Corrigé – Les piles

Exercice 1

1. La pile `p` contient, du haut vers le bas, les caractères `"i"` et `"n"`.
2. Les piles fonctionnent en mode LIFO (Last In First Out).

Exercice 2

```
# FONCTIONS
def echange(p):
    dernier = p.depile()
    avant_dernier = p.depile()
    p.empile(dernier)
    p.empile(avant_dernier)

# SCRIPT
pile = Pile()
pile.empile(1)
pile.empile(2)
pile.empile(3)
pile.empile(4)
print(pile)
echange(pile) # renvoie None (renvoyé par défaut car pas de return explicite)
print(pile) # la modification est faite en place donc pile doit avoir été modifiée
```

Exercice 3

```
# FONCTIONS
def inverse(p):
    p_inverse = Pile()
    p_auxiliaire = Pile()
    while not p.est_vide():
        x = p.depile()
        p_inverse.empile(x)
        p_auxiliaire.empile(x)
    # p est maintenant vide
    # on rempile les éléments de p_auxiliaireiliaire sur p
    while not p_auxiliaire.est_vide():
        p.empile(p_auxiliaire.depile())

    return p_inverse

# SCRIPT
pile = Pile()
pile.empile(1)
pile.empile(2)
pile.empile(3)
pile.empile(4)
```

```
pile_inverse = inverse(pile) # on renvoie ici une nouvelle pile comme demandé dans
↪  l'énoncé, pile n'est donc pas modifiée comme précédemment.
print(pile_inverse)
```

Exercice 4

```
# FONCTIONS
def copie(p):
    p_copie = Pile()
    p_auxiliaire = Pile()
    while not p.est_vide():
        p_auxiliaire.empile(p.depile())

    while not p_auxiliaire.est_vide():
        x = p_auxiliaire.depile()
        p.empile(x)
        p_copie.empile(x)

    return p_copie

def copie_maline(p):
    '''Une version courte qui réutilise la fonction écrite précédemment'''
    p_inverse = inverse(p)
    p_copie = inverse(p_inverse)
    return p_copie

# SCRIPT
pile_copie = copie(pile) # on utilise la pile définie dans l'exercice précédent
print(pile_copie)

pile_copie_maline = copie_maline(pile)
print(pile_copie_maline)
```

Exercice 5

```
# FONCTIONS

# 1)
def fond(p):
    p_aux = Pile()
    while not p.est_vide():
        p_aux.empile(p.depile())
    el_fond = p_aux.depile()
    while not p_aux.est_vide():
        p.empile(p_aux.depile())
    return el_fond
```

```python
# 2)
def rotation(p):
    pile_aux = Pile()
    futur_fond = p.depile()
    while not p.est_vide():
        pile_aux.empile(p.depile())
    futur_haut = pile_aux.depile()

    p.empile(futur_fond)
    while not pile_aux.est_vide():
        p.empile(pile_aux.depile())
    p.empile(futur_haut)

# SCRIPT
# On utilise la pile créée dans les exercices précédents
assert fond(pile) == 1
print(pile) # doit avoir 2 (et plus 1) tout en bas

# réinitialisation de la pile
pile = Pile()
pile.empile(1)
pile.empile(2)
pile.empile(3)
pile.empile(4)
pile.empile(5)
print(pile)

rotation(pile)
print(pile) # doit avoir 5 tout en bas et 1 tout en haut, le reste inchangé
```

Exercice 6

```python
# FONCTIONS
def test_parenthesage(expression):
    pile = Pile()
    for caractere in expression:
        if caractere in "([{":
            pile.empile(caractere)
        if caractere in ")]}":
            if pile.est_vide():
                return False
            caractere_deja_lu = pile.depile()
            if caractere_deja_lu=="(" and caractere!=")" \
               or caractere_deja_lu=="[" and caractere!="]" \
               or caractere_deja_lu=="{" and caractere!="}":
                return False
    return pile.est_vide()

# SCRIPT
assert test_parenthesage("(a)(b)(((c)(d)))")==True
# ... TSVP
```

```
assert test_parenthesage("([b]){((c))[d]}")==True
assert test_parenthesage("(")==False
assert test_parenthesage("(a))")==False
assert test_parenthesage("((a))}")==False
assert test_parenthesage("[[a]")==False
assert test_parenthesage("[(a])")==False
assert test_parenthesage("{((a)})")==False
```

Exercice 7

```
#FONCTIONS
def ranger(p):
    pile_assiettes_blanches = Pile()
    pile_assiettes_vertes = Pile()

    while not p.est_vide():
        assiette = p.depile()
        couleur = assiette[0]
        if couleur == "blanc":
            pile_assiettes_blanches.empile(assiette)
        else:
            pile_assiettes_vertes.empile(assiette)

    while not pile_assiettes_blanches.est_vide():
        p.empile(pile_assiettes_blanches.depile())
    while not pile_assiettes_vertes.est_vide():
        p.empile(pile_assiettes_vertes.depile())

# SCRIPT
pile_assiettes = Pile()
pile_assiettes.empile(("blanc", 3))
pile_assiettes.empile(("vert", 3))
pile_assiettes.empile(("vert", 4))
pile_assiettes.empile(("blanc", 5))
pile_assiettes.empile(("blanc", 6))
pile_assiettes.empile(("vert", 7))
pile_assiettes.empile(("blanc", 8))
pile_assiettes.empile(("vert", 9))
print(pile_assiettes)

ranger(pile_assiettes)
print(pile_assiettes)
```