NSI - Bonnes Pratiques de Programmation

G. VILLEMUS

Lycée Daumier

Septembre 2019

Pourquoi de bonnes pratiques ?

- travail en groupe
- reprise d'un ancien code
- lecture du code d'un autre

Pourquoi de bonnes pratiques ?

- travail en groupe
- reprise d'un ancien code
- lecture du code d'un autre
- lisibilité pour le correcteur ! !

Un code est plus souvent lu qu'exécuté !

Python is the "most powerful language you can still read".

Paul Dubois

PEP 20

PEP 20 connue comme "The Zen of Python"

```
import this
```

PEP 20

PEP 20 connue comme "The Zen of Python"

```
import this
```

Beautiful is better than uggly	Préfère le beau au laid
Explicit is better than implicit	Préfère l'explicite à l'implicite
Simple is better than complex	Préfère le simple au complexe
Complex is better than complicated	Préfère le complexe au compliqué
Flat is better than nested	Préfère le déroulé à l'imbriqué
Sparse is better than dense	Préfère l'aéré au compact

PEP 20 connue comme "The Zen of Python"

```
import this
```

Beautiful is better than uggly	Préfère le beau au laid
Explicit is better than implicit	Préfère l'explicite à l'implicite
Simple is better than complex	Préfère le simple au complexe
Complex is better than complicated	Préfère le complexe au compliqué
Flat is better than nested	Préfère le déroulé à l'imbriqué
Sparse is better than dense	Préfère l'aéré au compact
Readability counts	Prends en compte la lisibilité
Special cases aren't special enough to break the rules	Les cas particuliers ne le sont jamais assez pour violer les règles

Although practicality beats purity	Mais, à la pureté, préfère l'aspect pratique
Errors should never pass silently	Ne pas passe les erreurs sous silence
Unless explicitly silenced	... ou bâillonne-les explicitement.
In the face of ambiguity, refuse the temptation to guess	Face à l'ambiguïté, ne te laisse pas aller à deviner
...	...
Now is better than never	Mieux vaut maintenant que jamais
Although never is often better than *right* now	Cependant jamais est souvent mieux qu'immédiatement.
...	...

PEP 8 - Espaces

A faire :

```
variable = 'bonjour'
if a == b ...
1 + 2
```

plutôt que :

```
variable='bonjour'
if a==b ...
1+2
```

Exceptions :

```
delta = b*b - 4*a*c  #priorites operations
def fonction(argument='valeur') #parametres
2 * (3 + 1)   #crochets et parentheses
```

PEP 8 - Longueur des lignes

Lignes limitées à 80 caractères (largeur petits écrans)

On peut utiliser le chaînage (anti-slash) si besoin.

```
variable = "Respectez la largeur \
           des lignes pour PEP8"
liste = [1, 2, 3,
         4, 5, 6]
```

Les fonctions sont séparées par 2 lignes vides et commentées

(*docstings* avec des triples quotes)

```
def fonction(a, b):
    """Description ... (docstring)"""
    return ...


print(fonction(3, 2))
```

PEP 8 - Noms de variables

Lettres seules en minuscule : boucles et indices.

```
for x in range(10):
    print(x)
```

Lettres minuscules + underscores (snake case) : modules, variables, fonctions, méthodes.

```
ma_variable = 10

def ma_fonction():
    return ...
```

Lettres majuscules + underscores : constantes (pseudo)

```
TAILLE_MAX = 100
```

PEP 8 - Noms et importations

Camel Case : noms de classes

```
class CeciEstUneClasse:
    def methodiq(self):
        ...
```

Exception pour les acronymes :

```
class ClassHTML:
    def ...
```

Un seul import de module par ligne

```
import math
import random

#et pas : import math, random
```

Commentaires

Ils débutent toujours par le symbole # suivi d'un espace.

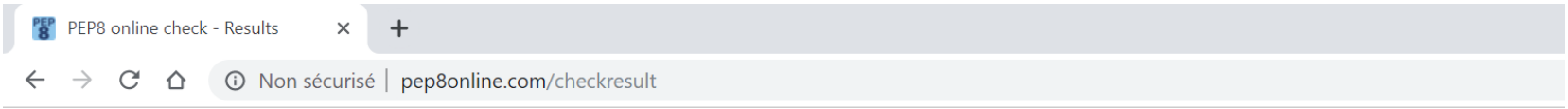
Ils donnent des explications claires sur le code et sont synchronisés avec lui (càd que si le code est modifié, les commentaires doivent l'être aussi) .

Ils sont sur le même niveau d'indentation que le code qu'ils commentent.

Ils sont constitués de phrases complètes, avec une majuscule au début (sauf si le premier mot est une variable) et un point à la fin.

PEP 8 recommande très fortement d'écrire les commentaires **en anglais**.

Contre-exemple



Check results

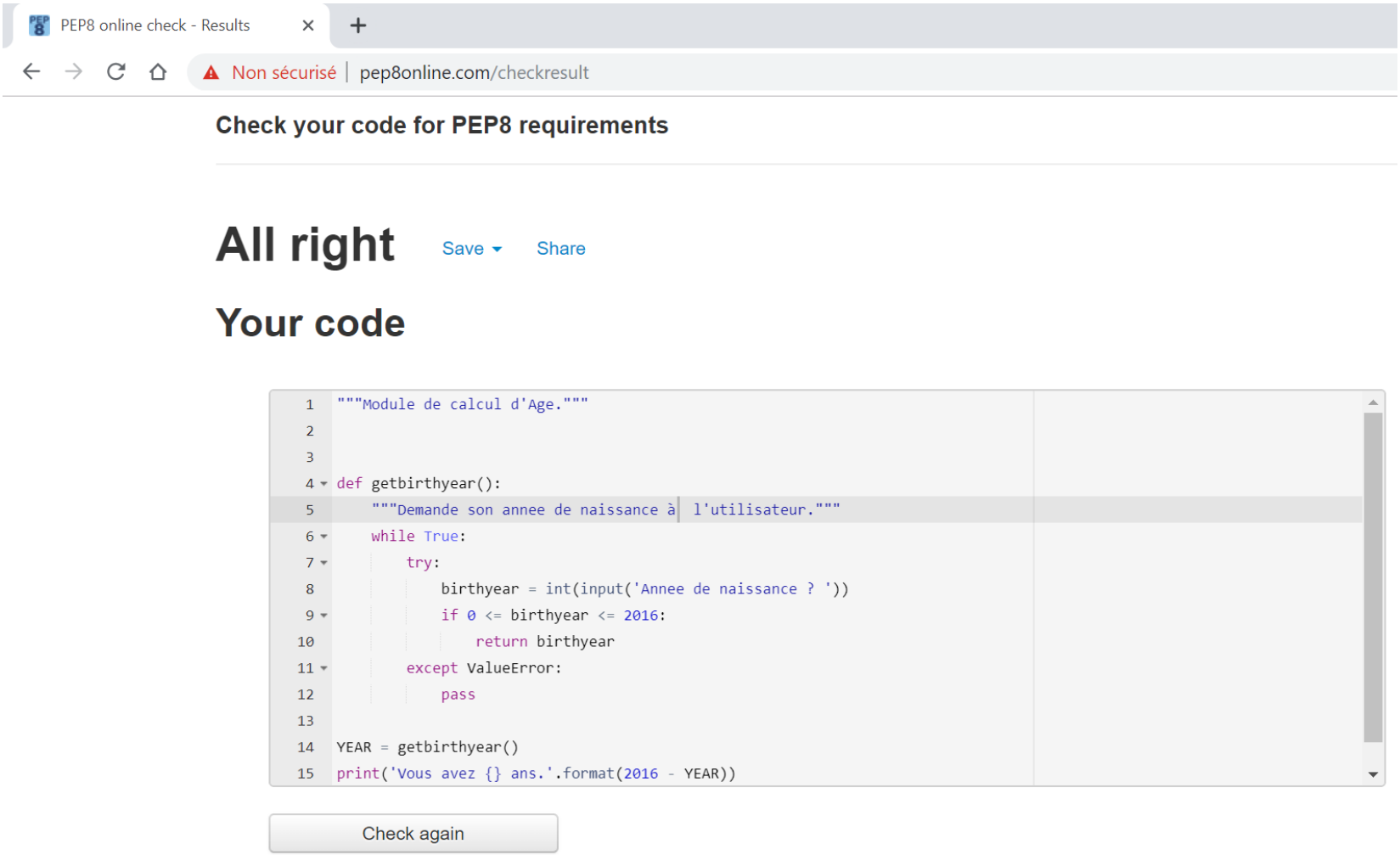
Save ▾ Share

Code	Line	Column	Text
E225	5	22	missing whitespace around operator
E701	6	38	multiple statements on one line (colon)
E272	6	39	multiple spaces before keyword

Your code

```
1  def getbirthyear():
2      """Demande son annee de naissance à  l'utilisateur."""
3      while True:
4          try:
5              birthyear=int(input('Annee de naissance ? '))
6              if 0 <= birthyear <= 2016:                return birthyear
7          except ValueError:
8              pass
9
10 YEAR = getbirthyear()
11 print('Vous avez {} ans.'.format(2016 - YEAR))
12
```

Exemple


PEP8 online check - Results
Non sécurisé | pep8online.com/checkresult
Check your code for PEP8 requirements
All right
Save
Share
Your code
1 """Module de calcul d'Age."""
2
3
4 def getbirthyear():
5 """Demande son annee de naissance à l'utilisateur."""
6 while True:
7 try:
8 birthyear = int(input('Annee de naissance ? '))
9 if 0 <= birthyear <= 2016:
10 return birthyear
11 except ValueError:
12 pass
13
14 YEAR = getbirthyear()
15 print('Vous avez {} ans.'.format(2016 - YEAR))
Check again