Classement : N° — **Auditoire** : (siège) **0**

Nom : 12/04/2016 15:00

Matricule : (135 minutes)

B1030 — Informatique et communication

IC1T Programmation

Examen Avril 2016

Consignes générales

— Pas de calculatrice.
— Feuilles de brouillon incluses, à rendre avec sa copie.
— Réponses au bic ou stylo, noir ou bleu.

Mise en forme du code

— Vous pouvez ajouter des commentaires pour expliquer des parties non triviales de vos réponses.
— Faites attention au soin de vos réponses, et en particulier à rendre l'indentation explicite.
— Ne pas recopier les entêtes des fonctions déjà fournies.

Réservé aux correcteurs (questionnaire A)

Q1								/3
Q2								/4
Q3								/7
Q4								/6
Total								**/20**

Nom : 12/04/2016 15:00

Matricule : (135 minutes)

B1030 — Informatique et communication

Nom : 12/04/2016 15:00

Matricule : (135 minutes)

B1030 — Informatique et communication

1 Vrai ou faux (3 points)

(Connaissances générales)

Pour chacune des affirmations suivantes, indiquez si elle est vraie ou fausse. *Si vous avez répondu correctement à toutes les affirmations d'un bloc, vous obtenez un point, si vous avez fait une faute vous obtenez un demi-point et sinon vous avez zéro (une abstention compte comme une faute).*

1.1 Bases du Python

Affirmation	Vrai	Faux

A/ Le programme suivant affiche 42 à l'écran lors de son exécution.
`Hello = 42`
`print("Hello")`
	☐	☐

B/ On peut déclarer une variable sans lui affecter une valeur	☐	☐

C/ Le programme suivant provoque une erreur lors de son exécution.
`x = 12`
`x = y`
`print(y)`
	☐	☐

D/ La variable x initialisée par x = "12" est de type int.	☐	☐
E/ Soient les variables x et y stockant des nombres entiers valant respectivement -1 et 17, l'expression suivante vaut False. `not (x + y == 16 and x > 1)`	☐	☐

1.2 Instruction conditionnelle et itérative

Affirmation	Vrai	Faux
A/ L'expression 12 < 7 est de type booléen.	☐	☐
B/ On peut adjoindre un bloc else à l'instruction while.	☐	☐

C/ Le programme suivant est une boucle infinie.
`x = 0`
`while True:`
`  if x == 0:`
`  break`
	☐	☐

D/ On n'est pas obligé de déclarer un bloc else lorsqu'on a un bloc if.	☐	☐

E/ Le programme suivant affiche 42 lors de son exécution.
`x = 4`
`if x == 4:`
`  print(x + "2")`
	☐	☐

Nom : 12/04/2016 15:00

Matricule : (135 minutes)

B1030 — Informatique et communication

1.3 Séquence

Affirmation	**Vrai**	**Faux**

A/ Le programme suivant affiche tic-tac à l'écran lors de son exécution.
`word = "tic-tic"`
`word[5:6] = ["a"]`
`print(word)`
	☐	☐

B/ Soit une liste stockée dans une variable data. On peut afficher la longueur de la liste à l'écran avec l'instruction suivante. `print(len(data))`	☐	☐
C/ Soit une liste stockée dans une variable data. L'instruction suivante affiche toujours à l'écran le premier élément de la liste. `print(data[0])`	☐	☐

D/ Soit une liste de nombres entiers stockée dans une variable data. Le programme suivant affiche le produit des éléments de la liste.
`product = 0`
`for elem in data:`
`  product *= elem`
`print(product)`
	☐	☐

E/ Le programme suivant affiche 10 lors de son exécution.
`sum = 0`
`for i in range(1, 5):`
`  sum += i`
`print(sum)`
	☐	☐

Nom : 12/04/2016 15:00

Matricule : (135 minutes)

B1030 — Informatique et communication

2 Tailles de mots (4 points) (Niveau élémentaire)

Définir une fonction qui prend une liste de chaines de caractères en paramètre et qui renvoie une liste contenant les tailles de ces chaines de caractères.

Par exemple, l'appel `sizes(["je", "programme", "en", "python"])` *renvoie* `[2, 9, 2, 6]`.

```
def sizes(strlist) :
```

Nom : 12/04/2016 15:00

Matricule : (135 minutes)

B1030 — Informatique et communication

3 Construction de mots (7 points) (Niveau moyen)

Définir une fonction qui prend une liste de chaines de caractères en paramètre et qui renvoie une liste de chaines de caractères dont le i^{e} élément est composé en concaténant les i^{e} caractère des chaines de la liste de départ (s'ils existent).

Par exemple, l'appel `decrypt(["Cbtp", "éore", "dièu", "rts", "i", "c"])` *renvoie* `["Cédric", "boit", "très", "peu"]`.

```
def decrypt(strlist) :
```

Nom : 12/04/2016 15:00

Matricule : (135 minutes)

B1030 — Informatique et communication

4 PGCD (6 points)

(Niveau avancé)

Définir une fonction récursive qui prend deux nombres entiers positifs a et b en paramètres et renvoie le plus grand commun diviseur entre a et b en se basant sur les propriétés récursives suivantes :

$$\begin{cases} pgcd(a,b) = pgcd(b,a) \\ pgcd(a,b) = pgcd(b, a\%b) \quad (\text{si } a > b \text{ et } b \neq 0) \\ pgcd(a,0) = a \end{cases}$$

```
def pgcd(a, b) :
```

Nom : 12/04/2016 15:00

Matricule : (135 minutes)

B1030 — Informatique et communication

Brouillon

Nom : 12/04/2016 15:00

Matricule : (135 minutes)

B1030 — Informatique et communication

(brouillon suite)

Nom : 12/04/2016 15:00

Matricule : (135 minutes)

B1030 — Informatique et communication

(brouillon suite)

Cheat Sheet

Programmation Python : Les bases

Général

`# commentaire`	insère un commentaire dans le programme
`print(v)`	affiche la valeur `v` à l'écran
`input("Phrase")`	affiche `Phrase` à l'écran et renvoie la phrase tapée à la console

Opérateurs arithmétiques

`+`	addition	`-`	opposé ou soustraction
`*`	multiplication	`/`	division
`**`	exponentiation		
`//`	division entière	`%`	reste de la division entière

Opérateurs de comparaison

`==`	égal	`!=`	différent
`<`	strictement plus petit	`>`	strictement plus grand
`<=`	plus petit ou égal	`>=`	plus grand ou égal

Opérateurs logiques

`not`	*« non »* logique		
`and`	*« et »* logique	`or`	*« ou »* logique

Conversion de données

`type(v)`	renvoie le type de `v`
`int(v)`	convertit `v` en nombre entier
`float(v)`	convertit `v` en nombre flottant
`complex(v)`	convertit `v` en nombre complexe
`bool(v)`	convertir `v` en booléen
`str(v)`	convertit `v` en chaine de caractères

Instructions conditionnelles

`if` *1re condition*`:` *1er bloc de code*	Exécute le *1er bloc de code* si la *1re condition* vaut `True`,
`elif` *2e condition*`:` *2e bloc de code*	sinon exécute le *2e bloc de code* si la *2e condition* vaut `True`
`else:` *3e bloc de code*	et exécute le *3e bloc de code* sinon

Instruction répétitive

`while` *condition*`:` *1er bloc de code*	répète l'exécution du *bloc de code* tant que la *condition* vaut `True`,
`else:` *2e bloc de code*	et exécute le *2e bloc de code* lorsqu'elle vaut `False`
`break`	interrompt l'exécution de la boucle
`continue`	interrompt l'itération courante de la boucle

Importation

`import` *n*	importe le module *n*
`from` *n* `import` *m*	importe *m* depuis le module *n*

Fonction

`def` *n*$(p_1, \ldots, p_r)$`:` *bloc de code*	définit une fonction *n* avec les *r* paramètres positionnels $p_1, \ldots, p_r$
`def` *n*$(p_1{=}v_1, \ldots, p_r{=}v_r)$`:` *bloc de code*	définit une fonction *n* avec les *r* paramètres optionnels $p_1, \ldots, p_r$ avec comme valeurs par défaut $v_1, \ldots, v_r$
`return v`	quitte la fonction en renvoyant la valeur `v`
`global v`	indique que `v` réfère une variable globale

Séquence

`len(s)`	taille de `s`
`s[i]`	élément d'indice `i` de `s` (premier indice : 0)
`del(s[i])`	supprime l'élément d'indice `i` de `s`
`s[i:j]`	sous-séquence de `i` (inclus) à `j` (exclu) *(on peut omettre les bornes `i` et/ou `j`)*
`s + t`	concatène les deux séquences `s` et `t`
`s * n`	répète `n` fois la séquence `s`
`v in s`	vaut `True` si `v` se trouve dans `s`, `False` sinon
`range(n, m)`	intervalle d'entiers de `n` (inclus) à `m` (exclu)
`for e in s:` *bloc de code*	exécute le *bloc de code* pour chaque élément `e` de `s`

Nom :
Matricule :

B1030 — Informatique et communication

Classement : N° **— Auditoire :** (siège) **0**

12/04/2016 15:00
(135 minutes)