TCA101 - 1er semestre : TP 2

Premiers programmes avec Python - conditions, fonctions, boucles

1 Syntaxe Python

1.1 Si, Alors, Sinon

La syntaxe générale est:

```
if <condition>:
  <début partie alors>
  ...
  <fin partie alors>
else:
  <début partie sinon>
  ...
  <fin partie sinon>
```

Exemple d'utilisation:

```
if age < age_min:
  print "Interdit aux moins de", age_min, "ans."
  sys.exit()
else
  print "Ok."
```

1.2 Fonctions

Syntaxe générale:

```
def <nom fonction> (<paramètres séparés par des virgules>):
  <début corps de la fonction>
  ...
  <fin corps de la fonction>
```

Dans le corps de la fonction, `return <quelque chose>` arrête la fonction en lui faisant renvoyer `<quelque chose>`.

Exemple d'utilisation:

```
def is_even (n):
  if (n % 2) == 0:
    return 1
  else:
    return 0
```

1.3 Boucle Tant Que

Syntaxe générale:

```
while <condition>:
  <début corps de la boucle>
  ...
  <fin corps de la boucle>
```

Exemple d'utilisation:

```
i = 0
while i < nb_nodes(G):
  s = node(G, i)
  print s, "est un sommet de", G
  i = i + 1
```

2 Exercices

Exercice 2.1 *Nombres premiers*

Dans votre répertoire `TCA101`, créez un répertoire `TP2` dans lequel vous mettrez vos fichiers de cette séance.

1. Ecrivez une fonction `is_prime_number` qui prend comme paramètre un entier n et qui renvoit 1 (vrai) si n est un nombre premier et 0 (faux) sinon.

2. Tester cette fonction en affichant tous les nombres premiers inférieurs à 1000.

Exercice 2.2 *Degré maximum*
On considère le type abstrait graphe appelé *type abstrait centré sommets.* Il contient les opérations suivantes :

- `nb_nodes(G)` : donne le nombre de sommets du graphe G,
- `node(G, i)` : donne le sommet numéro i du graphe G, les sommets étant numérotés à partir de 0,
- `nb_edges(s)` : donne le nombre d'arêtes ayant le sommet s comme extrémité,
- `edge(s, i)` : donne la $i$ème arête ayant le sommet s comme extrémité, ces arêtes étant numérotées à partir de 0,
- `follow_edge(s, e)` : l'une des extrémités de l'arête e étant le sommet s, donne l'autre sommet extrémité de e,
- `mark_node(s)` : marque le sommet s,
- `unmark_node(s)` : démarque le sommet s,
- `node_marked(s)` : détermine si le sommet s est marqué,
- `mark_edge(e)` : marque l'arête e,
- `unmark_egde(e)` : démarque l'arête e,

- `edge_marked(e)` : détermine si l'arête e est marquée.

Implémentez une fonction `highest_degree` qui prend comme paramètre un graphe G et qui renvoit le degré maximum des sommets du graphe G.

Exercice 2.3 *Sommets voisins*

On considère de nouveau le type abstrait centré sommets.

Ecrivez une fonction `is_neighbor_of` qui prend comme paramètres deux sommets s et t et qui renvoit 1 si s est adjacent à t et 0 sinon.

Exercice 2.4 *Jeu*

On veut implémenter un jeu où le joueur doit deviner un entier déterminé aléatoirement par la machine et compris entre 0 et n.

A chaque essai, le programme précise si le nombre à trouver est plus grand ou plus petit que le nombre entré. Le jeu doit s'arrêter quand le joueur a trouvé le bon nombre et doit afficher le nombre d'essais qui ont été nécessaires.

Ecrivez ce programme en choisissant la valeur de n que vous voulez. Comment faire pour gagner rapidement quand n est grand?

Etendez le programme pour que plusieurs parties puissent être jouées à la suite, et qu'entre chaque partie soit affiché le meilleur score (le plus petit nombre de tentatives) obtenu dans les parties précédentes.