Révisions python - à la main

BARBIER J.M. - LGT Dumezil - TNSI

1 Dérouler un programme sur papier

1.1 Pourquoi

— pas de machine à l'épreuve écrite
— lire un code = savoir dire ce qu'il fait, ligne par ligne
— même méthode que le mode **ardoise** de PyMemViz

1.2 Les trois choses à tenir à jour

1. **Le fil** : la suite des lignes exécutées, dans l'ordre réel (pas l'ordre du fichier).
2. **L'ardoise** : le tableau des variables — nom et valeur.
3. **La sortie** : ce que `print` affiche, une ligne à la fois.

1.3 Les règles

— une seule ligne à la fois
— une variable = une ligne de l'ardoise
— valeur qui change → on **raye** l'ancienne, on écrit la nouvelle en dessous
— on n'efface jamais : l'ardoise garde l'historique
— le type : pas obligatoire, seulement quand il est utile (divisions, conversions)
— `print` n'écrit **que** dans la sortie, jamais dans l'ardoise
— une variable qui n'existe pas encore n'a pas de ligne

1.4 Exemple

```
1  a = 3
2  b = a + 1
3  print(a, b)
4  a = a * 2
5  b = a - b
6  print(a, b)
```

1.4.1 Le fil

Étape	Ligne	Ce qui se passe
1	1	création de a
2	2	création de b
3	3	affichage

Étape	Ligne	Ce qui se passe
4	4	a change
5	5	b change
6	6	affichage

Ici le fil est simple : 1, 2, 3, 4, 5, 6. Avec un test ou une boucle, il saute.

1.4.2 L'ardoise

Nom	Valeur
a	~~3~~ 6
b	~~4~~ 2

À l'étape 4, `a = a * 2` : on lit l'ancienne valeur (3), on calcule (6), on raye 3, on écrit 6.

À l'étape 5, `b = a - b` : on lit **les valeurs courantes**, donc 6 - 4 = 2.

1.4.3 La sortie

```
3 4
6 2
```

1.5 À retenir

- = n'est pas une égalité : c'est une **affectation**, de droite à gauche (`a <--- a * 2`)
- on calcule d'abord la droite, **avec les valeurs actuelles**, puis on range dans la gauche
- `a += 3` équivaut à `a = a + 3`
- deux variables peuvent avoir la même valeur sans être la même case

> Les **fonctions** demandent une ardoise par appel (cf. PyMemViz : une carte `Programme principal` + une carte par appel). On verra ça plus tard.

1.6 Les corrections

Chaque exercice a un lien **PyMemViz** (mode *ardoise*). À ouvrir **après** avoir déroulé sur papier :

- comparer son fil, son ardoise et sa sortie avec ceux de la machine
- lancer soi-même l'exécution (bouton **Lecture auto.**), pour voir les valeurs changer et les ardoises apparaître / disparaître en direct
- bouton **Éditer** pour modifier le code et rejouer

2 Exercices : types et opérations

2.1 Exercice 1

```
print(7 // 2)
print(7 % 2)
print(7 / 2)
print(2 ** 5)
print(type(7 // 2))
print(type(7 / 2))
```

Donner la sortie complète.

Correction : dérouler dans PyMemViz

2.2 Exercice 2

```
print(int(4.99))
print(float(4))
print(int("12") + 3)
print("12" + "3")
print(0.1 + 0.2 == 0.3)
```

Donner la sortie. Expliquer les deux dernières lignes.

Correction : dérouler dans PyMemViz

3 Exercices : variables et affectation

3.1 Exercice 3

```
a = 2
b = 5
a += b - 6
b -= a
a *= b
print(a, b)
```

Ardoise complète (valeurs rayées) + sortie.

Correction : dérouler dans PyMemViz

3.2 Exercice 4

```
x = "chat"
y = "chien"
z = x
x = y
y = z
print(x, y, z)
```

Ardoise + sortie. Que fait ce programme, en une phrase ?

Correction : dérouler dans PyMemViz

3.3 Exercice 5

```
x = 3
y = 8
x = y
y = x
print(x, y)
```

Ardoise + sortie. Pourquoi l'échange ne marche-t-il pas ?

Correction : dérouler dans PyMemViz

3.4 Exercice 6

```
a = b = c = 3
b = 5
print(a, b, c)
x, y = 1, 2
x, y = y + 1, x + 1
print(x, y)
```

Ardoise + sortie. Sur la ligne `x, y = y + 1, x + 1` : qu'est-ce qui est calculé en premier ?

Correction : dérouler dans PyMemViz

3.5 Exercice 7

```
a = "chat"
b = "chien"
a, b = b, a
print(a, b)
```

Ardoise + sortie. Pourquoi ça marche, alors que la méthode de l'exercice 5 ne marche pas ? Combien de variables sont nécessaires, contre l'exercice 4 ?

Correction : dérouler dans PyMemViz

4 Exercices : booléens et tests

4.1 Exercice 8

```
a = 7
b = 7
print(a > 3 and b < 10)
print(a > 3 or b > 10)
print(not (a == b))
print(3 <= a < 7)
print(a != b or not (a > 100))
```

Donner la sortie.

Correction : dérouler dans PyMemViz

4.2 Exercice 9

Écrire (sans machine) le test correspondant :

- n est dans l'intervalle `[5, 12[`
- n est pair **et** strictement supérieur à 100
- n n'est **ni** un multiple de 3 **ni** un multiple de 5

Correction : dérouler dans PyMemViz

5 Exercices : structures conditionnelles

5.1 Exercice 10

```
note = 11
if note >= 16:
    mention = "TB"
elif note >= 14:
    mention = "B"
elif note >= 12:
    mention = "AB"
elif note >= 10:
    mention = "passable"
else:
    mention = "recalé"
print(mention)
```

Donner le **fil** (numéros de lignes exécutées) et la sortie, pour :

- `note = 11`
- `note = 17`
- `note = 9`

Correction : dérouler dans PyMemViz

5.2 Exercice 11

```
x = 15
if x > 5:
    print("plus de 5")
if x > 10:
    print("plus de 10")
if x > 20:
    print("plus de 20")
else:
    print("pas plus de 20")
```

Sortie ? Puis : réécrire avec des `elif` et redonner la sortie. Conclusion ?

Correction : dérouler dans PyMemViz

6 Exercices : boucles while

6.1 Exercice 12

```
compteur = 10
total = 0
while compteur > 0:
    total += compteur
    compteur -= 3
print(compteur, total)
```

Ardoise (toutes les valeurs successives, rayées) + sortie. Combien de tours ?

Correction : dérouler dans PyMemViz

6.2 Exercice 13

```
i = 0
while True:
    i += 1
    if i % 3 == 0:
        continue
    if i > 7:
        break
    print(i, end=" ")
print("fin", i)
```

Sortie ? Que fait `continue` ? que fait `break` ?

Correction : dérouler dans PyMemViz

6.3 Exercice 14

Écrire une boucle `while` qui calcule $6!$ ($= 1 \times 2 \times ... \times 6$) dans une variable `resultat`. Puis dérouler l'ardoise pour vérifier.

Correction : dérouler dans PyMemViz

7 Exercices : listes

7.1 Exercice 15

```
t = [4, 8, 15, 16, 23, 42]
print(t[0], t[3], t[-1])
print(len(t))
print(t[1:4])
print(t[:2])
print(t[-2:])
print(23 in t, 24 in t)
```

Donner la sortie.

Correction : dérouler dans PyMemViz

7.2 Exercice 16

```
t = [1, 2, 3]
t.append(4)
t.insert(1, 99)
del t[0]
t[2] = 0
print(t)
print(len(t))
```

Ardoise : **une seule** variable, une valeur rayée à chaque modification.

Correction : dérouler dans PyMemViz

7.3 Exercice 17

```
grille = [[0, 0, 0],
          [0, 0, 0],
          [0, 0, 0]]
grille[1][1] = 1
grille[0][2] = 2
grille[2][1] = 1
print(grille)
print(grille[1])
print(grille[1][2])
```

Dessiner la grille 3 × 3 à la fin, puis donner la sortie.

Correction : dérouler dans PyMemViz

8 Exercices : boucles for

8.1 Exercice 18

```
for i in range(3, 12, 4):
    print(i, end=" ")
print()
for i in range(4):
    print(i * i, end=" ")
```

Sortie ?

Correction : dérouler dans PyMemViz

8.2 Exercice 19

```
notes = [12, 8, 17, 3]
total = 0
maxi = 0
for n in notes:
    total += n
    if n > maxi:
        maxi = n
print(total, total / 4, maxi)
```

Ardoise (n, `total`, `maxi` changent à chaque tour) + sortie.

Puis : que renvoie ce programme si toutes les notes sont négatives ? Corriger.

Correction : dérouler dans PyMemViz

8.3 Exercice 20

```
mots = ["un", "deux", "trois", "quatre"]
for pos, mot in enumerate(mots):
    if len(mot) == 4:
        continue
    print(pos, mot)
    if mot == "trois":
        break
```

Fil + sortie. Combien de tours de boucle réellement exécutés ?

Correction : dérouler dans PyMemViz

8.4 Exercice 21

```
carres = [n * n for n in range(1, 6) if n % 2 == 1]
```

Réécrire avec une boucle `for` et `append`. Donner le contenu final de `carres`.

Correction : dérouler dans PyMemViz

9 Exercices : références

9.1 Exercice 22

```
a = [1, 2, 3]
b = a
c = a[:]
b[0] = 99
c[1] = 77
print(a, b, c)
```

Sortie ? Quelles variables désignent **la même** liste en mémoire ?

Correction : dérouler dans PyMemViz

10 Exercices : dictionnaires

10.1 Exercice 23

```
fiche = {"nom": "Martin", "age": 17}
fiche["classe"] = "TNSI"
fiche["age"] = 18
del fiche["nom"]
print(fiche)
print(len(fiche))
print("nom" in fiche, "classe" in fiche)
```

Ardoise (une ligne `fiche`, valeurs rayées) + sortie.

Correction : dérouler dans PyMemViz

10.2 Exercice 24

```
stock = {"pomme": 5, "poire": 0, "kiwi": 12}
for produit in stock:
    print(produit, end=" ")
print()
for n in stock.values():
    print(n, end=" ")
print()
total = 0
for produit, n in stock.items():
    if n > 0:
        total += n
print(total)
```

Sortie ? Différence entre les trois parcours ?

Correction : dérouler dans PyMemViz

10.3 Exercice 25

```
notes = {"alice": [10, 12], "bob": [14, 16]}
notes["alice"].append(8)
notes["charlie"] = [9]
print(notes["alice"][2])
print(len(notes))
print(notes)
```

Sortie ? Que compte `len(notes)` exactement ?

Correction : dérouler dans PyMemViz