

Olivier Reynet  
Test d'informatique  
MPSI - PCSI

gerard.berry@cpge.ker.bzh

## 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.
- Le barème n'est pas négatif.

## 1 Types

**Question 1** Quel est le type de "3.1415926" en Python ?

☐ int    ☐ double    ☐ float    ☐ real    ☒ str

**Question 2 ★** Quels sont les types de données possibles en langage Python ?

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

**Question 3** Quel est le type de `a+c*b` en Python ?

☐ bool    ☐ int    ☒ Cela dépend des types de a, b et c    ☐ str  
☐ float

**Question 4** Quel est le type de 9.54 en Python ?

☐ int    ☒ float    ☐ double    ☐ real    ☐ byte

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

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

**Question 6** Quel est le type de `True` en Python ?

☐ int    ☒ bool    ☐ double    ☐ str    ☐ const

**Question 7** La règle d'or Python est qu'une variable est une :

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

**Question 8 ★** Quels sont les types de données possibles en langage Python ?

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

**Question 9 ★** En Python, parmi les types suivants, lesquels sont des types composés ?

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

**Question 10 ★** Python est un langage à typage :

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

## 2 Opérateurs

**Question 11** Que vaut `a` à la fin de ces instructions ? `a = 3; a *= 2`

- ☐ float    ☐ "3232"    ☐ 9    ☒ 6    ☐ "aa"

**Question 12** Quel est le résultat de : `"Python" - "*2"` ?

- ☐ "Python"%2    ☐ "Python\*2"    ☐ "Python - Python"    ☐ "Python"\*\*2  
☒ "Python - Python - "

**Question 13** Quel est le type Python du résultat de : `(x<y) and (b == c)` ?

- ☒ bool    ☐ str    ☐ float    ☐ int    ☐ logical

**Question 14** Comment appelle-t-on l'opération `a = 3.14` ?

- ☐ transcription    ☐ fonction    ☒ affectation    ☐ expression  
☐ association

**Question 15** Quel est le type Python du résultat de : `3 + 4.5` ?

- ☐ real    ☐ int    ☐ integer    ☐ bool    ☒ float

**Question 16** Quel est le résultat de : `False and not True` ?

- ☐ Right    ☐ Wrong    ☐ Maybe    ☒ False    ☐ True

**Question 17** Que vaut `a` à la fin de ces instructions ? `a = "Hey "; b="Jude"; a = a+b`

- ☒ "Hey Jude"    ☐ False    ☐ "ab"    ☐ str    ☐ 8

**Question 18** Que vaut `c` à la fin de ces instructions ? `a = 21; b = 7; c = a % b`

- ☐ 1    ☐ 2    ☐ 7    ☒ 0    ☐ 14

**Question 19** Quel est le type du résultat de : `8 / 4` ?

- ☐ str    ☒ float    ☐ int    ☐ real    ☐ bool

**Question 20** Quel est le résultat de : `not False or False` ?

- ☐ Maybe    ☐ Right    ☐ False    ☐ Wrong    ☒ True

**Question 21** Quel est le type du résultat de : `8 // 4` ?

- ☐ float    ☐ real    ☐ str    ☐ bool    ☒ int

## CORRECTION

**Question 22** Les instructions suivantes ont été exécutées : `a=3; i=id(a); a=4;`. Que est le résultat de : `i == id(a)` ?

☐ 21    ☐ "test"    ☒ False    ☐ id    ☐ True

**Question 23** Comment appelle-t-on l'instruction `a += 1` ?

☐ expression    ☐ fonction    ☐ association    ☒ affectation  
☐ transcription

**Question 24** Quelle est l'opération effectuée par l'opérateur `//` ?

☐ Modulo    ☐ Division    ☐ Ratio    ☒ Division entière  
☐ Partie entière

**Question 25** Que vaut `a` à la fin de ces instructions ? `a = True; a = not a`

☐ Maybe    ☐ True    ☒ False    ☐ Right    ☐ Wrong

**Question 26** Que vaut `a` à la fin de ces instructions ? `a = 3; a = a**3`

☐ int    ☐ "aaa"    ☒ 27    ☐ "a3a3a3"    ☐ 9

**Question 27** Que vaut `c` à la fin de ces instructions ? `a = 42; b = 8; c = a % b`

☐ 1    ☐ 0    ☒ 2    ☐ 7    ☐ 14

**Question 28** Que vaut `a` à la fin de ces instructions ? `a = 1; a -= 3`

☐ 1    ☐ -1    ☐ -3    ☒ -2    ☐ 2

**Question 29** Quel est le type Python du résultat de : `"Inform" + "atique"` ?

☐ int    ☐ bool    ☒ str    ☐ chain    ☐ real

## 3 Mots-clefs

**Question 30 ★** Parmi ces mots, quels sont les mots-clefs Python ?

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

**Question 31 ★** Parmi ces mots, quels sont les mots-clefs Python ?

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

**Question 32 ★** Parmi ces mots, quels sont les mots-clefs Python ?

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

**Question 33 ★** Parmi ces mots, quels sont les mots-clefs Python ?

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

## 4 Bibliothèques

**Question 34 ★** Quelles sont les syntaxes possibles pour utiliser la fonction `randrange` du module `random` ?

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

**Question 35** Comment importer la fonction `sin` du module `math` ?

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

**Question 36** Comment importer les fonctions `sin` et `log` du module `math` ?

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

**Question 37 ★** Quelles sont les syntaxes possibles pour utiliser la fonction `array` du module `numpy` ?

- ☐ `import numpy; import array; a = 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])`
- ☐ `from numpy import array; a = numpy.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 38** 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])
```

- ☐ age
 ☐ False
 ☐ 42
 ☒ True
 ☐ None

## CORRECTION

**Question 39** Que vaut la variable `age` à la fin de ces instructions ?

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

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

**Question 40 ★** Que pensez vous ce programme ?

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

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

**Question 41** Que vaut la variable `age` à la fin de ces instructions ?

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

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

**Question 42 ★** Quels mots clefs permettent de créer une structure itérative en Python ?

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

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

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

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

CORRECTION

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

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

**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** En Python, l'indentation

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

**Question 47** Choisir la bonne réponse.

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

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

**Question 48** Que vaut la variable `age` à la fin de ces instructions ?

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

- ☐ 22
- ☒ 12
- ☐ 7
- ☐ 10
- ☐ 19

## 6 Listes

**Question 49** 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
```

- ☒ 10
- ☐ 4
- ☐ 15
- ☐ 5
- ☐ 6

**Question 50 ★** L'instruction `21 not in L` permet :

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

**Question 51** 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
- ☐ 4
- ☒ 5
- ☐ 6
- ☐ 10

**Question 52** 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) - i - 1]
print(L)
```

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

**Question 53** L'instruction `len(L)>0` permet de savoir :

- ☐ si la variable L est une liste d'entiers
- ☐ si la variable L est un conteneur positif
- ☐ si la variable L est une liste positive
- ☐ si la variable L est une liste
- ☒ si la variable L est un conteneur vide

**Question 54** 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 55** 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 56** Si M et L sont des listes Python, quel est le type de l'objet qui résulte de l'opération M+L ?

- ☐ `int`    ☐ `float`    ☒ `list`    ☐ `str`    ☐ `dict`

**Question 57** L'instruction `v=["3"]` permet de créer :

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

**Question 58** Que vaut la variable L à la fin de ce script ?

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

- ☐ `[0, 1, 2, 3, 4]`  
☐ `[2, 3, 4, 5, 6]`  
☐ `[5, 4, 3, 2, 1]`  
☐ `[4, 3, 2, 1, 0]`  
☒ `[1, 2, 3, 4, 5]`



**Question 59** 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)
```

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

**Question 60** 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
```

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

**Question 61** 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, 0, 1, 2, 0, 2]
- ☐ [0, 2, 1, 0, 1, 2]
- ☐ [2, 1, 2, 1, 2, 1]
- ☒ [1, 2, 1, 2, 1, 2]

**Question 62** 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, 2, 3, 4, 5, 6]
- ☐ [0, 2, 4, 6, 8, 10]
- ☐ [0, 1, 3, 5, 7, 9]
- ☐ [1, 4, 3, 2, 5, 6]
- ☐ [6, 5, 4, 3, 2, 1]

**Question 63 ★** 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
- ☒ d'affecter à `e` le dernier élément de la liste L
- ☐ de supprimer le premier élément de la liste L
- ☐ de consulter le premier élément de la liste L
- ☐ de consulter le dernier élément de la liste L
- ☐ Aucune de ces réponses n'est correcte.

**Question 64** 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]
- ☐ [0, 1, 2, 3, 4]
- ☐ [1, 2, 3, 4, 5]
- ☐ [5, 4, 3, 2, 1]
- ☒ []

**Question 65** 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]]
- ☐ [1, 2, 3, 4, 5, 6]
- ☒ [[1, 2, 3], [4, 5, 6]]
- ☐ [[6, 5, 4], [3, 2, 1]]
- ☐ [6, 5, 4, 3, 2, 1]

**Question 66 ★** 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` désignent la même variable en mémoire.
- ☐ `u` et `v` ne possèdent pas les mêmes éléments.
- ☒ `u` et `v` ne désignent pas la même variable en mémoire.
- ☒ `u` et `v` possèdent les mêmes éléments.
- ☐ `u` et `v` possèdent des éléments différents.
- ☐ Aucune de ces réponses n'est correcte.

**Question 67** L'instruction `L.append(3)` permet :

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

**Question 68** 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]`

**Question 69** 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]`
- ☐ `[0, 1, 1, 1]`
- ☒ `[0, 1, 0, 1]`
- ☐ `[1, 0, 0, 1]`
- ☐ `[1, 0, 1, 0]`

## 7 Fonctions

**Question 70** Dans le code ci-dessous, quel est le type retourné par la fonction ?

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

- ☐ `list`
☒ `bool`
☐ `str`
☐ `int`
☐ `float`

**Question 71 ★** Dans le code ci-dessous, la variable c

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

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

**Question 72** Dans le code ci-dessous, que vaut la variable `c` ?

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

c = f()
```

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

**Question 73** Dans le code ci-dessous, quel est le type retourné par la fonction ?

```
def f(a, b):
    if a + b > 0:
        return True
    else:
        return False
```

- ☒ bool
 ☐ float
 ☐ list
 ☐ int
 ☐ str

**Question 74** 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(5)`
☐ `u 5`
☐ `u`

**Question 75** Dans le code ci-dessous, 3 et 5 sont des paramètres

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

- ☐ formels
 ☐ inductifs
 ☐ optionnels
 ☐ entiers
 ☒ effectifs

**Question 76** Le mot clef `def` permet de définir :

- ☐ un nombre
 ☐ un paramètre
 ☐ une règle
 ☒ une fonction
 ☐ une variable

**Question 77** Dans le code ci-dessous, `a` et `b` sont des paramètres

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

- ☐ effectifs
 ☐ inductifs
 ☐ optionnels
 ☒ formels
 ☐ entiers

**Question 78** 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 valable pour l'intelligibilité du code
- ☐ idée logique par rapport à l'objectif de la fonction
- ☐ idée logique qui fait gagner du temps

**Question 79 ★** Les paramètres d'une fonction peuvent être :

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

**Question 80 ★** Le prototype d'une fonction Python indique :

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

## 8 Trier et rechercher

**Question 81** Le principe du tri par insertion est de

- ☐ 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 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.
- ☐ compter le nombre d'occurrences de chaque valeur entière puis de construire un nouveau tableau à partir de ce comptage
- ☐ 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)

**Question 82** 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 un ordre quelconque
- ☐ chercher l'élément en testant chaque case du tableau trié
- ☒ chercher l'élément en testant chaque case du tableau dans l'ordre des indices

**Question 83** Un tri stable est un algorithme de tri qui

- ☐ préserve la taille du tableau une fois trié
- ☐ ne préserve pas la taille des éléments dans le tableau 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 84 ★** La recherche dichotomique d'un élément dans un tableau

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

**Question 85 ★** Le tri par insertion est un tri

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

**Question 86** Le principe du tri par sélection est de

- ☐ 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
- ☐ 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 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)

**Question 87** Dans le meilleur des cas, le tri par insertion est

- ☐ aussi efficace que le tri par sélection
- ☐ moins efficace que le tri par sélection
- ☐ il n'y a pas de meilleur cas
- ☐ impossible à comparer aux autres tris
- ☒ plus efficace que le tri par sélection

**Question 88** Dans le pire des cas, le tri par insertion et le tri par sélection ont une complexité

- ☐ en  $O(n \log n)$
- ☒ en  $O(n^2)$
- ☐ en  $O(\log n)$
- ☐ en  $O(n)$

**Question 89** Un tri comparatif est un algorithme de tri qui procède en comparant

- ☐ les éléments à trier en commençant par la fin
- ☐ les indices des éléments à trier
- ☒ les éléments à trier deux à deux
- ☐ les indices des éléments à trier en partant du début
- ☐ les éléments du début avec ceux de la fin

**Question 90 ★** Un tri en place est un algorithme de tri qui

- ☐ ne peut pas 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
- ☒ peut être directement effectué dans le tableau initial
- ☐ nécessite l'allocation d'une nouvelle structure en mémoire
- ☐ Aucune de ces réponses n'est correcte.

**Question 91** Un tri en ligne est un algorithme de tri qui peut commencer

- ☐ le tri grâce à une connexion internet
- ☐ 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 s'il possède un comparateur incrémental des données

## 9 Récursivité

**Question 92** Un algorithme récursif

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

**Question 93** Le code suivant

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

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

**Question 94 ★** Pour formuler correctement un algorithme récursif, il est nécessaire de

- ☐ faire des appels récursifs avec des données plus proches de la condition de continuation
- ☒ prévoir une condition d'arrêt sans appels récursifs
- ☐ prévoir une condition de continuation avec appels récursifs
- ☐ prévoir des appels récursifs inconditionnels
- ☒ faire des appels récursifs avec des données plus proches de la condition d'arrêt
- ☐ Aucune de ces réponses n'est correcte.

**Question 95 ★** Le code suivant

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

- ☐ affiche 6 sur la console
- ☒ produit un résultat de type `int` si  $n$  est un `int`
- ☒ affiche 0 sur la console
- ☐ affiche 5 sur la console
- ☐ produit une exception de type `RecursionError`
- ☐ Aucune de ces réponses n'est correcte.

**Question 96** En Python, on peut effectuer

- ☐ un petit nombre d'appels récursifs à cause de la grande taille de la pile d'exécution
- ☐ 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