TD compléments

Première NSI

qkzk

2022-07-07

1. Vocabulaire sur les fonctions

On donne le script python suivant ;

```
def reste_division(nb: int, diviseur: int) -> int:
    """retourne le reste de la division euclidienne de 'nb'
    par 'diviseur' """
    reste = nb % diviser
    if reste < 0:
        reste = reste + abs(diviseur)
    return reste
```

1. Quel est le nom de la fonction ?
2. Quel est le rôle de `reste`. Est-ce une variable globale ?
3. Même question pour `diviser`
4. `return` est-il une variable ? Comment appelle-t-on ce type de mot ?

2. Instructions condtionnelles

On considère la fonction suivante :

```
def mystere(n: int) -> str:
    if n == 7:
        resultat = 'A'
    else:
        if n == 5:
            resultat = 'B'
        else:
            resultat = 'C'
    return resultat
```

1. Quelle est la valeur de `mystere(7)` ?
2. Quelle est la valeur de `mystere(2)` ?
3. Quelle est la valeur de `mystere(5)` ?
4. Quelle est la valeur de `mystere('7')` ?

3. Répétitions

1. Quelle structure de contrôle permet de répéter des traitements ?
2. Quelles sont les valeurs prises successivement par la variable `i` dans la boucle `for` ci-dessous ?

```
res = 0
for i in range(3):
    res = res + i
```

4. Lire des instructions conditionnelles

On définit la fonction mystère suivante :

```
def mystere(n: int) -> str:
    if n % 3 == 0 or n % 5 == 0:
        if n % 3 == 0:
            resultat = "A"
        else:
            resultat = "B"
    else:
        if n % 5 == 0:
            resultat = "C"
        else:
            resultat = "D"
    return resultat
```

1. Quelle est la valeur de `mystere(2)`
2. Quelle est la valeur de `mystere(6)`
3. Quelle est la valeur de `mystere(15)`
4. Quelle est la valeur de `mystere(10)`
5. Quelle est la valeur de `mystere(5)`

5. Programmer son chauffage électrique

C'est le weekend et Robert a programmé ses radiateurs de la façon suivante :

- ils sont en mode "Confort" de 9h à 22h ;
- le reste du temps, ils sont en mode "Eco".

1. Quel est le mode utilisé à 8h ? à 17h ?

On donne le code, incomplet, d'une fonction qui détermine le mode à utiliser en fonction de l'heure.

```
def mode_weekend(heure: float) -> str:
    """
    paramètre : 'heure' est un nombre à virgule
    résultat : le mode des radiateurs quand robert ne travaille pas
    précondition : ????
    """
    assert heure >= 0 and heure < 24
    if heure < 9:
        mode = "Eco"
    elif heure < 22:
        mode = "Confort"
    else:
        mode = ???
    return mode


assert mode(3) == "Eco"
assert mode(7.5) == ????
```

2. Compléter la ligne précondition dans la documentation.
3. Compléter les tests.
4. Dans cas passe-t-on par la ligne 14 (`mode = ???`)
5. Compléter la ligne 14.
6. En semaine, Robert veut faire des économies d'énergie. Il programme donc ses radiateurs de la façon suivante :

- en semaine (du lundi au vendredi) ils sont en mode “Confort” de 6h à 9h et de 17h à 22h ; le reste du temps, ils sont en mode “Eco” ;
- le weekend (le samedi et le dimanche) ils sont en mode “Confort” de 9h à 22h ; le reste du temps ils sont en mode “Eco”.

Écrire une fonction `mode_semaine` qui prend en paramètre une heure et qui renvoie le mode des radiateurs quand Robert travaille (du lundi au vendredi).

7. On donne le code incomplet de la fonction `mode`. Remplacer tous les ??? par du code correct.

```
def mode(jour: str, heure: float) -> str:
    """
    paramètres :
    - 'jour' est une chaîne de caractères indiquant le jour de la semaine
    - 'heure' est un nombre flottant
    résultat : le MODE des radiateurs de Robert au 'jour' et à l''heure'
    donnés
    préconditions :
    - joueur doit etre égal à à 'lundi' ou 'mardi' ou 'mercredi' ou 'jeudi'
    ou 'vendredi' ou 'samedi' ou 'dimanche'
    - l'heure doit ????
    """
    ???
    assert (heure >= 0 and heure < 24)
    if jour == 'samedi' or jour == 'dimanche':
        mode = ???
    else:
        mode = ???
    return mode

assert mode(???) == 'Eco'
assert mode(???) == 'Confort'
```

6. Identifier les différentes parties d’une fonction (1)

On définit le script ci-dessous :

```
def test(x: int, y: int) -> int:
    somme = 0
    for i in range(y):
        somme = somme + x
    return somme

z = test(5, 4)
```

1. Quel est le nom de la fonction ?
2. Quels sont les paramètres formels de la fonction ?
3. Quels sont les paramètres réels ?
4. Quelles sont les variables locales à la fonction ?
5. Quelle sera la valeur de `z` après l’exécution de ce script ?

7. Identifier les différentes parties d’une fonction (2)

On définit le script ci-dessous :

```python
def test(mot: str, lettre: str) -> int:
    compteur = 0
    for caractere in mot:
        if caractere == lettre:
            compteur = compteur + 1
    return compteur

n = test('Trololo', 'o')
```

1. Quel est le nom de la fonction ?
2. Quels sont les paramètres formels de la fonction ?
3. Quels sont les paramètres réels ?
4. Quelles sont les variables locales à la fonction ?
5. Quelle sera la valeur de `n` après l'exécution de ce script ?

8. Comprendre une boucle non bornée

On donne la fonction suivante.

```python
def mystere(nombre: int) -> int:
    while nombre > 5:
        nombre = nombre - 5
    return nombre
```

Parmi les affirmations suivantes, lesquelles sont vraies :

1. on sort de la boucle `while` dès que `nombre > 5`.
2. on sort de la boucle `while` dès que `nombre < 5`.
3. on sort de la boucle `while` dès que `nombre >= 5`.
4. on continue la boucle `while` tant que `nombre > 5`.
5. on continue la boucle `while` tant que `nombre < 5`.
6. on continue la boucle `while` tant que `nombre >= 5`.

9. Coût et complexité

Un programme traite des données dont la taille peut être mesurée à l'aide d'une variable n. Si $n = 100$, le programme retourne un résultat en 8 ns. On admet que le temps d'exécution de ce programme évolue proportionnellement à une certaine fonction de n.

Par exemple si le temps évolue proportionnellement à n^2, lorsque l'on triple la valeur de n, le temps est multiplié par $3^2 = 9$.

Pour les fonctions *non polynômiales*, comme $\log n$, il convient de d'abord calculer le facteur en posant $k = \dfrac{8}{\log 100}$. Ensuite on peut calculer $k \log(200)$ et on obtient un temps d'exécution en ns.

Compléter le tableau de durées approximatives ci-dessous :

n	$\log n$	n	$n \log n$	n^2	n^3	2^n
100	8 ns	8 ns	8 ns	8ns	8ns	8 ns
200	9.2 ns	16 ns	18 ns	32ns	64ns	1.01×10^{31} ns
1000						
10000						
100000						

10. Cumul

On dispose d'une liste de nombres et on souhaite calculer la liste des valeurs cumulées.

On dispose d'un tableau contenant des effectifs et l'on souhaite créer un tableau d'effectifs cu- mulés. Par exemple `eff=[2, 7, 8, 5, 6, 4]` donnera `eff_cumul=[2, 9, 17, 22, 28, 32]`.

Première approche

Une première approche consiste à additionner toutes les valeurs à chaque calcul :

```
N = longueur du tableau
eff_cumul = liste vide

Pour i allant de 0 à N - 1:
    cumul = 0
    Pour j allant de 0 à i:
        cumul = cumul + eff[j]
    Ajouter cumul à la fin de eff_cumul
```

1. Compléter le tableau ci-dessous

i	valeurs prises par j	calcul effectué	eff_cumul
0	de 0 à 0	2	[2]
1	de 0 à 1	2+7	[2, 9]
2			
3			
4			
5			

2. Combien d'additions ont-été effectuées au total ?
3. Pour un tableau de 100 nombres, combien d'additions seront effectuées ?

Seconde approche

Une nouvelle approche consiste à exploiter les calculs déjà faits à chaque calcul :

```
N prend la valeur de longueur(eff)
eff_cumul est une liste vide
cumul = 0

Pour i allant de 0 à N (exclus) :
    cumul = cumul + eff[i]
    Ajouter cumul à la fin de eff_cumul
```

2. Appliquer l'algorithme pour déterminer les effectifs cumulés de `[2, 7, 8, 5, 6, 4]`.
3. Quelle façon de procéder est la plus efficace ?

11. Tester

On propose ci-dessous le code d'une fonction permettant de compter le nombre de voyelles dans une chaîne de caractères :

```
def voyelles(chaine : str) -> int :
compteur = 0
for lettre in chaine :
    if lettre in "aeiouy" :
    compteur += 1
return compteur
```

1. Que renvoie l'appel `voyelles("Jean")` ? Est-ce correct ?

2. Proposer un test permettant de vérifier que cette fonction compte correctement le nombre de voyelles en minuscules.
3. Que renvoie l'appel `voyelles("Eric")` ? Est-ce correct ?
4. Proposer un test permettant de permettant de prouver que cette fonction gère mal les voyelles en majuscules.
5. Proposer un test permettant de permettant de prouver que cette fonction gère mal les voyelles accentuées.
6. Corriger le code de cette fonction afin qu'elle réponde aux tests précédents.

12. Tester

On propose le code python ci-dessous permettant de trouver l'indice de la dernière occurrence

```
def derniere_occurrence(liste : list, valeur) -> int:
    renverse = liste[::-1] # renverse la liste
    i = len(liste) - 1
    for elt in renverse :
        if elt == valeur :
            return i
        i -= 1
    return i
```

Indiquer ce que vérifient les tests ci-dessous :

1. `assert derniere_occurrence([3,2,1], 1) == 2`
2. `assert derniere_occurrence([3,2,1], 3) == 0`
3. `assert derniere_occurrence([3,2,1], 4) == -1`
4. `assert derniere_occurrence([3,2,1,1], 1) == 3`
5. `assert derniere_occurrence([], 1) == -1`
6. `assert derniere_occurrence([3,2,1], "a") == -1`
7. `assert derniere_occurrence([3,2,1], -1) == -1`

13. Variant de boucle

On considère l'algorithme ci-dessous :

```
a = 8
b = 20
Tant que a != b:
    a += 1
    b -= 2
```

1. Compléter le tableau ci-dessous présentant les valeurs des variables à l'issue d'un tour de boucle :

Tour	a	b	b - a
0	8	20	12
1	9	18	9
2			
3			
4			
5			

2. Proposer un variant de boucle pour cet algorithme.

14. Invariant de boucle

On condière le code Python ci-dessous qui calcule le nombre de bits nécessairs pour écrire un nombre entier positif en binaire :

```python
def nb_bits(n : int) -> int :
    assert n > 0
    i = 0
    reste = n
    while 2**i * reste < n :
        reste = reste / 2
        i = i + 1
    return i
```

1. Quel est le rôle de l'instruction `assert n > 0` ?

Afin de prouver la correction partielle de l'algorithme, on introduit l'invariant suivant :

$$\textit{À l'issue de chaque tour de boucle } i\textit{, on a } 2^i \times reste \leq n < 2^{i+1} \times reste$$

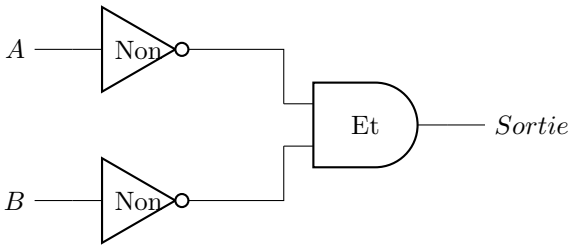
2. Prouver que cette propriété est vraie lors de l'initialisation de l'algorithme
3. Justifier que lors du i-ième tour de boucle on a :

$$reste = \frac{n}{2^i}$$

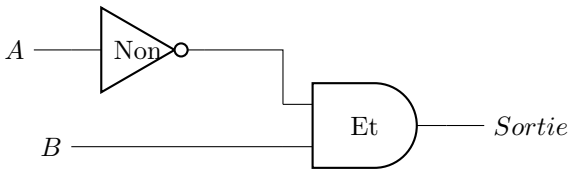
4. On suppose qu'il existe un entier k tel que l'invariant est vérifié après k tours de boucle. Prouver qu'il l'est aussi après $k+1$ tours.

15. Circuits logiques

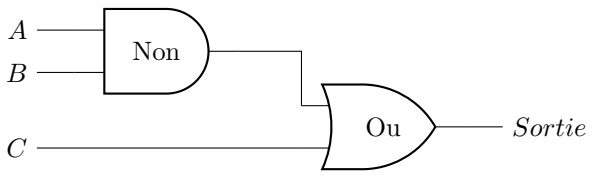
Compléter les tables de vérité des circuits logiques ci-dessous :



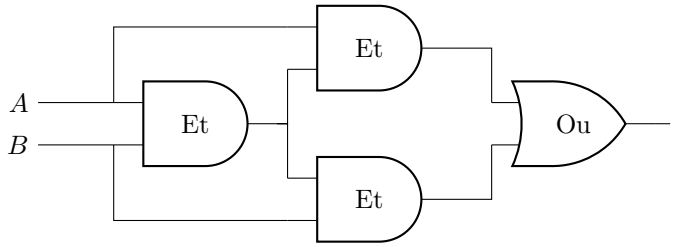
A	B	$Sortie$
0	0	1
0	1	
1	0	
1	1	



A	B	$Sortie$
0	0	
0	1	
1	0	
1	1	



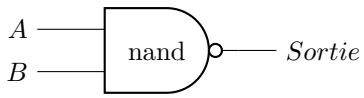
A	B	C	$Sortie$
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	



A	B	$Sortie$
0	0	
0	1	
1	0	
1	1	

16. Non et

On fournit ci-dessous le schéma et la table de vérité de la porte logique *non et* (*nand*)

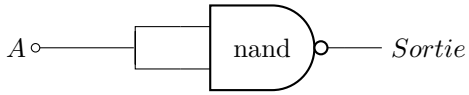


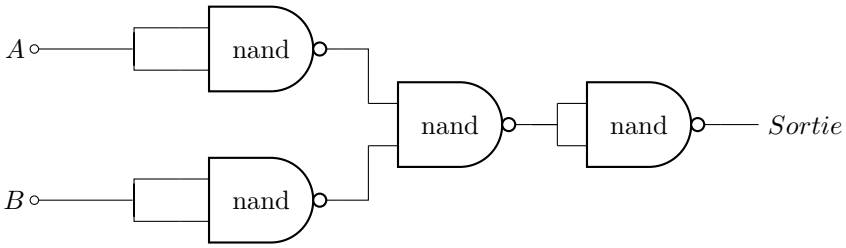
A	B	*Sortie*
0	0	1
0	1	1
1	0	1
1	1	0

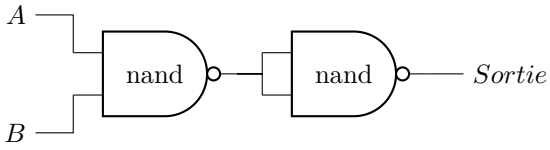
Cette porte permet à elle-seule de construire les portes logiques suivantes :

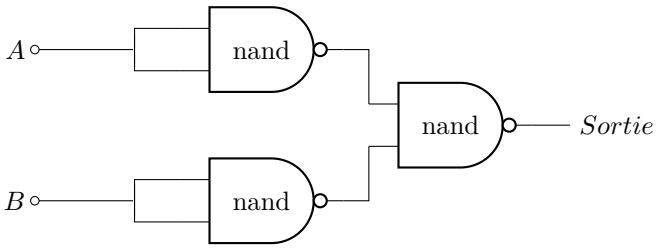
- non
- et
- ou
- non ou
- ou exclusif (*xor*)
- non ou exclusif

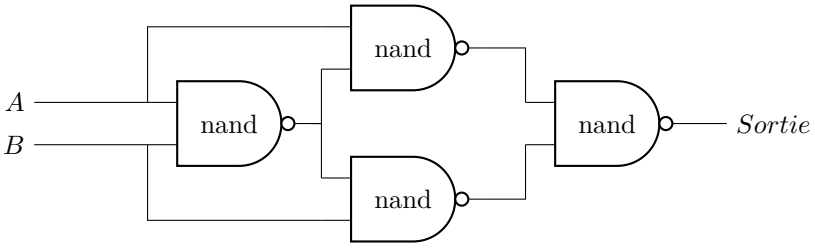
Associer chaque circuit à la bonne porte logique

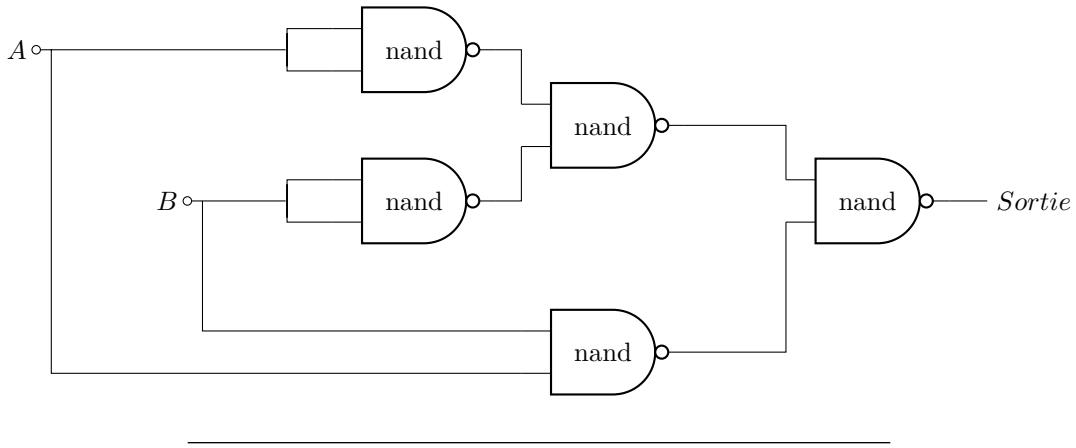












17. Carré de Polybe

Le codage dit du carré de Polybe remplace chaque caractère par sa position dans une grille définie à l'aide d'une clé. Voici la grille utilisée dans cet exercice. La clé est Polybe. Par habitude, on omet le W.

	0	1	2	3	4
0	P	O	L	Y	B
1	E	A	C	D	F
2	G	H	I	J	K
3	M	N	Q	R	S
4	T	U	V	X	Z

La lettre O est codée par 01 et la lettre E par 10.

1. Décoder le message 0422103123014110

On a codé le carré de Polybe à l'aide d'une liste dans Python :

```
carre = [["P", "O", "L", "Y", "B"], ["E", ...], ...]
```

2. Programmer une fonction `lettre` prenant deux paramètres entiers `i`, `j` ainsi qu'un `carre` (tel qu'au dessus) et qui renvoie la lettre associée aux coordonnées `(i, j)`.
3. Recopier et compléter le code ci-dessous :

```
def decode_polybe(message: str, carre: list) -> str:
    """Décode un message chiffré avec le carré de Polybe"""
    clair = ""
    for indice in range(0, len(...), ...):
        i = int(message[indice])
        j = ...
        clair += lettre(i, j, carre)
    return ...
```

4. Décoder le message suivant : 14100222122240114022013134404111343310413434 22

18. Écraser

Écrire en python la fonction `ecrase` prenant en argument une liste et un nombre `n` et renvoyant une copie de la liste. Cette fonction remplacera toutes les valeurs d'indices supérieurs ou égaux à `n` par 0. Par exemple `ecrase([3, 7, 8, 7, 9], 2)` renverra `[3, 7, 0, 0, 0]` alors que `ecrase([3, 7, 8, 7, 9], 5)` renverra `[3, 7, 8, 7, 9]`.

19. Supprimer

Écrire en python la fonction `supprime` prenant en argument une liste et un nombre `n` et renvoyant une copie de la liste. Cette fonction supprimera la première occurrence de `n` dans la liste en décalant toutes les valeurs vers le début et renverra

la liste ainsi formée. Par exemple `supprime([3, 7, 8, 7, 9], 7)` renverra `[3, 8, 7, 9]` alors que `supprime([3, 7, 8, 7, 9], 2)` renverra `[3, 7, 8, 7, 9]`.

20. Contient

Écrire une fonction `contient` qui prend une liste et un élément de n'importe quel type et renvoie un booléen `True` si l'élément figure dans la liste et `False` sinon. *Le test `a in b` est proscrit.*

21. Est croissant

Écrire une fonction `est_croissant` Python qui prend une liste d'entiers et renvoie un booléen valant `True` si la liste est triée par ordre croissant et `False` sinon.

22. Présenter des numéros

Un répertoire de contacts est enregistré sous la forme d'une liste de dictionnaires :

```
repertoire = [
  {"nom": "Duchmol", "prénom": "Raymond", "tel": "0611223344", "adresse": "1, rue Basse, 75000 Paris"},
  ...
]
```

Écrire une fonction `presenter` qui affiche une ligne par contact de la forme :

```
Raymont Duchmol, 0611223344, habite au 1, rue Basse, 75000 Paris.
...
```

_On pourra commencer par une fonction prenant un dictionnaire de cette forme et affichant une ligne.

23. Regrouper des dictionnaires

On décrit le contenu du stock d'un entrepôt à l'aide d'un dictionnaire. Par exemple `{"pommes" : 200, "fraises" : 100}` signifie que l'on stocke 200 kg de pommes et 100 de fraises.

On souhaite "regrouper" deux entrepôts/dictionnaires en cumulant les valeurs associées à des clés identiques. Ainsi, si les deux entrepôts stockent des pommes, le dictionnaire créé aura pour valeur associée à "pomme" la somme des deux valeurs.

Écrire en python la fonction `regroupe` qui prend en argument les deux dictionnaires et répond à la question.

Par exemple l'appel :

```
>>> regroupe({"pommes" : 200, "fraises" : 100}, {"pommes" : 100, "oranges" : 150})
{"pommes" : 300, "fraises" : 100, "oranges" : 150}`.
```

24. Mettre à jour des dictionnaires

Écrire une fonction `mettre_a_jour` qui prend deux dictionnaires en paramètres et en renvoie un nouveau tel que :

- si une clé figure dans seulement l'un des deux, elle figure avec sa valeur dans la sortie,
- si une clé figure dnas les deux, on ne garde que la valeur du *second* dictionnaire.

25. Dictionnaires de dictionnaires

Un lycée stocke les informations de ses élèves dans un dictionnaire dont les clés sont les noms des élèves (on suppose qu'il n'y a pas de doublons) et les valeurs leurs informations.

Par exemple :

```
lycee = {"Jean Dupont" : {"classe" : "1e1", "LVA" : "anglais", ...},
         "Romane Simon": {"classe" : "1e3", "LVA" : "allemand", ...},
         ...
}
```

1. Comment afficher la liste des noms des élèves du lycée ?
2. Comment afficher la classe de `"Jean Dupont"` ?
3. a. Écrire le code comptant le nombre d'élèves au lycée.
   b. Écrire le code comptant le nombre d'élèves en Première 1.

26. YAML, l'aller

Le format YAML est un format de données textuelles. Son acronyme signifie Yet Another Markup Language.

Dans ce format, c'est l'indentation qui organise l'information – comme en python.

Par exemple, pour les données décrites dans l'exercice précédent on obtient :

```yaml
eleves:
    - nom: Jean Dupont
      classe: 1e1
      LVA: anglais
    - nom: Romane Simon
      classe: 1e3
      LVA: allemand
```

Ecrire le code transformant le dictionnaire en un texte au format YAML.

27. YAML, le retour

Nous allons écrire une fonction qui lit un contenu YAML et renvoie un dictionnaire Python.

On se limite à une structure comme :

```yaml
livres:
    - auteur: Jean Dupont
      titre: Je suis le meilleur
      date: 2022
    - auteur: Fanny Rouelle
      titre: C'est moi la meilleure
      date: 2021
```

Qui sera convertie en :

```
[
    {
        "auteur": "Jean Dupont",
        "titre": "Je suis le meilleur",
        "date": "2022",
    },
    {
        "auteur": "Fanny Rouelle",
        "titre": "C'est moi la meilleure",
        "date": "2021"
    }
]
```