SEQUENCE 1

Les deux premières activités ont été extraites et adaptées du livre NSI 1re - Ed. Bordas

ACTIVITE 1 - Constructions élémentaires & langages

Adam a lu que dans la grande majorité des langages de programmation on retrouve les mêmes **constructions élémentaires**. Les deux premières constructions élémentaires auxquelles il s'intéresse sont les **affectations** et les **séquences d'instructions** qui sont présentes dans la fonction écrite en Python, suivante:

```
def conversion(duree_en_minute):
    heure = duree_en_minute // 60
    minutes = duree_en_minute % 60
    return heures, minutes
```

!!! question "Analyser" A quelles lignes peut-on trouver:

```
* une affectation?
* une séquence d'instructions?
```

Parmi les deux fonctions suivantes, laquelle contient une boucle bornée ? une boucle non bornée ? Préciser les **mots clés** utilisés.

```
def estimer_quantite(quantite_initiale, annee):
    resultat = quantite_initiale
    while resultat <= quantite_initiale * 3:
        resultat = resultat * 1.33
        annee = annee + 1
    return annee

def augmentation(quantite_initiale):
    resultat = quantite_initiale
    for i in range(5):
        resultat = resultat * 1.33
    return resultat
```

On considère maintenant les 6 fonctions suivantes.

```
def maximum(a, b):
    if a > b:
        return a
    else:
```

```
        return b

def maximum2(a, b, c, d):
    return maximum(maximum(a, b), maximum(c, d))

def positive(valeur):
    if valeur < 0:
        valeur = -valeur
    return valeur

def conversion(en_bit):
    en_octet = en_bit / 8
    return en_octet

def racine(a):
    x = 1
    for i in range(10):
        x = (x + a / x) / 2
    return x

def compte_bits(valeur):
    nb_bits = 1
    while valeur > 2**nb_bits:
        nb_bits = nb_bits + 1
    return nb_bits
```

!!! question "Analyser" Dans quelle(s) fonction(s) retrouve-t-on:

1. une séquence d'instructions ?
2. une ou des affectations ?
3. un test ?
4. une boucle bornée ?
5. une boucle non bornée ?
6. un appel de fonction ?

Adam s'attache maintenant à écrire certaines fonctions précédentes avec le langage **JavaScript** afin de les comparer à celle écrite en Python. Il commence par déclarer une fonction nommée `maximum()`.

```
function maximum(a, b){
    if (a > b) {
        return a
    } else {
        return b
    }
}
```

!!! question "Analyser" Citer les points communs avec le langage Python et repérer dans cet exemple 3 points particuliers du langage JavaScript.

Adam écrit à présent cette fonction sans les indentations (version 1) et sur une seule ligne (version 2).

=== "Version 1"

```
``` javascript
function maximum(a, b) {
if (a > b) {
return a
} else {
return b
}
}
```
```

=== "Version 2"

```
``` javascript
function maximum(a,b){ if(a>b) { return a } else { return b } }
```
```

!!! question "Analyser" * Sachant que les deux écritures de la fonction `maximum()` sont correctes, quelles hypothèses peut-on formuler quant aux indentations et aux retours à la ligne en javascript ? * Quel caractère est utilisé en Javascript pour remplacer les indentations en python ?

En gardant les indentations qui améliore la lisibilité du code, Adam s'intéresse à présent à la fonction `augmentation()`.

```
function augmentation(quantite_initiale) {
    let resultat = quantite_initiale
    for(let i=0; i<5; i++) {
        resultat = resultat * 1.33
    }
    return resultat
}
```

!!! question "Analyser" * En considérant le programme JavaScript `augmentation()` ci-dessus, citer deux nouveaux points communs et repérer deux différences entre les langages Python et JavaScript. * Formuler une hypothèse sur l'utilité d'écrire le mot clé `let` aux lignes 2 et 3.

Adam réécrit également la première fonction de cette activité, `conversion()`, en Javascript.

```
function conversion(duree_en_minute) {
    let heures = Math.floor(duree_en_minute / 60)
```

```
    let minutes = duree_en_minute % 60
    return [heures, minutes]
}
```

!!! question "Analyser" * En Javascript une fonction ne peut pas renvoyer un couple de valeur (en fait un *n-uplet*). Comment Adam a-t-il choisi de renvoyer le résultat ? * L'opérateur qui calcule le reste de la division euclidienne est identique en python et en javascript: `%`. En revanche, il n'existe pas d'opérateur en javascript pour calculer le quotient de la division euclidienne. Il faut calculer le quotient de la division et récupérer sa partie entière en faisant appel à une fonction native du langage JavaScript qui s'appelle `floor()`. Adam se rappelle avoir déjà utilisé une fonction Python qui avait le même nom. D'une manière générale, quelles difficultés peut-on rencontrer quand on utilise un nouveau langage de programmation ?