ULB ECOLE POLYTECHNIQUE DE BRUXELLES

Quelques bonnes pratiques de programmation

Traduction d'un problème en programme

1. *Analysez le problème*

- Identifiez clairement ce que sont les **données fournies**, ainsi que les **résultats** attendus (et leurs types) à l'issue du traitement.
- Formalisez une **démarche générale de résolution** par une séquence d'opérations simples.
- Vérifiez que vous **envisagez tous les cas** de figures (en particuliers les cas "limites").

2. *Découpez votre problème en fonctions*

- Chaque fonction doit réaliser ***une* tâche** clairement identifiée.
- Limitez les fonctions à **15 lignes** maximum, sauf dans des cas exceptionnels.
- **Eviter la redondance** dans le code (copier/coller). Si cela arrive, c'est qu'il manque soit une fonction, soit une boucle, soit que des tests conditionnels peuvent être regroupés.
- N'utilisez **pas de variables globales**.
- Veillez à ce que tous les paramètres et variables d'une fonction soient **utilisés** dans cette fonction.
- Pour une fonction qui renvoie un résultat, organisez le code pour qu'il ne contienne qu'**un seul** `return`, placé comme dernière instruction de la fonction.
- Gérez les erreurs éventuelles et/ou précisez le domaine de validité de paramètres. □ Exemple: Utilisez "`raise Exeption`".

3. *Testez le code au fur et à mesure du développement*

- Créez des **scénarios de test**, pour lesquels vous choisissez les données fournies et vous vérifiez que le résultat de la fonction est conforme à ce que vous attendez.
- Vérifiez les cas particuliers et les *conditions aux limites*. □ Exemples: Pour le calcul d'une racine carrée, que se passe-t-il lorsque le paramètre est un nombre négatif ?

Programmation

1. *Style de programmation*

- Utilisez la forme raccourcie `if(is_leap_year(2008))` plutôt que la forme équivalente `if(is_leap_year(2008)==true)`
- Utilisez la forme `return` *`<expression booléenne>`* plutôt que la forme équivalente

```
if <expression booléenne>:
  res = true
else:
  res = false
return res
```

- N'exécutez pas plusieurs fois une fonction alors qu'une exécution suffit.

2. *Quelques erreurs classiques*

- Vous avez oublié d'**initialiser une variable**.
- L'**alignement des blocs** de code n'est pas respecté. □ Exemple: Un `return` dans une boucle plutôt qu'après.

Nommage de variables, fonctions, *etc.*

1. *Utilisez une convention de nommage par catégorie d'entité*

`joined_lower`	pour les **variables** (attributs), et **fonctions** (méthodes)
`ALL_CAPS`	pour les **constantes**
`StudlyCaps`	pour les **classes**

2. *Choisissez bien les noms*

- Utilisez des noms courts et explicites. Pour une variable de stockage : que contient-elle ? Pour un tableau : que contient un élément ?
- Codez en **anglais**.
- Évitez les noms trop proches les uns des autres.
- Utilisez aussi systématiquement que possible les mêmes genres de noms de variables.
  □ Exemples: `i`, `j`, `k` pour des indices, `max_length` pour une variable, `is_even()` pour une fonction, etc.

Documentation du code

1. *Soignez la clarté de votre code*

... c'est la première source de documentation.

- Décrivez bien les arguments des fonctions.
- Donnez des noms de variables qui expriment leur contenu, des noms de fonctions qui expriment ce qu'elles font (cf. règles de nommage ci-dessus).

2. *Utilisez les* `docstrings`

Décrivez brièvement ce que fait la fonction, pas comment elle le fait, et précisez ses entrées et sorties.

3. *Commentez votre code avec parcimonie.*

- Évitez d'indiquer le *fonctionnement* du code dans les commentaires. □ Exemples: Avant l'inscruction "`for car in line:`", ne pas indiquer qu'on va boucler sur tous les caractères de la `line`...
- Évitez de paraphraser le code. N'utilisez les commentaires que lorsque la fonction d'un bout de code est difficile à comprendre.
- Ne modifiez pas les *paramètres* et ne réutilisez pas des variables. □ Exemple: Si vous recevez une borne inférieure `first` et une supérieure `last` et que vous devez itérer de la première à la dernière, n'incrémentez pas `first` dans la boucle, car la signification n'en serait plus claire, mais créez plutôt une variable locale `pos` que vous initialisez à `first`.

Derniers conseils

1. *N'hésitez pas à consulter Internet*

(et autres sources de documentation) pour résoudre vos problèmes concrets et pour répondre à vos questions.