Partie 1

Fiche d'exercices : Les fonctions en Python

Classe : 1ère NSI
Thème : Les fonctions

1.1 Comprendre et appeler des fonctions

1.1.1 Lecture de code

On définit la fonction suivante :

```
def double(x):
    return x * 2
```

a) Que renvoie `double(5)` ?

b) Que renvoie `double(12)` ?

c) Que renvoie `double(-3)` ?

1.1.2 Fonctions avec plusieurs paramètres

On définit la fonction suivante :

```
def diviser(a, b):
    return a / b
```

a) Que renvoie `diviser(10, 2)` ?

b) Que renvoie `diviser(15, 3)` ?

c) Que renvoie `diviser(20, 4)` ?

d) Écrire l'instruction qui permet de calculer $100 \div 5$ avec cette fonction.

1.2 Écrire des fonctions simples

1.2.1 Première fonction

Écrire une fonction `triple(x)` qui renvoie le triple d'un nombre.

Exemple : `triple(4)` doit renvoyer 12

1.2.2 Fonction de soustraction

Écrire une fonction `difference(a, b)` qui renvoie la différence entre a et b (c'est-à-dire a - b).

Exemple : `difference(10, 3)` doit renvoyer `7`

1.3 Fonctions mathématiques

1.3.1 Fonction affine

(a) Écrire une fonction `f(x)` qui calcule la fonction mathématique f(x) = 2x + 5

Exemple : `f(3)` doit renvoyer `11`

(b) Écrire alors une fonction `celsius_vers_fahrenheit(celsius)` qui convertit une température de Celsius en Fahrenheit. **Formule** : F = C × 9/5 + 32

1.3.2 Fonction du second degré

Écrire une fonction `g(x)` qui calcule la fonction mathématique $g(x) = x^2 - 3x + 1$

Exemple : `g(2)` doit renvoyer `-1`

1.3.3 Fonction mystère

Quelle est la fonction mathématique correspondant au code suivant ?

```
def mystere(x):
    return 3 * x**2 - 2 * x + 5
```

Calculer `mystere(2)` à la main, puis vérifier avec Python.

1.3.4 Périmètre d'un rectangle

Écrire une fonction `perimetre_rectangle(longueur, largeur)` qui calcule le périmètre d'un rectangle.

Rappel : Périmètre = 2 × (longueur + largeur)

Exemple : `perimetre_rectangle(5, 3)` doit renvoyer `16`

1.4 Fonctions avec chaînes de caractères

1.4.1 Lecture de code avec chaînes

On définit la fonction suivante :

```python
def repete(mot):
    return mot + mot
```

a) Que renvoie `repete("bon")` ?

b) Que renvoie `repete("123")` ?

c) Que renvoie `repete("A")` ?

1.4.2 Concaténation

Écrire une fonction `bonjour(prenom)` qui renvoie la chaîne “Bonjour” suivie du prénom.

Exemple : `bonjour("Alice")` doit renvoyer `"Bonjour Alice"`

1.4.3 Répétition

Écrire une fonction `triple_mot(mot)` qui renvoie le mot répété trois fois.

Exemple : `triple_mot("ha")` doit renvoyer `"hahaha"`

1.4.4 Encadrement

Écrire une fonction `encadre(texte)` qui renvoie le texte encadré par des crochets.

Exemple : `encadre("Python")` doit renvoyer `"[Python]"`

1.5 Fonctions avec conditions

1.5.1 Test de parité

Écrire une fonction `est_pair(n)` qui renvoie `True` si le nombre est pair, et `False` sinon.

Rappel : Un nombre est pair si le reste de sa division par 2 est égal à 0 (utiliser l’opérateur %)

Exemple : `est_pair(4)` doit renvoyer `True`

1.5.2 Maximum de deux nombres

Écrire une fonction `maximum(a, b)` qui renvoie le plus grand des deux nombres.

Exemple : `maximum(5, 8)` doit renvoyer `8`

1.5.3 Valeur absolue

Écrire une fonction `valeur_absolue(x)` qui renvoie la valeur absolue d'un nombre.

Rappel : La valeur absolue de x est x si $x \geq 0$, et -x si $x < 0$

Exemple : `valeur_absolue(-5)` doit renvoyer `5`

1.6 Fonctions plus complexes

1.6.1 Moyenne de trois notes

Écrire une fonction `moyenne(note1, note2, note3)` qui calcule la moyenne de trois notes.

Exemple : `moyenne(12, 15, 9)` doit renvoyer `12.0`

1.6.2 Nombre de secondes

Écrire une fonction `en_secondes(heures, minutes, secondes)` qui convertit un temps donné en heures, minutes et secondes en un nombre total de secondes.

Exemple : `en_secondes(1, 2, 30)` doit renvoyer 3750 (1h = 3600s, 2min = 120s)

1.6.3 Maximum de trois nombres

Écrire une fonction `maximum_trois(a, b, c)` qui renvoie le plus grand des trois nombres.

Aide : Utiliser la fonction native `max()` ou des conditions imbriquées

Exemple : `maximum_trois(5, 12, 8)` doit renvoyer `12`

1.6.4 Divisibilité

Écrire une fonction `est_divisible(n, d)` qui renvoie `True` si n est divisible par d, et `False` sinon. Uriliser %.

Exemple : `est_divisible(15, 3)` doit renvoyer `True`
