9 avant l'arrêt de la boucle.

Valeur de a ligne 2	Condition	Tour de boucle	Action ligne 3	Valeur de a ligne 3
1	1 < 8 Vrai	1	affectation a=1*3	3
3	3 < 8 Vrai	2	affectation a=3*3	9
9	9 < 8 Faux	–	–	9

C Utilisation du compteur

Il est parfois utile de compter le nombre de répétitions effectuées dans une boucle **while**. Dans ce cas, on peut définir un compteur. Ce compteur est initialisé à la valeur 0 et augmente de 1 à chaque tour de boucle.

Exemple :

```
1 a=1
2 compteur=0
3 while a<100:
4     a=a*3
5     compteur=compteur+1
```

Ligne 2, on initialise le compteur à la valeur 0.

Ligne 5, le compteur augmente de 1 à chaque tour de boucle.

À la fin de l'exécution de ce programme, le compteur est égal à 5.

Automatismes à compléter

1

```
..... a>2:
```

Mot-clé pour créer une boucle non bornée : ____

2

```
while a!=5:
```

Condition d'arrêt du while : ____

3

```
while x**2+3>2*y-1:
```

Condition d'arrêt du while : ____

4

```
1 a=3
2 while a==3:
3     a=5
4     a=6
```

Valeur de a à la fin du programme : ____

5

```
1 a=1
2 while a<10:
3     print("$")
4     a=a+4
```

Nombre de dollars affichés : ____

6

```
1 x=5
2 while x>=3:
3     x=x-1
```

Valeur de x à la fin du programme : ____

QCM Cocher la bonne réponse

1 Combien de tours de boucle sont effectués dans le programme suivant ?

```
1 x=0
2 while x<3:
3     x=x+2
```

a. ☐ 0
b. ☐ 1
c. ☐ 2

2 Quelles sont les valeurs affichées dans le programme suivant ?

```
1 a=2
2 while a!=8:
3     print(a)
4     a=a*2
```

a. ☐ 2, 4, 8
b. ☐ 1, 2, 3
c. ☐ 2, 4

3 Quelle condition faut-il écrire pour que la boucle s'arrête quand $x \geqslant 9$?

```
1 x=1
2 while .....:
3     x=x+3
```

a. ☐ x > 9
b. ☐ x < 9
c. ☐ x > =9

4 Quelle est la valeur affiché lienm

```
1 x=1
2 while x<3:
3     x=4*x
4 print(x)
```

a. ☐ 1 b. ☐ 4 c. ☐ 16

5 À quelle condition cette boucle s'arrête-t-elle ?

```
while a==5:
```

a. ☐ si a est égal à 5.
b. ☐ si a est différent de 5.
c. ☐ si a est nul.

6 Pour quelle valeur de a la boucle while fait-elle exactement 3 tours ?

```
while a<4:
    a=a+1
```

a. ☐ 0 b. ☐ 1 c. ☐ 2

Réponses du qcm : 1 C 2 C 3 B 4 B 5 B 6 B

Réponses Automatismes : 1) while 2) a = 5 3) $x^2 + 3 \leq 2y - 1$ 4) 6 5) 3 6) 2

Les bibliothèques

- Une **bibliothèque** est un **ensemble de fonctions et de constantes** (comme π) prêtes à être utilisées. Il existe une multitude de bibliothèques ayant chacune sa **thématique** propre.
- Pour en utiliser une, il faut la **charger** dans le programme. On dit alors qu'**on importe** la bibliothèque.

A Une première bibliothèque : la bibliothèque math

- La bibliothèque **math** regroupe de nombreuses **fonctions** et **constantes mathématiques**.
- La fonction nommée **sqrt** permet de calculer la racine carrée d'un nombre, la constante nommée pi donne la valeur approchée de π.

Exemple : dans la ligne 1, on importe toutes les méthodes de la bibliothèque math.
Dans la ligne 2, on affecte $\sqrt{2}$ à la variable *a*.
Dans la ligne 3, on affecte une valeur approchée de π à la variable *b*.

```
1 from math import*
2 a=sqrt(2)
3 b=pi
```

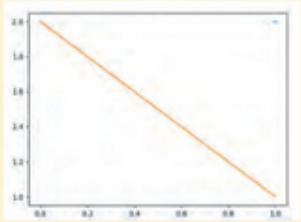
Remarque : on peut également calculer les sinus, cosinus et tangente d'un angle avec les fonctions sin, cos et tan.

B La bibliothèque matplotlib

- La bibliothèque **matplotlib** regroupe des outils pour **tracer** et **visualiser** des données sous forme de graphiques.

Exemple : dans la ligne 1, on importe le paquet **pyplot** de la bibliothèque **matplotlib**.
Dans la ligne 2, on place sur le graphique le point de coordonnées (1 ; 2). Le paramètre "+" permet d'obtenir une croix.
Dans la ligne 3, on trace le segment passant par les points de coordonnée (0 ; 2) et (1 ; 1). Notons qu'il faut mettre d'abord toutes les **abscisses** entre crochets puis toutes les **ordonnées**.
Dans la ligne 4, l'instruction show() permet d'afficher toutes les figures.

```
1 from matplotlib.pyplot import*
2 plot(1,2,"+")
3 plot([0,1],[2,1],"-")
4 show()
```



Remarque : la syntaxe pour un segment passant par les deux points de coordonnées (x_1,y_1) et (x_2,y_2) est : plot($[x_1,x_2]$,$[y_1,y_2]$,"-").

C La bibliothèque random

- La bibliothèque **random** permet de créer des **nombres aléatoires.** Elle contient en particulier la fonction random qui génère un nombre aléatoire entre 0 et 1 (1 exclu) et la fonction **randint** qui génère un **entier** aléatoire entre deux bornes (leurs valeurs incluses).

Exemple : dans la ligne 1, on importe la bibliothèque random.
Dans la ligne 2, on affecte un nombre aléatoire compris entre 0 et 1 (1 exclu) à la variable *a*.
Dans la ligne 3, on affecte un nombre aléatoire compris entre 1 et 6 (1 et 6 inclus) à la variable *b*.

```
1 from random import*
2 a=random()
3 b=randint(1,6)
```

QCM Cocher la bonne réponse

1 Pour importer une bibliothèque en Python, on peut utiliser la syntaxe :
a. ☐ from... import...
b. ☐ includes... from...
c. ☐ download... from ...

2 Lorsque la bibliothèque math est importée, pour obtenir $\sqrt{3}$, il faut utiliser l'instruction :
a. ☐ racine(3) b. ☐ sqrt(3) c. ☐ SPQR(3)

3 Une fois la bibliothèque random importée, l'instruction `randint(1,3)` simule un nombre entier dont les valeurs possibles sont :
a. ☐ 1, 2 ou 3 b. ☐ 1 ou 2 c. ☐ 2 ou 3

4 Dans le programme suivant, on a importé la bibliothèque :

```
1 from random import*
2 b=randint(1,6)
3 print(b)
```

a. ☐ random b. ☐ randint c. ☐ print

Automatismes à compléter

1
```
1 from math .....*
```
Mot-clé manquant : ☐

2
```
1 from matplotlib import*
2 plot(1,2,"+")
```
Que représente-t-on ?

3
```
1 from ..... import*
2 a=randint(1,4)
```
Nom de la bibliothèque :

4
```
1 from math import*
2 r=sqrt(2)
```
Que fait sqrt ?

5
```
1 from random import*
2 randint(4,8)
```
Que simule cette instruction ?

6
```
1 from random import*
2 print(random())
```
Qu'affiche le programme ?

Réponses du qcm : 1 A 2 B 3 A 4 A

Réponses Automatismes : 1) import 2) Le signe + au point de coordonnées (1 ; 2) 3) random 4) sqrt calcule la racine carrée 5) Un nombre entier aléatoire entre 4 et 8 compris 6) Un nombre flottant compris entre 0 et 1 exclu.

(p.12 à p.15 du cahier d'algo)