Didactique de l'informatique

M1 MEEF NSI

Manuels scolaires, Exercices, Compétences

Benjamin Wack

Institut national supérieur du professorat et de l'éducation
Académie de Grenoble

2023 – 2024

Retour sur le programme de NSI 1ère

- Identifier des dépendances entre les notions abordées

Retour sur le programme de NSI 1ère

- Identifier des dépendances entre les notions abordées
- Identifier des thèmes pouvant être traités conjointement

Retour sur le programme de NSI 1ère

- Identifier des dépendances entre les notions abordées
- Identifier des thèmes pouvant être traités conjointement
- Et la démarche de projet ?

Manuels scolaires

- Quels types de rubriques retrouve-t-on ?
- Pour quels usages ?

Adaptation d'exercices

- Quelles tâches sont à la charge des élèves dans l'activité ci-contre ?

Applications

Activité 8. Le négatif d'une image N & B

Pour générer le négatif d'une image enregistrée dans un fichier PBM, il suffit de suivre l'algorithme général de traitement d'image pixel par pixel ▸ cours page 159.
Le programme Python Chap8_Negatif.py mis à disposition en téléchargement parcourt en lecture le fichier Chap8_smiley.pbm tout en générant un nouveau fichier nommé Chap8_negatif.pbm.

```
f = open("Chap8_smiley.pbm", "r")
g = open("Chap8_negatif.pbm", "w")
entete = f.readline()
g.write(entete)
definition = f.readline()
g.write(definition)
largeur,hauteur = definition.split(" ")
largeur = int(largeur)
hauteur = int(hauteur)
for lig in range(hauteur):
    for col in range(largeur):
        pix = f.read(1)
        g.write(pix)
    f.read(1)
    g.write('\n')
f.close()
g.close()
```

ouverture en lecture de l'image
ouverture en écriture du négatif
lecture de la 1re ligne qui contient P1
lecture de la 2e ligne avec la définition
pour chaque ligne
pour chaque pixel
lire le pixel de l'image
écrire ce pixel dans le négatif

Télécharger
Chap8_Negatif.py
Chap8_smiley.pbm
Chap8_chatnoir.pbm

1. Après les avoir téléchargés, enregistrer le programme et le fichier image dans le même dossier. Exécuter le programme et visualiser le nouveau fichier généré.

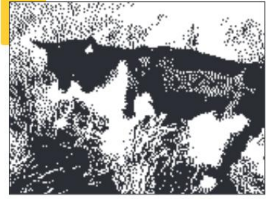
2. a. Modifier le programme pour y inclure le traitement de chaque pixel avant de le réécrire dans le nouveau fichier par l'instruction g.write(pix).
b. Exécuter le programme modifié, puis visualiser le contenu du nouveau fichier Chap8_negatif.pbm.

Pour passer au négatif, il suffit de changer les pixels blancs (0) en noirs (1), et inversement.

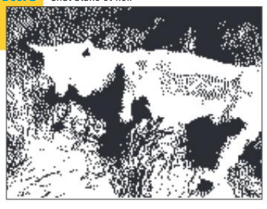
3. Modifier à nouveau le programme pour le tester avec un autre fichier.
On pourra utiliser une image préalablement transformée en PBM avec le logiciel GIMP ▸ Activité 3 partie 2 ou l'image à disposition Chap8_chatnoir.pbm.
L'image du chat noir et blanc ▸ Doc. 1 en négatif donne un chat blanc et noir ▸ Doc. 2.

GIMP ajoute des caractères « *passage à la ligne* » dès que la ligne dépasse 70 caractères, ce qui oblige soit à les enlever, soit à modifier le programme pour les lire.

Doc. 1 Chat noir et blanc



Doc. 2 Chat blanc et noir



Adaptation d'exercices

- Quelles tâches sont à la charge des élèves dans l'activité ci-contre ?
- La réécrire pour leur laisser plus de responsabilité

Applications

Activité 8. Le négatif d'une image N & B

Pour générer le négatif d'une image enregistrée dans un fichier PBM, il suffit de suivre l'algorithme général de traitement d'image pixel par pixel ▸ cours page 159.
Le programme Python Chap8_Negatif.py mis à disposition en téléchargement parcourt en lecture le fichier Chap8_smiley.pbm tout en générant un nouveau fichier nommé Chap8_negatif.pbm.

```
f = open("Chap8_smiley.pbm", "r")
g = open("Chap8_negatif.pbm", "w")
entete = f.readline()
g.write(entete)
definition = f.readline()
g.write(definition)
largeur,hauteur = definition.split(" ")
largeur = int(largeur)
hauteur = int(hauteur)
for lig in range(hauteur):
    for col in range(largeur):
        pix = f.read(1)
        g.write(pix)
    f.read(1)
    g.write('\n')
f.close()
g.close()
```

ouverture en lecture de l'image
ouverture en écriture du négatif
lecture de la 1re ligne qui contient P1
lecture de la 2e ligne avec la définition
pour chaque ligne
pour chaque pixel
lire le pixel de l'image
écrire ce pixel dans le négatif

Télécharger
Chap8_Negatif.py
Chap8_smiley.pbm
Chap8_chatnoir.pbm

Pour passer au négatif, il suffit de changer les pixels blancs (0) en noirs (1), et inversement.

1. Après les avoir téléchargés, enregistrer le programme et le fichier image dans le même dossier. Exécuter le programme et visualiser le nouveau fichier généré.

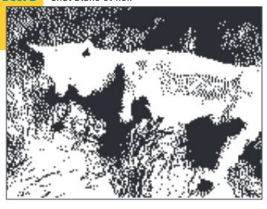
2. a. Modifier le programme pour y inclure le traitement de chaque pixel avant de le réécrire dans le nouveau fichier par l'instruction g.write(pix).
b. Exécuter le programme modifié, puis visualiser le contenu du nouveau fichier Chap8_negatif.pbm.

3. Modifier à nouveau le programme pour le tester avec un autre fichier.
On pourra utiliser une image préalablement transformée en PBM avec le logiciel GIMP ▸ Activité 3 partie 2 ou l'image à disposition Chap8_chatnoir.pbm.
L'image du chat noir et blanc ▸ Doc. 1 en négatif donne un chat blanc et noir ▸ Doc. 2.

GIMP ajoute des caractères « *passage à la ligne* » dès que la ligne dépasse 70 caractères, ce qui oblige soit à les enlever, soit à modifier le programme pour les lire.

Doc. 1 Chat noir et blanc

Doc. 2 Chat blanc et noir



Les compétences en jeu en informatique

Préambule commun aux programmes de NSI

[Cet enseignement] permet de développer des compétences :

- analyser et modéliser un problème en termes de flux et de traitement d'informations ;
- décomposer un problème en sous-problèmes, reconnaître des situations déjà analysées et réutiliser des solutions ;
- concevoir des solutions algorithmiques ;
- traduire un algorithme dans un langage de programmation, en spécifier les interfaces et les interactions, comprendre et réutiliser des codes sources existants, développer des processus de mise au point et de validation de programmes ;
- mobiliser les concepts et les technologies utiles pour assurer les fonctions d'acquisition, de mémorisation, de traitement et de diffusion des informations ;
- développer des capacités d'abstraction et de généralisation.

Les compétences en jeu en informatique

Préambule commun aux programmes de NSI

[Cet enseignement] permet de développer des compétences :

- **analyser** et modéliser un problème en termes de flux et de traitement d'informations ;
- **décomposer** un problème en sous-problèmes, reconnaître des situations déjà analysées et réutiliser des solutions ;
- **concevoir** des solutions algorithmiques ;
- traduire un algorithme dans un langage de programmation, en spécifier les interfaces et les interactions, **comprendre** et réutiliser des codes sources existants, développer des processus de mise au point et de validation de programmes ;
- mobiliser les concepts et les technologies utiles pour assurer les fonctions d'acquisition, de mémorisation, de traitement et de diffusion des informations ;
- développer des capacités d'**abstraction** et de **généralisation**.

Un référentiel plus condensé (d'après C. Declercq)

Évaluer : attribuer mentalement une valeur à un programme donné.
(cf. *analyser* et *comprendre des codes sources existants*)

Un référentiel plus condensé (d'après C. Declercq)

Évaluer : attribuer mentalement une valeur à un programme donné.
(cf. *analyser* et *comprendre des codes sources existants*)

Anticiper : se mettre en posture de programmeur pour décrire l'enchaînement des opérations, avant le début de l'exécution.
(cf. *concevoir des solutions algorithmiques*)

Un référentiel plus condensé (d'après C. Declercq)

Évaluer : attribuer mentalement une valeur à un programme donné.
(cf. *analyser* et *comprendre des codes sources existants*)

Anticiper : se mettre en posture de programmeur pour décrire l'enchaînement des opérations, avant le début de l'exécution.
(cf. *concevoir des solutions algorithmiques*)

Décomposer : transformer un problème complexe en un ensemble de problèmes plus simples équivalent au problème initial.

Un référentiel plus condensé (d'après C. Declercq)

Évaluer : attribuer mentalement une valeur à un programme donné.
(cf. *analyser* et *comprendre des codes sources existants*)

Anticiper : se mettre en posture de programmeur pour décrire l'enchaînement des opérations, avant le début de l'exécution.
(cf. *concevoir des solutions algorithmiques*)

Décomposer : transformer un problème complexe en un ensemble de problèmes plus simples équivalent au problème initial.

Généraliser : inférer un problème général à partir d'une instance, repérer dans un problème particulier la répétition de traitements ou de données suivant un même schéma.

Un référentiel plus condensé (d'après C. Declercq)

Évaluer : attribuer mentalement une valeur à un programme donné. (cf. *analyser* et *comprendre des codes sources existants*)

Anticiper : se mettre en posture de programmeur pour décrire l'enchaînement des opérations, avant le début de l'exécution. (cf. *concevoir des solutions algorithmiques*)

Décomposer : transformer un problème complexe en un ensemble de problèmes plus simples équivalent au problème initial.

Généraliser : inférer un problème général à partir d'une instance, repérer dans un problème particulier la répétition de traitements ou de données suivant un même schéma.

Abstraire : "faire abstraction" des informations non pertinentes et créer des solutions où la manière de résoudre un problème peut être "abstraite" à l'aide d'une interface pertinente. (cf. aussi *modéliser*)

Évaluer

Par exemple calculer le type du résultat, ou calculer le résultat lui-même.
Exemple de question : *quel est le résultat de l'exécution du programme suivant ?*

```
p = [1, 4, 3]
x = 10
p[0] * x**2 + p[1] * x + p[2]
```

Anticiper

Par exemple, imaginer une suite de calculs permettant d'obtenir le résultat.
Exemple d'activité : *écrire une suite d'affectations calculant le même résultat sans utiliser l'opérateur **.*

```
y = 0
y = y * x + p[0]
y = y * x + p[1]
y = y * x + p[2]
y
```

Généraliser

permet de reconnaître un schéma de répétition

```
y = 0
for i in range (3):
    y = y * x + p[i]
y
```

Généraliser

permet de reconnaître un schéma de répétition

```
y = 0
for i in range (3):
    y = y * x + p[i]
y
```

et aussi de passer à une instance plus générale du problème

```
y = 0
for i in range (len(p)):
    y = y * x + p[i]
y
```

Abstraire

Permet de donner une solution qui pourra être utilisée sans en connaître les détails internes.

```
def EvaluerPolynome(p, x):
    '''Evalue le polynome p en un point x
    Le polynome est donné par la liste de
    ses coefficients par degrés décroissants'''

    y = 0
    for i in range (len(p)):
        y = y * x + p[i]
    return(y)

EvaluerPolynome([1, 4, 3], 10)
```

Décomposer

Nécessaire si la question initiale est plus complexe, par exemple « trouver les zéros d'une fonction polynôme quelconque par une méthode numérique ».

On identifie alors plusieurs sous-problèmes pouvant être isolés, comme par exemple :

Décomposer

Nécessaire si la question initiale est plus complexe, par exemple « trouver les zéros d'une fonction polynôme quelconque par une méthode numérique ».

On identifie alors plusieurs sous-problèmes pouvant être isolés, comme par exemple :

- « évaluer un polynôme en un point x »
- « déterminer un intervalle contenant un zéro »
- « calculer une valeur approchée d'un zéro dans un intervalle donné »

Recherche et préparation d'exercices

Dans un des manuels scolaires proposés, choisir un exercice (ou une activité).
Analyser cet exercice :

- connaissances mises en jeu
- compétences mobilisées

Puis proposer une correction *telle que vous la présenteriez à une classe du niveau concerné*.