Prérequis pour entrer en première NSI

Cette fiche teste les acquis de seconde (SNT et algorithmique en mathématiques) indispensables pour aborder la première NSI sereinement. Essaie chaque exercice **sans ton cours**, note tes réponses au brouillon, puis déplie la solution correspondante pour te corriger. Si tu bloques, c'est le signe qu'il faut reprendre le point correspondant avant d'aller plus loin.

Réussi sans hésitation : passe au suivant. **Réussi avec hésitation** : fais deux exercices supplémentaires pour consolider. **Échoué** : retravaille la leçon correspondante avant de poursuivre.

① Types de base et opérateurs

Exercice 1 *Questions*

1. Sans exécuter le code, donner le type et la valeur de chaque expression :

```
a = 7 // 2
b = 7 / 2
c = 7 % 2
d = "3" + "4"
e = int("3") + int("4")
f = 2 ** 10
```

2. Expliquer en une phrase la différence entre les opérateurs `/`, `//` et `%`.
3. Prédire le résultat de `type(3.0) == type(3)`. Justifier.

② Variables et affectation

Exercice 2 *Questions*

1. Quelle est la différence entre = et == en Python ?
2. Compléter le tableau d'état des variables après l'exécution de chaque ligne :

```
a = 3
b = 5
a = b
b = a + 2
```

Ligne	a	b
après `a = 3`		
après `b = 5`		
après `a = b`		
après `b = a + 2`		

3. Parmi ces noms de variables, lesquels sont valides en Python : `2var`, `ma_variable`, `mon-nom`, `maValeur`, `_temp`, `for` ?

③ Structures conditionnelles

Exercice 3 *Questions*

1. Écrire une fonction `valeur_absolue(x)` qui renvoie $|x|$ sans utiliser `abs()`.
2. Sans exécuter le code, déterminer la valeur affichée :

```
x, y = 5, 0
print(not (x > 3 and y > 0))
```

3. Écrire une fonction `mention(note)` qui renvoie `"TB"` si la note est supérieure ou égale à 16, `"B"` si elle est supérieure ou égale à 14, `"AB"` si elle est supérieure ou égale à 12, `"Admis"` si elle est supérieure ou égale à 10, et `"Refusé"` sinon.
4. Écrire une fonction `est_bissextile(annee)` qui renvoie `True` si l'année est bissextile (divisible par 4 et non divisible par 100, sauf si elle est divisible par 400).

④ Boucles

Exercice 4 *Questions*

1. Écrire une fonction `somme_entiers(n)` qui renvoie $1 + 2 + \cdots + n$ à l'aide d'une boucle `for`.
2. Écrire une fonction `puissance(x, n)` qui renvoie x^n à l'aide d'une boucle `while` (sans utiliser `**`).
3. Compléter le tableau d'état suivant pour `somme_entiers(4)` :

Itération	`i`	Calcul	`total`
init	–	–	
1			
2			
3			
4			

4. Quand choisit-on `for` plutôt que `while` ?

⑤ Fonctions

Exercice 5 *Questions*

1. Écrire une fonction `est_pair(n)` qui renvoie `True` si `n` est pair, `False` sinon.
2. Sans exécuter le code, prédire ce qui est affiché :

```
def mystere(x):
    print(x * 3)

resultat = mystere(4)
print(resultat)
```

3. Quelle est la différence entre `return` et `print` dans une fonction ?
4. Écrire une fonction `aire_rectangle(longueur, largeur)` qui renvoie l'aire d'un rectangle, et l'utiliser pour afficher l'aire d'un rectangle de 7 sur 3.

6 Chaînes de caractères et listes

Exercice 6 *Questions*

1. Soit `s = "algorithme"`. Donner les valeurs de `s[0]`, `s[-1]`, `s[2:5]`, `len(s)`.
2. Soit `L = [7, 3, 9, 1, 5]`. Écrire le code qui affiche chaque élément de `L` sur une ligne.
3. Écrire une fonction `somme_liste(L)` qui renvoie la somme des éléments d'une liste de nombres.
4. Que vaut `L` après l'exécution de ces lignes ?

```
L = [1, 2, 3]
L.append(4)
L[0] = 10
```

7 Représentation binaire

Exercice 7 *Questions*

1. Convertir 13 en base 2.
2. Convertir 1011_2 en base 10.
3. Sur 8 bits, quel est le plus grand entier positif représentable (non signé) ?
4. Combien de valeurs différentes peut-on représenter avec n bits ? En déduire combien de couleurs distinctes on peut coder avec 24 bits (8 bits par canal RGB).

Solutions des exercices

Corrigé de l'exercice 1.

1. `a = 3` (`int`, division entière) ; `b = 3.5` (`float`, division réelle) ; `c = 1` (`int`, reste de la division) ; `d = "34"` (`str`, concaténation de chaînes) ; `e = 7` (`int`, addition d'entiers) ; `f = 1024` (`int`, puissance).
2. `/` est la division réelle (résultat `float`) ; `//` est la division entière, elle renvoie le quotient (résultat `int`) ; `%` renvoie le reste de la division euclidienne.
3. `False` : `3.0` est de type `float`, `3` est de type `int`. Même si les valeurs sont égales (`3.0 == 3` vaut `True`), leurs types sont distincts.

Si tu échoues : reprends la distinction entre `int` et `float`, le rôle des trois opérateurs de division (`/`, `//`, `%`), et la différence entre concaténation de chaînes et addition numérique. Ces réflexes sont indispensables dès la première semaine de première. Fiche à revoir : Les types de base.

Corrigé de l'exercice 2.

1. `=` est l'*affectation* : on stocke une valeur dans une variable (instruction). `==` est la *comparaison d'égalité* : on teste si deux valeurs sont égales (expression qui renvoie `True` ou `False`).
2. Tableau d'état :

Ligne	`a`	`b`
après `a = 3`	3	non définie
après `b = 5`	3	5
après `a = b`	5	5
après `b = a + 2`	5	7

3. Valides : `ma_variable`, `maValeur`, `_temp`. Invalides : `2var` (commence par un chiffre : un nom doit commencer par une lettre ou `_`), `mon-nom` (le tiret est interprété comme soustraction), `for` (mot-clé réservé du langage).

Si tu échoues : reprends la notion d'affectation (on écrit dans une case mémoire, l'ordre des lignes compte), la différence syntaxique `=` vs `==`, et les règles de nommage (commence par une lettre ou `_`, que des caractères alphanumériques ou `_`, pas de mot-clé réservé). Fiche à revoir : Variables et affectation.

Corrigé de l'exercice 3.

1.
```
def valeur_absolue(x):
    if x >= 0:
        return x
    else:
        return -x
```

2. `x > 3` vaut `True`, `y > 0` vaut `False`. `True and False` vaut `False`. `not False` vaut `True`. Affichage : `True`.

3.

```python
def mention(note):
    if note >= 16:
        return "TB"
    elif note >= 14:
        return "B"
    elif note >= 12:
        return "AB"
    elif note >= 10:
        return "Admis"
    else:
        return "Refusé"
```

4.

```python
def est_bissextile(annee):
    return (annee % 4 == 0 and annee % 100 != 0) or annee % 400 == 0
```

On vérifie : 2024 (divisible par 4 et non par 100) est bissextile ; 1900 (divisible par 100 mais pas par 400) ne l'est pas ; 2000 (divisible par 400) l'est.

Si tu échoues : reprends l'évaluation des opérateurs booléens `and`, `or`, `not` (table de vérité), l'ordre des tests dans une chaîne `if/elif/else` (le premier test vrai déclenche la branche correspondante), et la composition de conditions avec les parenthèses. Fiche à revoir : Structures conditionnelles.

Corrigé de l'exercice 4.

1.

```python
def somme_entiers(n):
    total = 0
    for i in range(1, n + 1):
        total += i
    return total
```

2.

```python
def puissance(x, n):
    resultat = 1
    while n > 0:
        resultat *= x
        n -= 1
    return resultat
```

3. Tableau d'état pour `somme_entiers(4)` :

Itération	`i`	Calcul	`total`
init	–	–	0
1	1	$0+1$	1
2	2	$1+2$	3
3	3	$3+3$	6
4	4	$6+4$	10

On vérifie avec la formule de Gauss : $\frac{4 \times 5}{2} = 10$. ✓

4. On choisit `for` quand le nombre d'itérations est connu à l'avance (parcours d'une plage d'entiers, d'une liste, d'une chaîne). On choisit `while` quand l'arrêt dépend d'une condition qu'on ne peut pas traduire en nombre fixe d'itérations (par exemple, chercher quand un calcul dépasse un seuil, attendre une entrée valide de l'utilisateur).

Si tu échoues : reprends la construction `range(a, b)` (qui s'arrête à $b-1$), le rôle de l'accumulateur initialisé avant la boucle, et le choix `for`/`while`. Écrire un tableau d'état à la main est le meilleur moyen de comprendre une boucle avant d'en écrire une. Fiche à revoir : Les boucles.

Corrigé de l'exercice 5.

1.
```
def est_pair(n):
    return n % 2 == 0
```

2. Le programme affiche `12` (par le `print` à l'intérieur de la fonction), puis `None` (la fonction ne contient pas de `return`, donc `resultat` vaut `None`).
3. `return` renvoie une valeur exploitable par l'appelant (qu'on peut stocker dans une variable, réutiliser dans un calcul) ; `print` affiche à l'écran et ne renvoie rien (`None`). Écrire `print` là où il faudrait `return` fait perdre le résultat.

4.
```
def aire_rectangle(longueur, largeur):
    return longueur * largeur

print(aire_rectangle(7, 3))  # affiche 21
```

Si tu échoues : reprends la syntaxe `def nom(parametres):`, la notion de paramètre (variable locale à la fonction) et la distinction fondamentale `return`/`print`. Une fonction sans `return` renvoie implicitement `None` : c'est la source la plus fréquente de bugs en début de première. Fiche à revoir : Les fonctions.

Corrigé de l'exercice 6.

1. `s[0] = "a"` ; `s[-1] = "e"` (dernier caractère) ; `s[2:5] = "gor"` (indices 2, 3, 4, le 5 est exclu) ; `len(s) = 10`.

2.

```
L = [7, 3, 9, 1, 5]
for x in L:
    print(x)
```

On peut aussi écrire `for i in range(len(L)): print(L[i])`, mais la première forme est plus idiomatique.

3.
```
def somme_liste(L):
    total = 0
    for x in L:
        total += x
    return total
```

4. `L = [10, 2, 3, 4]` : `append(4)` ajoute 4 en fin de liste, puis `L[0] = 10` remplace le premier élément.

Si tu échoues : reprends l'indexation à partir de 0, *l'indexation négative* (`-1` *désigne le dernier*), *la tranche* `[a:b]` (*du a inclus au b exclu*), *la fonction* `len()`, *et le parcours direct* `for x in L`. *Ces constructions seront utilisées dans presque chaque exercice de première NSI. Fiches à revoir :* Les chaînes de caractères, Les listes.

Corrigé de l'exercice 7.

1. $13 = 8 + 4 + 1 = 2^3 + 2^2 + 2^0$, donc $13 = 1101_2$.
2. $1011_2 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 8 + 0 + 2 + 1 = 11$.
3. Sur 8 bits non signés, on représente les entiers de 0 à $2^8 - 1 = 255$.
4. Avec n bits, on représente 2^n valeurs distinctes. Avec 24 bits, on obtient $2^{24} = 16\,777\,216$ couleurs, soit environ 16,7 millions (ce qu'on appelle les couleurs « vraies »).

Si tu échoues : reprends l'écriture en base 2 (*décomposition en puissances de* 2) *et la formule fondamentale : n bits codent* 2^n *valeurs. Cette formule est à la racine de toutes les questions de codage* (*entiers, caractères, couleurs, sons*) *que tu rencontreras en première NSI. Fiche à revoir :* Représentation des données.