

BERRY GÉRARD

Consignes

- Lire attentivement la question et les réponses avant de **noircir la case complètement au crayon bic**. En cas d'erreur, utiliser du blanc et reformer le contour de la case proprement. La lecture automatique sera moins certaine.
- Les questions marquées du symbole ★ comportent zéro, une ou plusieurs réponses correctes. Les autres questions n'en comportent qu'une seule.

1 Types

Question 1 Quel est le type de `False` en Python ?

- int byte str bool char

Question 2 Si `a` est de type `int`, quel est le type de `3.3*a` en Python ?

- str bool int float
 On ne peut pas conclure sur le type de cette expression

Question 3 Quel est le type de `9.54` en Python ?

- int byte double float real

Question 4 Quel est le type de `"3.1415926"` en Python ?

- real str double int float

Question 5 ★ En Python, parmi les types suivants, lesquels sont des types composés ?

- tuple int list bool str
 Aucune de ces réponses n'est correcte.

Question 6 Quel est le type de `a+c*b` en Python ?

- str int float bool
 Cela dépend des types de `a`, `b` et `c`

Question 7 Quel est le type de `True` en Python ?

- bool const double int str

Question 8 La règle d'or Python est qu'une variable est une :

- valeur d'un type quelconque
 liste
 référence contenant l'adresse d'un objet en mémoire
 valeur flottante
 valeur booléenne

CORRECTION

Question 9 ★ Parmi les éléments suivants, quels sont les types de données possibles en langage Python ?

- bool "string" str chain complex True
 Aucune de ces réponses n'est correcte.

Question 10 ★ Parmi les éléments suivants, quels sont les types de données possibles en langage Python ?

- int byte const float real double
 Aucune de ces réponses n'est correcte.

Question 11 ★ En Python, parmi les types suivants, lesquels sont des types simples ?

- list bool int dict tuple
 Aucune de ces réponses n'est correcte.

Question 12 ★ Python est un langage à typage :

- implicite dynamique transtypique explicite
 statique Aucune de ces réponses n'est correcte.

2 Opérateurs

Question 13 Que vaut c après ces instructions ? a = 42; b =8; c = a % b

- 2 1 14 0 7

Question 14 Que vaut a après ces instructions ? a = 1; a -= 3

- 1 -2 -3 -1 2

Question 15 Que vaut l'expression not False or False ?

- Right Wrong Maybe False True

Question 16 Quel est le type de l'expression "Inform"+"atique" ?

- bool str int chain real

Question 17 Quelle est l'opération effectuée par l'opérateur // ?

- Division Modulo Partie entière Ratio
 Division entière

Question 18 Que vaut a après ces instructions ? a = True; a = not a

- Wrong True False Maybe Right

Question 19 Quel est le type de l'expression 3 + 4.5 ?

- real bool float integer int

Question 20 Quel est le résultat de : "Python - "*2 ?

- "Python - Python - " "Python"%2 "Python - Python"
 "Python"***2 "Python*2"

CORRECTION

Question 21 Comment appelle-t-on l'opération `a += 1` ?

- association expression transcription
 incrémentation et affectation fonction incrémentale

Question 22 Que vaut `a` après ces instructions ? `a = "Hey "`; `b="Jude"`; `a = a+b`

- False 8 str "ab" "Hey Jude"

Question 23 Quelle est l'opération effectuée par l'opérateur `%` ?

- Division entière Division Ratio Modulo
 Partie entière

Question 24 Que vaut `c` après ces instructions ? `a = 21`; `b = 7`; `c = a % b`

- 1 14 2 0 7

Question 25 Que vaut l'expression `False and not True` ?

- Wrong False Right Maybe True

Question 26 Quel est le type de l'expression `8 // 4` ?

- real int bool str float

Question 27 Que vaut `c` après cette instruction ? `c = -1**4`

- 0 L'instruction engendre une exception de type `ValueError` 4
 -1 1

Question 28 Quel est le type de l'expression `(x<y) and (b == c)` ?

- bool float int logical str

Question 29 Que vaut `a` à la fin de ces instructions ? `a = 3`; `a *= 2`

- "aa" 9 6 "3232" float

Question 30 Comment appelle-t-on l'opération `a = 3.14` ?

- transcription expression fonction association
 affectation

Question 31 Quel est le type de l'expression `8 / 4` ?

- real bool int str float

Question 32 Que vaut `a` après ces instructions ? `a = 3`; `a = a**3`

- 9 27 "aaa" int "a3a3a3"

Question 33 Les instructions suivantes ont été exécutées : `a = 3`; `i = id(a)`; `a = 4`. Que est le résultat de `i == id(a)` ?

- True 21 id False "test"

3 Mots-clefs

Question 34 ★ Parmi ces mots, quels sont les mots-clefs Python ?

- do end while repeat for
 Aucune de ces réponses n'est correcte.

Question 35 ★ Parmi ces mots, quels sont les mots-clefs Python ?

- Wrong Other True None Right
 Aucune de ces réponses n'est correcte.

Question 36 ★ Parmi ces mots, quels sont les mots-clefs Python ?

- then end else if elif
 Aucune de ces réponses n'est correcte.

Question 37 ★ Parmi ces mots, quels sont les mots-clefs Python ?

- select use from import take
 Aucune de ces réponses n'est correcte.

4 Bibliothèques

Question 38 ★ Quelles sont les syntaxes possibles pour utiliser la fonction `array` du module `numpy` ?

- `import numpy from array; numpy.a = array([1,2,3,4])`
 `from numpy import array; a = numpy.array([1,2,3,4])`
 `from numpy import array; a = array([1,2,3,4])`
 `import numpy as np; a = np.array([1,2,3,4])`
 `import numpy; import array; a = array([1,2,3,4])`
 Aucune de ces réponses n'est correcte.

Question 39 Comment importer les fonctions `sin` et `log` du module `math` ?

- `import math.sin; import math.log`
 `import sin, log as math`
 `import sin, log from math`
 `from sin, log import math`
 `from math import sin, log`

Question 40 Comment importer la fonction `sin` du module `math` ?

- `import math.sin` `from math import sin` `import sin as math`
 `import sin from math` `from sin import math`

Question 41 ★ Quelles sont les syntaxes possibles pour utiliser la fonction `randrange` du module `random` ?

- `from random import randrange; random.randrange(10)`
 `from random import randrange; randrange(10)`
 `from random import *; random.randrange(10)`
 `import random; randrange(10)`
 `import random; random.randrange(10)`
 Aucune de ces réponses n'est correcte.

5 Programmation structurée

Question 42 Que vaut la variable `responsable` à la fin de ces instructions ?

```
import random
age = 21
responsable = False
if age < 18:
    responsable = False
elif 18 <= age < 45:
    responsable = True
else:
    responsable = random.choice([False, True])
```

- True
 42
 age
 None
 False

Question 43 Que vaut la variable `age` à la fin de ces instructions ?

```
age = 0
for k in range(1, 10, 3):
    age += k
```

- 7
 19
 22
 12
 10

Question 44 Quels mots clefs permettent de créer une structure alternative en Python ?

- `while ...`
 `if ... else..`
 `def ...`
 `for...`
 `try ... catch`

Question 45 Que vaut la variable `age` à la fin de ces instructions ?

```
age = 21
for k in range(1, 3):
    age += 2
```

- 25
 27
 19
 23
 Cette boucle ne se termine jamais.

Question 46 Choisir la bonne réponse.

```
acc = 1
n = 10.5
for k in range(n):
    acc = k*k + acc
print(acc)
```

- Ce code engendre une exception de type `TypeError`
 La valeur de `acc` aurait dû être initialisée à zéro
 La valeur de `acc` aurait dû être initialisée à -1
 Ce code affiche la valeur de `acc` sur la console

CORRECTION

Question 47 ★ Choisir les bonnes réponses.

```
age = 18
if age <= 18:
    citizen = True
elif age >= 18:
    citizen = False
else:
    citizen = False
```

- À la fin de ce script, `citizen` vaut `True`
 La structure alternative est judicieuse et logique
 La structure alternative est illogique et maladroite
 À la fin de ce script, `citizen` vaut `False`
 Aucune de ces réponses n'est correcte.

Question 48 ★ Quels mots clefs permettent de créer une structure itérative en Python?

- `try ... catch` `def ...` `if ... else..` `for...`
 `while ...` Aucune de ces réponses n'est correcte.

Question 49 En Python, l'indentation

- est significative et délimite un bloc d'instructions
 est dépourvue de sens
 signale une exception dans un bloc
 n'est pas nécessaire pour la lecture du code

Question 50 Que vaut la variable `age` à la fin de ces instructions?

```
age = 21
while age > 0:
    age -= 1
```

- Cette boucle ne se termine jamais. 0 None 1
 -1

Question 51 Choisir la bonne réponse.

```
u = 0
for k in range(4):
    u *= 2
print(u)
```

- Ce code engendre une exception de type `TypeError`
 Ce code affiche 0 sur la console
 Ce code affiche 14 sur la console
 Ce code affiche 32 sur la console
 Ce code ne termine jamais

Question 52 Que vaut la variable `age` à la fin de ces instructions?

```
age = 21.7
while age != 0:
    age -= 1
```

- 1 Cette boucle ne se termine jamais. 0 None
 -1

Question 53 ★ Que pensez vous ce programme ?

```
b = True
age = 21
while b:
    age -= 1
    if age < 0:
        b = False
```

- À la fin `age` vaut 0 Il se termine Il ne se termine jamais
 À la fin `age` vaut 1 À la fin `age` vaut -1
 Aucune de ces réponses n'est correcte.

Question 54 Choisir la bonne réponse.

```
u = 1
for k in range(5):
    u *= 2
print(u)
```

- Ce code affiche 32 sur la console
 Ce code affiche 14 sur la console
 Ce code affiche 0 sur la console
 Ce code engendre une exception de type `ValueError`
 Ce code ne termine jamais

Question 55 Que vaut la variable `age` à la fin de ces instructions ?

```
age = 21
for k in range(3):
    age -= 1
```

- 17 None Cette boucle ne se termine jamais. 18
 19

6 Fonctions

Question 56 ★ Le prototype d'une fonction Python indique :

- un brouillon de la fonction comment se servir de la fonction
 la fin de l'exécution du sucre syntaxique
 le nom de la fonction et le nom des paramètres
 Aucune de ces réponses n'est correcte.

Question 57 Dans le code ci-dessous, quel est le type retourné par la fonction ?

```
def f(a, b):
    return a + b > 0
```

- `list` `str` `bool` `float` `int`

Question 58 On définit la fonction `u` comme dans le code ci-dessous. La syntaxe correcte pour utiliser cette fonction est :

```
import math
def u(n):
    return 3 * math.sqrt(n) - 3
```

- `u` `u 5` `u_5 = u(5)` `u(5)` `u_5 = u 5`

CORRECTION

Question 59 Dans le code ci-dessous, que vaut la variable `c` ?

```
def u(n):
    u = 0
    for i in range(1, n):
        u = 3 * u + 1
    return u
```

`c = u(4)`

- 9 17 13 7 1

Question 60 Appeler une fonction qui calcule une surface `fonct` est une

- excellente idée pour l'intelligibilité du code
 idée logique qui fait gagner du temps
 idée valable pour l'intelligibilité du code
 idée logique par rapport à l'objectif de la fonction
 mauvaise idée pour l'intelligibilité du code

Question 61 Dans le code ci-dessous, quel est le type retourné par la fonction ?

```
def g(a, b):
    return a + b > 0
```

- float bool int str list

Question 62 Dans le code ci-dessous, que vaut la variable `c` ?

```
def f(a, b):
    return a + b
```

`c = f(3,4)`

- 4 0 7 1 9

Question 63 Dans le code ci-dessous, `a` et `b` sont des paramètres

```
def f(a, b):
    return a + b
```

- inductifs optionnels formels effectifs entiers

Question 64 Le mot clef `def` permet de définir :

- un nombre un paramètre une règle une variable
 une fonction

Question 65 ★ Dans le code ci-dessous, la variable `c`

```
def f(a, b):
    return a + b
c = f(3,5)
```

- se voit affecter la valeur retour de la fonction `f`
 vaut 8
 permet d'exécuter la fonction `f`
 est affectée à la fonction `f`
 se voit affecter la valeur d'entrée de la fonction `f`
 Aucune de ces réponses n'est correcte.

CORRECTION

Question 66 Dans le code ci-dessous, 3 et 5 sont des paramètres

```
def f(a, b):
    return a + b
c = f(3,5)
```

- formels entiers inductifs optionnels effectifs

Question 67 Dans le code ci-dessous, que vaut la variable c ?

```
def f():
    for i in range(3):
        return i

c = f()
```

- (0,1,2) [0,1,2] 0 1 (0,0,0)

Question 68 ★ Les paramètres d'une fonction peuvent être :

- caractériels ou négatifs évènementiels ou positifs formels ou effectifs
 naturels ou capacitifs essentiels ou inductifs
 Aucune de ces réponses n'est correcte.

7 Listes

Question 69 Si M et L sont des listes Python, quel est le type de l'objet qui résulte de l'opération M+L ?

- str float list dict int

Question 70 Que vaut la variable L à la fin de ce script ?

```
L = [0, 1, 2, 3, 4]
for i in range(len(L)):
    L[i] = L[i] + 1
```

- [5, 4, 3, 2, 1] [1, 2, 3, 4, 5] [4, 3, 2, 1, 0] []
 [0, 1, 2, 3, 4]

Question 71 ★ L'instruction `21 not in L` permet :

- de savoir si L contient 21 éléments
 de savoir si 21 est un élément de la liste L
 de savoir si 21 n'est pas un élément de la liste L
 de savoir si L ne contient pas 21 éléments
 Aucune de ces réponses n'est correcte.

Question 72 Que vaut la variable L à la fin de ce script ?

```
M = [[1, 1, 1], [-1, -1, -1]]
L = []
while len(M) > 0:
    L.append(M.pop())
```

- [-1, -1, -1, 1, 1, 1] [[-1, -1, -1], [1, 1, 1]] [[-1, 1], [-1, 1], [-1, 1]]
 [-1, -1, -1, 1, 1, 1] [[1, 1, 1], [-1, -1, -1]]

CORRECTION

Question 73 Le test `len(L)>0` permet de savoir :

- si la variable L est une liste non vide
- si la variable L est une liste non nulle
- si la variable L est une liste d'entiers
- si la variable L est une liste dont on ne peut pas calculer la longueur
- si la variable L est une liste positive

Question 74 Que vaut la variable `count` à la fin de ce script ?

```
L = [0, 1, 2, 3, 4]
count = 0
for i in range(len(L)):
    count += 1
```

- 15 10 6 5 4

Question 75 Que vaut la variable L à la fin de ce script ?

```
L = [0, 1, 2, 3, 4]
i = 0
while len(L) < 6:
    L.append(i)
```

- [5, 4, 3, 2, 1] [4, 3, 2, 1, 0, 1] [1, 2, 3, 4, 5, 6] [0, 1, 2, 3, 4]
- [0, 1, 2, 3, 4, 0]

Question 76 ★ L'instruction `e=L.pop()` permet :

- de supprimer le dernier élément de la liste L
- d'affecter à `e` le premier élément de la liste L
- de supprimer le premier élément de la liste L
- de consulter le dernier élément de la liste L
- d'affecter à `e` le dernier élément de la liste L
- de consulter le premier élément de la liste L
- Aucune de ces réponses n'est correcte.

Question 77 ★ On a exécuté le code suivant.

```
M = [1,2,3]; L = [4,5]; v = M + L; u = []
for i in range(len(M)):
    u.append(M[i])
for i in range(len(L)):
    u.append(L[i])
```

Quelles sont les affirmations exactes ?

- u et v ne désignent pas la même variable en mémoire.
- u et v désignent la même variable en mémoire.
- u et v ne possèdent pas les mêmes éléments.
- u et v possèdent les mêmes éléments. u et v possèdent des éléments différents.
- Aucune de ces réponses n'est correcte.

Question 78 Que vaut la variable L à la fin de ce script ?

```
L = [0, 1, 2, 3, 4]
while len(L) > 0:
    L.pop()
```

- [0, 1, 2, 3, 4] [1, 2, 3, 4, 5] [4, 3, 2, 1, 0] [5, 4, 3, 2, 1]
- []

CORRECTION

Question 79 Que vaut la variable L à la fin de ce script ?

```
a = 21
L = [a % i for i in range(1,5)]
```

- [1, 0, 0, 1] [1, 1, 0, 1] [1, 0, 1, 0] [0, 1, 1, 1]
 [0, 1, 0, 1]

Question 80 Que vaut la variable L à la fin de ce script ?

```
M = [[1, 2, 3], [4, 5, 6]]
L = []
for v in M:
    for e in v:
        L.append(e)
```

- [0, 2, 4, 6, 8, 10] [1, 4, 3, 2, 5, 6] [6, 5, 4, 3, 2, 1]
 [0, 1, 3, 5, 7, 9] [1, 2, 3, 4, 5, 6]

Question 81 Que vaut la variable L à la fin de ce script ?

```
L = []
a = 3
for i in range(10):
    if i % a != 0:
        L.append(i % a)
```

- [1, 2, 1, 2, 1, 2] [[1, 2], [1, 2], [1, 2]] [2, 1, 2, 1, 2, 1]
 [0, 2, 1, 0, 1, 2] [1, 0, 1, 2, 0, 2]

Question 82 Que vaut la variable L à la fin de ce script ?

```
L = [0, 1, 2, 3, 4]
for i in range(len(L)):
    L[i] = L[len(L) - 1 - i]
```

- [5, 4, 3, 4, 5] [4, 3, 2, 3, 4] [0, 1, 2, 3, 4] [4, 3, 2, 1, 0]
 []

Question 83 Que vaut la variable L à la fin de ce script ?

```
M = [[1, 2, 3], [4, 5, 6]]
L = []
for i in range(len(M)):
    L.append(M[i])
```

- [1, 2, 3, 4, 5, 6] [6, 5, 4, 3, 2, 1] [[6, 5, 4], [3, 2, 1]]
 [[1, 2, 3], [4, 5, 6]] [[1, 2],[3, 4], [5, 6]]

Question 84 Que vaut la variable L à la fin de ce script ?

```
L = [i + 1 for i in range(5)]
```

- [2, 3, 4, 5, 6] [5, 4, 3, 2, 1] [4, 3, 2, 1, 0] [1, 2, 3, 4, 5]
 [0, 1, 2, 3, 4]

Question 85 Que vaut la variable L à la fin de ce script ?

```
L = [(-1) ** i for i in range(5)]
```

- [-1, 1, 1, 1, -1] [1, -1, 1, -1, 1] [-1, 1, -1, 1, -1]
 [1, -1, -1, -1, 1] [-1, -1, 1, -1, -1]

Question 86 Que vaut la variable L à la fin de ce script ?

```
L = []
for i in range(2):
    L.append([])
    for j in range(3):
        L[i].append((-1) ** (i + j))
```

- [1, -1, 1, -1, 1, -1, 1, -1, 1] [-1, 1, -1, 1, -1, 1, -1, 1, -1]
 [[1, -1, 1], [-1, 1, -1]] [[-1, 1, -1], [1, -1, 1], [-1, 1, -1]]
 [[1, -1],[1, -1], [1, -1], [1, -1, 1]]

Question 87 L'instruction `v=["3"]` permet de créer :

- une liste comportant un élément et dont le nom de variable est v
 une liste vide à trois cases dont le nom de variable est v
 une liste comportant trois éléments et dont le nom de variable est v
 une chaîne de caractères dont le nom de variable est v

Question 88 L'instruction `L.append(3)` permet :

- d'ajouter L à la liste 3
 d'ajouter 3 à la fin de la liste L
 d'ajouter 3 cases à la liste L
 d'insérer en tête 3 à la liste L
 d'insérer 3 au milieu de la liste L

Question 89 Que vaut la variable `count` à la fin de ce script ?

```
L = [0, 1, 2, 3, 4]
count = 0
for e in L:
    count += e
```

- 4 5 6 15 10

8 Trier et rechercher

Question 90 Un tri en ligne est un algorithme de tri qui peut commencer

- le tri s'il possède un comparateur incrémental des données
 le tri s'il possède déjà l'intégralité des données
 le tri avant même sans être connecté à internet
 le tri grâce à une connexion internet
 le tri avant même d'avoir reçu l'intégralité des données

Question 91 ★ La recherche dichotomique d'un élément dans un tableau

- s'effectue sur un tableau non trié
 est plus efficace que la recherche séquentielle
 divise par deux le résultat
 s'effectue sur un tableau déjà trié
 est moins efficace que la recherche séquentielle
 Aucune de ces réponses n'est correcte.

CORRECTION

- Question 92** La recherche séquentielle d'un élément dans un tableau consiste à
- chercher l'indice de élément en testant chaque case du tableau trié
 - chercher l'élément en testant chaque case du tableau dans l'ordre des indices
 - chercher l'élément en testant chaque case du tableau dans un ordre quelconque
 - chercher l'élément en testant chaque case du tableau trié

- Question 93** Un tri comparatif est un algorithme de tri qui procède en comparant
- les indices des éléments à trier en partant du début
 - les éléments du début avec ceux de la fin
 - les éléments à trier en commençant par la fin
 - les éléments à trier deux à deux
 - les indices des éléments à trier

- Question 94** Dans le meilleur des cas, le tri par insertion est
- il n'y a pas de meilleur cas
 - plus efficace que le tri par sélection
 - moins efficace que le tri par sélection
 - impossible à comparer aux autres tris
 - aussi efficace que le tri par sélection

- Question 95** Le principe du tri par sélection est de
- chercher le plus petit élément du tableau (de droite) et de l'insérer à la fin du tableau trié (de gauche)
 - chercher la place d'un élément quelconque dans le tableau (de droite) et de l'échanger avec le plus grand élément du tableau trié (de gauche)
 - compter le nombre d'occurrences de chaque valeur entière puis de construire un nouveau tableau à partir de ce comptage
 - chercher le plus petit indice dans le tableau (de droite) et de l'échanger avec le plus grand élément du tableau trié (de gauche)
 - chercher à insérer le premier élément non trié du tableau (de droite) dans le tableau trié (de gauche) à la bonne place.

- Question 96** ★ Le tri par insertion est un tri

- comparatif, non stable, hors ligne
- comparatif, stable, en place et en ligne
- non comparatif et stable
- comparatif, non stable, en place et en ligne
- Aucune de ces réponses n'est correcte.

- Question 97** Dans le pire des cas, le tri par insertion et le tri par sélection ont une complexité
- en $O(n^2)$
 - en $O(n \log n)$
 - en $O(\log n)$
 - en $O(n)$

- Question 98** Le principe du tri par insertion est de
- chercher la place d'un élément quelconque dans le tableau (de droite) et de l'échanger avec le plus grand élément du tableau trié (de gauche)
 - chercher le plus petit élément du tableau (de droite) et de l'insérer à la fin du tableau trié (de gauche)
 - compter le nombre d'occurrences de chaque valeur entière puis de construire un nouveau tableau à partir de ce comptage
 - chercher à insérer le premier élément non trié du tableau (de droite) dans le tableau trié (de gauche) à la bonne place.
 - chercher le plus petit indice dans le tableau (de droite) et de l'échanger avec le plus grand élément du tableau trié (de gauche)

- Question 99** Un tri stable est un algorithme de tri qui
- ne préserve pas la taille des éléments dans le tableau trié
 - préserve la taille du tableau une fois trié
 - préserve la taille du tableau initialement non trié
 - préserve l'ordre initial des éléments dans le tableau trié
 - préserve l'apparence des éléments dans le tableau trié

- Question 100** ★ Un tri en place est un algorithme de tri qui
- nécessite l'allocation d'une nouvelle structure en mémoire
 - ne peut pas être directement effectué dans le tableau initial
 - peut être directement effectué dans le tableau initial
 - nécessite l'allocation de deux nouvelles structures en mémoire
 - ne nécessite pas l'allocation d'une nouvelle structure en mémoire
 - Aucune de ces réponses n'est correcte.

9 Récursivité

- Question 101** ★ Pour formuler correctement un algorithme récursif, il est nécessaire de
- prévoir une condition de continuation avec appels récursifs
 - faire des appels récursifs avec des paramètres plus proches de la condition de continuation
 - faire des appels récursifs avec des paramètres plus proches de la condition d'arrêt
 - prévoir une condition d'arrêt sans appels récursifs
 - prévoir des appels récursifs incondtionnels
 - Aucune de ces réponses n'est correcte.

- Question 102** Le code suivant

```
def fact(n):
    return n * fact(n - 1)
print(fact(3))
```

- affiche 5 sur la console
- produit une exeception de type `RecursionError`
- est exécutable uniquement si on a importé la fonction `print`
- affiche 6 sur la console
- produit un résultat de type `int` si `n` est un `int`

- Question 103** Un algorithme récursif
- s'utilise lui-même pour résoudre un problème avec des données d'entrées identiques.
 - utilise la valeur retour pour résoudre un problème avec des données de sortie identiques.
 - utilise une sous-fonction pour résoudre un problème avec des données d'entrées différentes.
 - s'utilise lui-même pour résoudre un problème avec des données d'entrées différentes.

- Question 104** En Python, on peut effectuer
- un nombre fini d'appels récursifs à cause de la taille finie de la pile d'exécution
 - un nombre infini d'appels récursifs grâce à la taille finie de la pile d'exécution
 - un petit nombre d'appels récursifs à cause de la grande taille de la pile d'exécution
 - un nombre important d'appels récursifs grâce à la petite taille de la pile d'exécution

Question 105 ★ Le code suivant

```
def fact(n):
    acc = 0
    while n > 1:
        acc *= n
        n -= 1
    return acc
print(fact(3))
```

- affiche 5 sur la console
- produit une exception de type RecursionError
- affiche 0 sur la console
- affiche 6 sur la console
- n'effectue aucun appel récursif
- Aucune de ces réponses n'est correcte.

10 Numpy

Question 106 ★ Dans le code suivant,

```
import numpy as np
L = [1, 2, 3]
v = np.array(L)
```

- v est initialisé avec les mêmes valeurs que L
- v possède trois dimensions de tailles (1,2,3)
- la variable v est de type `list` et L de type `array`
- la variable v est de type `array` et L de type `list`
- la variable v est de type `np` et L de type `tuple`
- Aucune de ces réponses n'est correcte.

Question 107 ★ Si A est une matrice sous la forme d'un type numpy array de dimension (2,2), alors il est possible d'écrire :

- A(1,2)
- `np.exp(A)`
- A{0,2}
- A**2
- A[1,2,3]
- A[1,0]
- Aucune de ces réponses n'est correcte.

Question 108 ★ Numpy procure notamment :

- un type tableau statique multidimensionnel nommé `array`
- un type tableau dynamique multidimensionnel nommé `list`
- des outils dédiés à l'algèbre linéaire et aux statistiques
- des outils dédiés aux bases de données
- une alternative libre et gratuite à Matlab (payant et très cher)
- Aucune de ces réponses n'est correcte.

Question 109 Dans le code suivant,

```
import numpy as np
A = np.zeros((3,3))
B = np.ones((4,5,6))
```

- A est un vecteur de 0 et B un matrice de 1
- A est une matrice de 0 et B un tableau tridimensionnel de 1
- A et B sont des listes de 0 et de 1
- A et B sont des variables immuables
- A est une tableau tridimensionnel de 0 et B un vecteur de 1

Question 110 ★ Numpy est :

- un logiciel pour les physiciens
- un logiciel libre
- un logiciel dédié aux mathématiciens
- une bibliothèque logicielle Python dédiée au calcul numérique et scientifique
- une bibliothèque logicielle Python dédiée aux bases de données numériques
- Aucune de ces réponses n'est correcte.

Question 111 Dans le code suivant :

```
import numpy as np
A = np.array([[1,2,3],[4,5,6]])
C = A + 1
```

- C est de type `list`
- C vaut `[[1 2 3][4 5 6][1]]`
- C vaut `[[2 3 4][5 6 7]]`
- une exception de type `TypeError` est levée
- C vaut `[[1 3 5][2 4 6]]`

Question 112 ★ Dans le code suivant :

```
import numpy as np
A = np.array([1,2,3])
B = np.array([2,3,4])
C = A + B
```

- C vaut `[1 3 5]`
- C vaut `[4 5 6]`
- le calcul de C est un exemple de calcul éléments par éléments
- C vaut `[3 5 7]`
- C est de type `array`
- C est de type `list`
- Aucune de ces réponses n'est correcte.

Question 113 ★ Dans le code suivant :

```
import numpy as np
A = np.array([1,2,3])
B = np.array([2,3,4])
C = A * B
```

- C vaut `[[1 3 5][2 4 6]]`
- C vaut `[[1 2 3][4 5 6][1]]`
- C est de type `list`
- le calcul de C est un exemple de calcul éléments par éléments
- C est un vecteur comportant trois éléments
- Aucune de ces réponses n'est correcte.

CORRECTION

Question 114 Pour utiliser Numpy, quelle est l'instruction la plus fréquemment utilisée ?

- `import numpy as calculs`
- `from numpy import matplotlib`
- `from np import numpy`
- `from matplotlib import *`
- `import numpy as np`

Question 115 ★ Dans le code suivant :

```
import numpy as np
A = np.array([[1,2],[0,1]])
B = np.array([2,3])
X = np.array([1,1])
Xp = A*X + B
```

- Xp est un vecteur colonne
- Xp vaut `[[3 2][2 3]]`
- Xp est un array de dimension (2,2)
- Xp vaut `[2 3]`
- une exception de type `TypeError` est levée
- Aucune de ces réponses n'est correcte.