Notes de préparation de la correction du QCM-01

NSI Première, sujet type E3C

qkzk

2020/03/24

Thème A : types de base

- QA1. B. sur 32 bits on peut représenter `2**32` nombres.
- QA2. C. utf-8 permet de coder tous les caractères. (majuscules déjà dans ASCII)
- QA3. D. -1 : que des 1. sur un octet : `1111 1111` on décroît à partir de moins un en enlevant 1 à chaque fois.
- QA4. B. Sur 8 bits en complément à deux, on a 256 valeurs possibles de -128 à 127.
- QA5. D. `3F5` contient le chiffre `F` donc il faut au moins 16 valeurs (012345789ABCDEF). Donc c'est de l'hexadécimal
- QA6. C. `1101 1001 + 11 0110 =  1 0000 1111`

Thème B : types construits

- QB1. C. `L=[123456]` donc `L[3]` est la quatrième valeur, 4.
- QB2. D. `T=[[1,2,3],[4,5,6],[7,8,9]]` valeur 7 pour `T[2][0]`
- QB3. C. `[(i, i+1) for i in range(2)]` retourne `[(0, 1), (1, 2)]`
- QB4. C. `[1,4,9,16,25,36]` est la liste des carrés des entiers de 1 à 6 donc `[n*n for n in range(1, 7)]`
- QB5. D. `note = ["do","ré","mi","fa","sol","la","si"]` l'index de "sol" 4, note n'a pas d'index, l'index de "si" est 6, l'index de "mi" est 2.
- QB6. B. on obtient le nombre d'éléments d'une liste python avec `len` donc `hauteur = len(G)` et `largeur=len(G[0])`

Thème C : traitement des données en table

- QC1. A. le nombre de pommes est obtenu avec `T[2]['nombre']`
- QC2. A. `mon_fichier = open("exemple.txt", "r")` ouvre un fichier, dont le contenu est dans la variable `mon_fichier` en mode "lecture seulement". Il faut que le fichier soit dans le même dossier que le script.
- QC3. D. csv, pour **comma** (virgule) separated values, est un format de fichiers texte. Tous les langages modernes ont des librairies pour ouvrir de tels fichier. Les données sont en clair et non chiffrées.

- QC4. B. la quantité de scies et accessibles par `t[1]['quantité']`
- QC5. C. `fonctionMystere(table)` renvoie les éléments `ligne[1]` de chaque ligne de table, si l'élément 2 de `ligne` a pour valeur `'F'` On obtient donc les élements `['Hopper', 'Lovelace']`.

```
def fonctionMystere(table):
  mystere = []
  for ligne in table:
    if ligne[2] == 'F':
      mystere.append(ligne[1])
  return mystere
```

- QC6. C. l'instruction change la valeur de `table[1][2]` pour 5. le second `[1, 2, 3]` devient `[1, 2, 5]`.

Thème D : intéractions entre l'homme et la machine sur le web

- QD1. C. hyperlien vers Educsol `<a href="...">site Educsol</a>`
- QD2. B. `this` fait référence à l'élément du domaine de la page qui transmet l'événement. C'est donc le texte du bouton qui va devenir rouge.
- QD3. D. Question inutile après la QD1, c'est la balise `<a>`
- QD4. A. Le fichier CSS permet de définir les éléments de style de la page.
- QD5. C. L'échange est sécurisé si le protocole est `https`
- QD6. B. javascript et css sont exécutés côté client. html n'est pas exécuté ni interprété mais transcrit, côté client. PHP est exécuté côté serveur.

Thème E : architectures matérielles et systèmes d'exploitation

- QE1. A. De manière générale on accède à la documentation avec `$ man commande`, parfois `$ commande --help` renvoie le descriptif rapide des options
- QE2. D. Tout le local fonctionne, les problème est au delà du réseau local, peut-être dans le routeur. Celui-ci étant la passerelle, les bonnes réponses sont B et D. C'est étrange néanmoins, le routeur fait le lien avec les autres ressources locales (de l'établissement...)
- QE3. A. le moniteur est un périphérique de sortie.
- QE4. B. `cd ..` pour remonter d'un dossier dans l'arborescence.
- QE5. D. Le document est dans le dossier courant on le copie dans un dossier situé 1 étage plus haut donc `cp NSI.txt ../sauvegardes/NSI2.txt`
- QE6. C. copie ce fichier dans le répertoire courant.

Thème F : langages et programmation

- QF1. B. `random.randint(1, 6)` les bornes sont incluses. Première fois que je vois la doc python en français.
- QF2. D. une infinité de tests. On ne peut jamais s'assurer qu'elle fonctionne en la testant.
- QF3. A. puissance(2, 0) va faire échouer la boucle. Il ne se passe rien.

```
def puissance(x, y):
  p = x
```

```
  for i in range(y - 1):
    p = p * x
  return p
```

- QF4. D. `from math import sqrt`
- QF5. B. `calcul(5)` retourne `f(4)` qui vaut `9`

```
def f(x):
  y = 2 * x + 1
  return y

def calcul(x):
  y = x - 1
  return f(y)
```

- QF6. C. montant vaut d'abord capital, ensutie on a ajoute `n*interet`. On s'arrête quand on dépasse cette valeur : `capital + n * interet > 2 * capital`

```
def capital_double(capital, interet):
  montant = capital
  n = 0
  while montant <= 2 * capital:
    montant = montant + interet
    n = n + 1
  return n
```

Thème G : langages et programmation

- QG1. D. l'invariant de boucle est que la somme est `0 + 1 + 2 + ... + N` et `i<N+1` (i augmente une dernière fois)

```
i = 0
somme = 0
while i < N:
  i = i + 1
  somme = somme + 1
```

- QG2. C. `c` augmente quand la valeur de `L[k]` est 2 pour k parcourant la liste. c=3
- QG3. B. code horrible, `max` est déjà une fonction python... et l'indice ne sert à rien. On réalise une comparaison par élément de la liste. Si la longueur est `n` on réalise `n` comparaisons et `n+1` lectures. Chaque lecture est constante (liste python). La complexité est linéaire.

```
def rechercheMaximum(L):
  max = L[0]
  for i in range(len(L)):
    if L[i] > max:
      max = L[i]
    return max
```

- QG4. C. `s = 0 + 1 + 2 + 3 + 4 + 5 = 15` et `i=5`. Au dernier tour de la boucle, `i` et `s` sont incrémentés encore une fois !

```
s = 0
i = 1
while i < 5:
  s = s + 1
  i = i + 1
```

- QG5. C. Le tri par sélection est quadratique (complexité calculatoire en $O(n^2)$). On réalise de l'ordre de 2500^2 comparaisons. Presque exactement la moitié, d'ailleurs.
- QG6. B. On divise 1000 par 2 jusqu'à trouver 1 : (1000, 500, 250, 125, 63, 32, 16, 8, 4, 2, 1) il faut 10 divisions par 2 donc de l'ordre de 10 comparaisons.