Contrôle 

NOM et PRÉNOM :

Barème sur 29.5 points.

Partie SANS calculatrice

Exercice 1 (2 points) –
Citer le nom de 4 langages de programmation.

Résolution.
Voir le cours.

Exercice 2 (5 points) –
Après exécution des 4 lignes suivantes dans une console Python :

```
>>> a = 42
>>> a = a + 2
>>> b = '666'
>>> b = '777' + b
```

1. Quelle est la valeur désignée par le nom `a` ?
2. Quel est le type de la valeur désignée par le nom `a` ?
3. Quelle est la valeur désignée par le nom `b` ?
4. Quel est le type de la valeur désignée par le nom `b` ?
5. On ajoute une ligne après les 4 lignes précédentes. La console est maintenant :

```
>>> a = 42
>>> a = a + 2
>>> b = '666'
>>> b = '777' + b
>>> c = a + b
```

   Que se passe-t-il avec cette dernière ligne ?

Résolution.
Après exécution des 4 lignes suivantes dans une console Python :

```
>>> a = 42 # a a maintenant pour valeur 42
>>> a = a + 2 # a a maintenant pour valeur 44
>>> b = '666' # b a maintenant pour valeur '666'
>>> b = '777' + b # b a maintenant pour valeur '777666'
```

1. Valeur désignée par le nom `a` : 44
2. Type de la valeur désignée par le nom `a` : int
3. Valeur désignée par le nom `b` : '777666'
4. Type de la valeur désignée par le nom `b` : str
5. On ajoute une ligne après les 4 lignes précédentes. La console est maintenant :

```
>>> a = 42
>>> a = a + 2
>>> b = '666'
>>> b = '777' + b
>>> c = a + b
```

Avec cette dernière ligne, une erreur est déclenchée car on essaie d'additionner la valeur d'un int et la valeur d'un str, ce qui n'a pas de sens a priori.

Exercice 3 (3 points) –
Traduire en langage Python le pseudocode ci-dessous :

```
s ← 0
Pour k prenant les valeurs entières de 5 à 12:
    s ← s + 3k
```

Résolution.
Une traduction Python :

```
s = 0
for k in range(5,13):
    s = s + 3*k
```

Exercice 4 (4 points) –
Après exécution des lignes suivantes dans une console Python :

```
>>> gaston = "lagaffe"
>>> lagaffe = "gaston"
>>> gaston = gaston + "gaston"
>>> fantasio = lagaffe
>>> espadrille = gaston
```

1. Quelle est la valeur étiquetée `lagaffe` ?
2. Quelle est la valeur étiquetée `gaston` ?
3. Quelle est la valeur étiquetée `espadrille` ?
4. Quelle est la valeur étiquetée `fantasio` ?

Résolution.

```
>>> gaston = "lagaffe"  # gaston a maintenant pour valeur "lagaffe"
>>> lagaffe = "gaston" # lagaffe a maintenant pour valeur "gaston"
>>> gaston = gaston + "gaston" # gaston a maintenant pour valeur "lagaffegaston"
>>> fantasio = lagaffe # fantasio a maintenant pour valeur "gaston"
>>> espadrille = gaston # espadrille a maintenant pour valeur "lagaffegaston"
```

On a donc :

1. la valeur étiquetée `lagaffe` : ”gaston”
2. la valeur étiquetée `gaston` : ”lagaffegaston”
3. la valeur étiquetée `espadrille` : ”lagaffegaston”
4. la valeur étiquetée `fantasio` : ”gaston”

Exercice 5 (2.5 points) –
On exécute dans une console python les lignes suivantes :

```
1 a = 5
2 b = 2*a
3 a = a + b
4 b = b - a
5 a = 2*b + a
```

Compléter le tableau suivant qui indique les valeurs d'étiquettes `a`, `b` après exécution de l'instruction rappelée à gauche :

ligne de l'instruction	instruction	valeur de a	valeur de b
1	`a = 5`		
2	`b = 2*a`		
3	`a = a + b`		
4	`b = b - a`		
5	`a = 2*b + a`		

Résolution.
Le tableau des valeurs successives :

ligne de l'instruction	instruction	valeur de a	valeur de b
1	`a = 5`	5	/
2	`b = 2*a`	5	10
3	`a = a + b`	15	10
4	`b = b - a`	15	−5
5	`a = 2*b + a`	5	−5

Exercice 6 (1 point) –
On écrit le code Python ci-dessous dans un fichier .py :

```
for j in range(0, 5):
    s = s + 3
```

L'exécution de ce script déclenche une erreur. Pourquoi ?

Résolution.
Lors du premier passage dans la boucle, l'instruction `s = s + 3` présente à droite du symbole d'affectation le nom d'une variable (s) non encore affectée.

Exercice 7 (2 points) –
On considère la série d'instructions suivantes, écrites en pseudocode :

```
1  s ← 0
2  s ← s + 5
3  s ← s + 5
4  s ← s + 5
5  s ← s + 5
6  s ← s + 5
7  s ← s + 5
8  s ← s + 5
9  s ← s + 5
10 s ← s + 5
11 s ← s + 5
```

Proposer une réécriture plus concise en pseudocode en utilisant une boucle `Pour`.

Résolution.
Un pseudocode avec boucle :

```
s ← 0
Pour j prenant les valeurs entières de 1 à 10:
    s ← s + 5
```

Exercice 8 (3 points) –
On considère la série d'instructions suivantes, écrites en pseudocode :

```
s ← 0
s ← s + 1
s ← s + 2
s ← s + 3
s ← s + 4
s ← s + 5
s ← s + 6
s ← s + 7
s ← s + 8
s ← s + 9
s ← s + 10
```

Proposer une réécriture plus concise en pseudocode en utilisant une boucle `Pour`.

Résolution.
Un pseudocode avec boucle :

```
s ← 0
Pour j prenant les valeurs entières de 1 à 10:
    s ← s + j
```

Exercice 9 (3 points) –
On considère le script Python suivant :

```
s = 0
t = 1
v = 10
for k in range(1,5):
    t = t*k
    s = s + 2*k
    v = v + t
```

Compléter les lignes du tableau suivant :

valeurs de k	valeurs de t	valeurs de s	valeurs de v
	1	0	10
1			
2			
3			
4			

Résolution.
Le tableau des valeurs successives :

valeurs de k	valeurs de t	valeurs de s	valeurs de v
	1	0	10
1	1	2	11
2	2	6	13
3	6	12	19
4	24	20	43

Exercice 10 (4 points) –
Les lignes suivantes sont écrites dans une console Python :

```
>>> a = 5
>>> b = a + 4
>>> "c" = b
>>> d = "c"
>>> 666 = r
>>> e = a + k
>>> z = "invalide"
>>> "valide" = "ok"
```

Indiquer les numéros de ligne correspondant à une affectation non valide en expliquant ce qui la rend invalide.

Résolution.

```
>>> a = 5 # VALIDE
>>> b = a + 4  # VALIDE
>>> "c" = b  # NON VALIDE
>>> d = "c"  # VALIDE
>>> 666 = r # NON VALIDE
>>> e = a + k  # NON VALIDE
>>> z = "invalide"   # VALIDE
>>> "valide" = "ok" # NON VALIDE
```

En lignes 3, 5 et 8, la non validité est due au fait qu'à gauche du symbole d'affectation, on ne trouve pas un nom de variable :

— En lignes 3 et 8, on trouve la valeur d'un objet de type str.

— En ligne 5, on trouve la valeur d'un objet de type int.

La non validité en ligne 6 est due au fait que l'on trouve à droite du symbole d'affectation le nom d'une variable non encore affectée.

Exercice 11 (bonus, hors barème) –
Réécrire le script Python suivant de façon plus synthétique à l'aide d'une boucle `for` :

```
s = 0
s = s + 3
s = 2*s + 6
s = 3*s + 9
s = 4*s + 12
s = 5*s + 15
s = 6*s + 18
```

Résolution.
Un code Python possible :

```
s = 0
for j in range(1,7):
    s = j * s + 3 * j
```

Exercice 12 (bonus, hors barème) –
Réécrire le script Python suivant de façon plus synthétique à l'aide d'une boucle `for` :

```
s = 0
s = 10*s + 9
s = 20*s + 14
s = 30*s + 19
```

```
s = 40*s + 24
s = 50*s + 29
s = 60*s + 34
s = 70*s + 39
```

Résolution.
Un code Python possible :

```
s = 0
for j in range(1,8):
    s = 10 * j * s + 5 * j + 4
```