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

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"

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 ?

☐ Right ■ True ■ None ☐ Other ☐ Wrong
☐ *Aucune de ces réponses n'est correcte.*

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

☐ do ■ for ☐ repeat ■ while ☐ end
☐ *Aucune de ces réponses n'est correcte.*

Question 36 ★ Une assertion

- [x] est l'évaluation d'un test à l'aide du mot-clef `assert`
- [ ] est l'évaluation récursive d'un test
- [x] une bonne pratique en programmation
- [ ] est l'évaluation d'une fonction
- [ ] peut générer une exception si la fonction est appelée
- [x] peut générer une exception si le test échoue
- [ ] *Aucune de ces réponses n'est correcte.*

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

- [x] from
- [x] import
- [ ] select
- [ ] use
- [ ] take
- [ ] *Aucune de ces réponses n'est correcte.*

Question 38 ★ Parmi ces mots, quels sont les mots-clefs Python ?

- [x] if
- [x] elif
- [ ] then
- [x] else
- [ ] end
- [ ] *Aucune de ces réponses n'est correcte.*

4 Bibliothèques

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

- [ ] `import math.sin; import math.log`
- [ ] `from sin, log import math`
- [ ] `import sin, log as math`
- [x] `from math import sin, log`
- [ ] `import sin, log from math`

Question 40 ★ Quelles sont les syntaxes possibles pour utiliser la fonction `randrange` du module `random` ?

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

Question 41 Comment importer la fonction `sin` du module `math` ?

- [ ] `import sin as math`
- [ ] `import sin from math`
- [ ] `import math.sin`
- [ ] `from sin import math`
- [x] `from math import sin`

Question 42 ★ Quelles sont les syntaxes possibles pour utiliser la fonction `array` du module `numpy` ?

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

5 Programmation structurée

Question 43 Choisir la bonne réponse.

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

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

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

☐ `while ...` ☐ `for...` ■ `if ... else..` ☐ `def ...` ☐ `try ... catch`

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

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

☐ `0` ☐ `1` ☐ `None` ■ `Cette boucle ne se termine jamais.` ☐ `-1`

Question 46 Choisir la bonne réponse.

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

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

Question 47 Que vaut la variable `age` à la fin de ces instructions ?

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

☐ 7 ■ 12 ☐ 19 ☐ 10 ☐ 22

Question 48 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])
```

☐ 42 ■ True ☐ False ☐ None ☐ age

Question 49 Que vaut la variable `age` à la fin de ces instructions ?

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

☐ Cette boucle ne se termine jamais. ■ 18 ☐ None ☐ 19
☐ 17

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

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

☐ 27 ■ 25 ☐ 23 ☐ 19
☐ Cette boucle ne se termine jamais.

Question 51 Que vaut la variable `age` à la fin de ces instructions ?

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

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

Question 52 ★ 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`
☐ À la fin de ce script, `citizen` vaut `False`
■ La structure alternative est illogique et maladroite
☐ La structure alternative est judicieuse et logique
☐ *Aucune de ces réponses n'est correcte.*

Question 53 Choisir la bonne réponse.

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

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

Question 54 ★ Que pensez vous ce programme ?

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

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

Question 55 En Python, l'indentation

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

Question 56 ★ Quels mots clefs permettent de créer une structure itérative en Python ?

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

6 Fonctions

Question 57 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)
```

- [ ] 1
- [ ] 17
- [ ] 9
- [ ] 7
- [x] 13

Question 58 Dans le code ci-dessous, quel est le type renvoyé par la fonction ?

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

- [ ] `list`
- [ ] `str`
- [ ] `int`
- [ ] `float`
- [x] `bool`

Question 59 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(5)` ☐ `u 5` ■ `u_5 = u(5)` ☐ `u` ☐ `u_5 = u 5`

Question 60 ★ Le prototype d'une fonction Python indique :

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

Question 61 ★ Dans le code ci-dessous, la variable `c`

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

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

Question 62 Appeler une fonction qui calcule une surface `fonct` est une
■ mauvaise idée pour l'intelligibilité du code
☐ excellente idée pour l'intelligibilité du code
☐ idée logique par rapport à l'objectif de la fonction
☐ idée logique qui fait gagner du temps
☐ idée valable pour l'intelligibilité du code

Question 63 Dans le code ci-dessous, `3` et `5` sont des paramètres

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

☐ entiers ■ effectifs ☐ formels ☐ inductifs ☐ optionnels

Question 64 Dans le code ci-dessous, que vaut la variable `c` ?

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

c = f()
```

☐ `(0,0,0)` ☐ `(0,1,2)` ☐ `1` ☐ `[0,1,2]` ■ `0`

Question 65 ★ Les paramètres d'une fonction peuvent être :

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

Question 66 Dans le code ci-dessous, quel est le type renvoyé par la fonction ?

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

■ bool ☐ float ☐ str ☐ list ☐ int

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

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

c = f(3,4)
```

■ 7 ☐ 0 ☐ 1 ☐ 9 ☐ 4

Question 68 Dans le code ci-dessous, `a` et `b` sont des paramètres

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

☐ entiers ☐ effectifs ☐ inductifs ■ formels ☐ optionnels

Question 69 Le mot clef `def` permet de définir :

☐ une règle ☐ un nombre ☐ une variable ☐ un paramètre
■ une fonction

7 Listes

Question 70 L'instruction `v=["3"]` permet de créer :

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

Question 71 Que vaut la variable `L` à la fin de ce script ?

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

☐ [5, 4, 3, 2, 1] ☐ [0, 1, 2, 3, 4] ☐ [2, 3, 4, 5, 6] ■ [1, 2, 3, 4, 5]
☐ [4, 3, 2, 1, 0]

Question 72 Le test `len(L)>0` permet de savoir :

- ☐ 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 un liste positive
- ■ si la variable `L` est une liste non vide
- ☐ si la variable `L` est une liste non nulle

Question 73 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])
```

☐ [[6, 5, 4], [3, 2, 1]] ■ [[1, 2, 3], [4, 5, 6]] ☐ [6, 5, 4, 3, 2, 1]
☐ [[1, 2],[3, 4], [5, 6]] ☐ [1, 2, 3, 4, 5, 6]

Question 74 Si `M` et `L` sont des listes Python, quel est le type de l'objet qui résulte de l'opération `M+L` ?

■ `list` ☐ `int` ☐ `str` ☐ `float` ☐ `dict`

Question 75 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 76 Que vaut la variable `L` à la fin de ce script ?

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

☐ [1, 1, 0, 1] ☐ [1, 0, 1, 0] ■ [0, 1, 0, 1] ☐ [1, 0, 0, 1]
☐ [0, 1, 1, 1]

Question 77 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
```

■ 5 ☐ 10 ☐ 15 ☐ 4 ☐ 6

Question 78 L'instruction `L = [] for _ in range(10)` permet :

☐ de créer une liste vide
☐ de créer une liste vide à 10 éléments
■ de créer une liste de listes vides à 10 éléments
☐ génère une exception `Out of Range`
☐ de créer une liste de listes à 9 éléments

Question 79 L'instruction `L.append(3)` permet :

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

Question 80 ★ L'instruction `e=L.pop()` permet :

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

Question 81 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)
```

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

Question 82 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] [x] [1, -1, 1, -1, 1]

Question 83 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())
```

[x] [[-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 84 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] [x] [1, 2, 3, 4, 5] [ ] [0, 1, 2, 3, 4] [ ] [] [ ] [4, 3, 2, 1, 0]

Question 85 ★ 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 possèdent pas les mêmes éléments.
[ ] `u` et `v` désignent la même variable en mémoire.
[x] `u` et `v` ne désignent pas la même variable en mémoire.
[x] `u` et `v` possèdent les mêmes éléments. [ ] `u` et `v` possèdent des éléments différents.
[ ] *Aucune de ces réponses n'est correcte.*

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

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

☐ [4, 3, 2, 1, 0] ☐ [5, 4, 3, 2, 1] ☐ [0, 1, 2, 3, 4] ■ []
☐ [1, 2, 3, 4, 5]

Question 87 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]] ☐ [2, 1, 2, 1, 2, 1] ☐ [1, 0, 1, 2, 0, 2]
■ [1, 2, 1, 2, 1, 2] ☐ [0, 2, 1, 0, 1, 2]

Question 88 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)
```

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

Question 89 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] ☐ [0, 1, 2, 3, 4] ■ [4, 3, 2, 3, 4] ☐ []
☐ [4, 3, 2, 1, 0]

Question 90 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
```

☐ 6 ☐ 15 ☐ 4 ■ 10 ☐ 5

8 Trier et rechercher

Question 91 ★ Un tri en place est un algorithme de tri qui
- ☐ nécessite l'allocation de deux nouvelles structures en mémoire
- ☐ ne peut pas être directement effectué dans le tableau initial
- ■ peut être directement effectué dans le tableau initial
- ■ ne nécessite pas l'allocation d'une nouvelle structure en mémoire
- ☐ nécessite l'allocation d'une nouvelle structure en mémoire
- ☐ *Aucune de ces réponses n'est correcte.*

Question 92 Le principe du tri par sélection 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 indice 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 élément du tableau (de droite) et de l'insérer à la fin 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 93 Un tri stable est un algorithme de tri qui
- ☐ préserve l'apparence 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
- ☐ ne préserve pas la taille des éléments dans le tableau trié

Question 94 ★ Le tri par insertion est un tri
- ☐ comparatif, non stable, hors ligne
- ☐ non comparatif et stable
- ■ comparatif, stable, en place et en ligne
- ☐ comparatif, non stable, en place et en ligne
- ☐ *Aucune de ces réponses n'est correcte.*

Question 95 ★ La recherche dichotomique d'un élément dans un tableau
- ☐ divise par deux le résultat
- ☐ s'effectue sur un tableau non trié
- ■ est plus efficace que la recherche séquentielle
- ☐ est moins efficace que la recherche séquentielle
- ■ s'effectue sur un tableau déjà trié
- ☐ *Aucune de ces réponses n'est correcte.*

Question 96 Dans le meilleur des cas, le tri par insertion est
- ☐ il n'y a pas de meilleur cas
- ☐ impossible à comparer aux autres tris
- ■ plus efficace que le tri par sélection
- ☐ moins efficace que le tri par sélection
- ☐ aussi efficace que le tri par sélection

Question 97 Un tri comparatif est un algorithme de tri qui procède en comparant
- ☐ les éléments du début avec ceux de la fin
- ☐ les indices des éléments à trier en partant du début
- ☐ les indices des éléments à trier
- ■ les éléments à trier deux à deux
- ☐ les éléments à trier en commençant par la fin

Question 98 Un tri en ligne est un algorithme de tri qui peut commencer
- ☐ le tri s'il possède un comparateur incrémental des données
- ☐ le tri avant même sans être connecté à internet
- ☐ le tri s'il possède déjà l'intégralité des données
- ■ le tri avant même d'avoir reçu l'intégralité des données
- ☐ le tri grâce à une connexion internet

Question 99 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 $O(n)$ ☐ en $O(\log n)$

Question 100 Le principe du tri par insertion est de
- ☐ chercher le plus petit élément du tableau (de droite) et de l'insérer à la fin du tableau trié (de gauche)
- ☐ 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 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 à insérer le premier élément non trié du tableau (de droite) dans le tableau trié (de gauche) à la bonne place.
- ☐ compter le nombre d'occurrences de chaque valeur entière puis de construire un nouveau tableau à partir de ce comptage

Question 101 La recherche séquentielle d'un élément dans un tableau consiste à
- ☐ chercher l'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'indice de élément en testant chaque case du tableau trié

9 Récursivité

Question 102 ★ Le code suivant

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

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

Question 103 ★ Pour formuler correctement un algorithme récursif, il est nécessaire de
- ■ faire des appels récursifs avec des paramètres plus proches de la condition d'arrêt
- ☐ faire des appels récursifs avec des paramètres plus proches de la condition de continuation
- ☐ prévoir des appels récursifs inconditionnels
- ■ prévoir une condition d'arrêt sans appels récursifs
- ☐ prévoir une condition de continuation avec appels récursifs
- ☐ *Aucune de ces réponses n'est correcte.*

Question 104 En Python, on peut effectuer

- ☐ une nombre important d'appels récursifs grâce à la petite taille de la pile d'exécution
- ☐ une nombre infini d'appels récursifs grâce à la taille finie de la pile d'exécution
- ■ une nombre fini d'appels récursifs à cause de 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

Question 105 Le code suivant

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

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

Question 106 Un algorithme récursif

- ■ s'utilise lui-même pour résoudre un problème avec des données d'entrées différentes.
- ☐ utilise une sous-fonction pour résoudre un problème avec des données d'entrées différentes.
- ☐ utilise la valeur retour pour résoudre un problème avec des données de sortie identiques.
- ☐ s'utilise lui-même pour résoudre un problème avec des données d'entrées identiques.