

# Notion d'algorithme

## *Boucles et tests avec et sans Python*

Stéphane Gonnord

`stephane@gonnord.org`

`www.mp933.fr`

Lycée du parc - Lyon

Vendredi 4, 11 et 18 octobre 2013

Lycée du parc

### Plan

#### Formaliser des tâches répétitives

Dans la vraie vie

En mathématiques

#### Des maths très élaborées

Additionner

Multiplier

Mais encore...

#### Boucles et tests en Python

Syntaxe des boucles

Syntaxe des tests

#### Exemples et exercices

Reprise des algorithmes précédents

Du rabe

À l'envers !

Exercices

#### Ce qui reste à faire

Correction et terminalison

Complexité

1. Formaliser des tâches répétitives :
  - ▶ des exemples de la vraie vie ;
  - ▶ un peu de mathématiques
2. Des maths un peu élaborées :
  - ▶ L'addition ;
  - ▶ la multiplication ;
  - ▶ etc.
3. Boucles et tests en Python :
  - ▶ boucles ;
  - ▶ tests.
4. Des tas d'exemples et exercices :
  - ▶ primalité, indice de dépassement... ;
  - ▶ up side down.
5. Ce qui reste à faire :
  - ▶ prouver ;
  - ▶ évaluer la complexité.

## Plan

### Formaliser des tâches répétitives

Dans la vraie vie

En mathématiques

### Des maths très élaborées

Additionner

Multiplier

Mais encore...

### Boucles et tests en Python

Syntaxe des boucles

Syntaxe des tests

### Exemples et exercices

Reprise des algorithmes précédents

Du rabe

À l'envers !

Exercices

### Ce qui reste à faire

Correction et terminaison

Complexité

# Un premier exemple gonflé (1/2)

## Exercice

Comment obtenir cet affichage ?

Allo ?

Allo ?

Allo ?

Non mais allo quoi !

► **Algorithme :**

***pour* *i* allant de 10 à 12 faire**

└ Afficher « Allo ? »

Afficher « Non mais allo quoi ! »

► **Programme :**

```
for i in range(10,13):  
    print("Allo ?")  
print("Non mais allo quoi !")
```

# Un premier exemple gonflé (2/2)

## Plan

### Formaliser des tâches répétitives

#### Dans la vraie vie

En mathématiques

### Des maths très élaborées

Additionner

Multiplier

Calculer encore...

### Boucles et tests en Python

Syntaxe des boucles

Syntaxe des tests

### Exemples et exercices

Reprise des algorithmes précédents

Du rabe

À l'envers !

Exercices

### Ce qui reste à faire

Correction et terminaison

Complexité

**pour  $i$  allant de 10 à 12 faire**

└ Afficher « Allo ? »

Afficher « Non mais allo quoi ! »

**pour  $i$  allant de 10 à 12 faire**

└ Afficher « Allo ? »

└ Afficher « Non mais allo quoi ! »

```
for i in range(10,13):
```

```
    print("Allo ?")
```

```
print("Non mais allo quoi !")
```

```
for i in range(10,13):
```

```
    print("Allo ?")
```

```
    print("Non mais allo quoi !")
```

Allo ?

Allo ?

Allo ?

Non mais allo quoi !

Allo ?

Non mais allo quoi !

Allo ?

Non mais allo quoi !

Allo ?

Non mais allo quoi !

# Compter et sommer des machins

## Exercice

Comment compter le nombre d'entiers  $\leq 1000$  qui sont premiers ? Et leur somme ?

```
c ← 0  # un compteur
pour i allant de 1 à 1000 faire
  | si i est premier alors
  |   | c ← c + 1
```

**Résultat : c**

```
s ← 0  # un «sommeur»
pour i allant de 1 à 1000 faire
  | si i est premier alors
  |   | s ← s + i
```

**Résultat : s**

## Plan

### Formaliser des tâches répétitives

Dans la vraie vie

En mathématiques

### Des maths très élaborées

Additionner

Multiplier

Mais encore...

### Boucles et tests en Python

Syntaxe des boucles

Syntaxe des tests

### Exemples et exercices

Reprise des algorithmes précédents

Du rabe

À l'envers !

Exercices

### Ce qui reste à faire

Correction et terminaison

Complexité

# IRL : corriger des copies

- Un premier point de vue :

***pour  $c$  décrivant l'ensemble  $\mathcal{C}$  des copies faire***

- pour  $p$  décrivant l'ensemble  $\mathcal{P}$  des parties du DS faire******
- └ Corriger la partie  $p$  de la copie  $c$  ; noter cette partie*
- └ Noter  $c$*

- Un autre :

***pour  $p$  décrivant  $\mathcal{P}$  faire***

- pour  $c$  décrivant  $\mathcal{C}$  faire******
- └ Corriger la partie  $p$  de la copie  $c$  ; noter cette partie*

***pour  $c$  décrivant  $\mathcal{C}$  faire***

- └ Noter  $c$*

- Il ne s'agit pas de boucles « sur des entiers »

# Exponentiation

## Exercice

Comment calculer  $841^{42}$  avec seulement des multiplications ?

- ▶  $r \leftarrow 841$  ( $r$  va être multiplié par 841... 41 fois)
- ▶  $r \leftarrow r \times 841$  ( $r$  vaut alors  $841^2$ )
- ▶  $r \leftarrow r \times 841$  ( $r$  vaut alors  $841^3$ )
- ▶ ...
- ▶  $r \leftarrow r \times 841$  ( $r$  vaut alors  $841^{42}$ )

## Algorithme

$r \leftarrow 841$

**pour**  $i$  allant de 0 à 40 **faire**

$r \leftarrow r \times 841$

**Résultat** :  $r$

- ▶ La deuxième instruction n'était pas  $r \leftarrow r \times r$
- ▶ Pour  $x^n$ , il vaut mieux partir de  $r \leftarrow 1$  ; pourquoi ?

# Factorielle

## Exercice

Comment calculer  $841!$  avec seulement des multiplications ?

- ▶  $r \leftarrow 1$  ( $r$  va être multiplié par 2, puis 3, ... puis 841)
- ▶  $r \leftarrow r \times 2$  ( $r$  vaut alors 2)
- ▶  $r \leftarrow r \times 3$  ( $r$  vaut alors  $2 \times 3$ )
- ▶ ...
- ▶  $r \leftarrow r \times 841$  ( $r$  vaut alors  $841!$ )

## Algorithme

$r \leftarrow 1$

**pour**  $i$  allant de 2 à 841 **faire**

$r \leftarrow r \times i$

**Résultat** :  $r$



# IRL : Mr G. et la trigonométrie

- *Un algorithme pratiqué par tous :*

Faire apprendre un formulaire basique de trigonométrie

Poser deux ou trois formules en DS

**tant que** *ce n'est pas bien appris* **faire**  
    grogner ;

    exiger que ce soit travaillé mieux que ça ;

    poser deux ou trois formules en DS

Passer à la suite

- Problème de la terminaison...

## Exercice

Formaliser l'algorithme, dans le cas (improbable) où la trigo ne serait *jamais* bien apprise.

### Plan

#### Formaliser des tâches répétitives

Dans la vraie vie

En mathématiques

#### Des maths très élaborées

Additionner

Multiplier

Mais encore...

#### Boucles et tests en Python

Syntaxe des boucles

Syntaxe des tests

#### Exemples et exercices

Reprise des algorithmes précédents

Du rabe

À l'envers !

Exercices

#### Ce qui reste à faire

Correction et terminaison

Complexité

# Encore une boucle « Tant que »

## Exercice

Déterminer le plus petit entier  $n$  tel que  $n! \geq 10^{10}$

- Premier algorithme :

$n \leftarrow 1$

**tant que**  $n! < 10^{10}$  **faire**

$n \leftarrow n + 1$

**Résultat** :  $n$

- Et si on s'interdit la fonction factorielle ?

$n \leftarrow 1$

$f \leftarrow 1$  #  $f$  vaudra  $n!$  à (presque) tout moment

**tant que**  $f < 10^{10}$  **faire**

$n \leftarrow n + 1$

$f \leftarrow f \times n$

**Résultat** :  $n$

## Plan

### Formaliser des tâches répétitives

Dans la vraie vie

En mathématiques

### Des maths très élaborées

Additionner

Multiplier

Mais encore...

### Boucles et tests en Python

Syntaxe des boucles

Syntaxe des tests

### Exemples et exercices

Reprise des algorithmes précédents

Du rabe

À l'envers !

Exercices

### Ce qui reste à faire

Correction et terminaison

Complexité

# Addition : à deux chiffres (1/3)

## Exercice

Calculer  $62 + 15$

► Première solution :

```
>>> 62 + 15
```

```
77
```

► Et sinon :

$$\begin{array}{r} 62 \\ + 15 \\ \hline ? ? \end{array} \quad \begin{array}{r} 62 \\ + 15 \\ \hline ? 7 \end{array} \quad \begin{array}{r} 62 \\ + 15 \\ \hline 77 \end{array}$$

# Addition : à deux chiffres (2/3)

## Exercice

Calculer  $68 + 15$

$$\begin{array}{r} 68 \\ + 15 \\ \hline ? ? \end{array} \quad \begin{array}{r} \overset{1}{6}8 \\ + 15 \\ \hline ? 3 \end{array} \quad \begin{array}{r} \overset{1}{6}8 \\ + 15 \\ \hline 83 \end{array}$$

Algorithme général ?

$$\begin{array}{r} a \ b \\ + c \ d \\ \hline e \ f \ g \end{array}$$

# Addition : à deux chiffres (3/3)

$$\begin{array}{r} a \ b \\ + \ c \ d \\ \hline e \ f \ g \end{array}$$

**Entrées** :  $a, b, c, d$

$s_1 \leftarrow b + d$

**si**  $s_1 < 10$  **alors**

$g \leftarrow s_1$  # pas de retenue  
     $r \leftarrow 0$

**sinon**

$g \leftarrow s_1 - 10$  # retenue de 1  
     $r \leftarrow 1$

$s_2 \leftarrow r + a + c$

**si**  $s_2 < 10$  **alors**

$(e, f) \leftarrow (0, s_2)$

**sinon**

$(e, f) \leftarrow (1, s_2 - 10)$

**Résultat** :  $d, e, f$

## Stéphane Gonnord

- ▶ nature des entrées...
- ▶ et du résultat ;
- ▶ propagation d'une retenue...
- ▶ qu'on peut oublier après utilisation.

# Multiplication : $n$ et 1 chiffres

## Exercice

Calculer  $841 \times 7$

$$\begin{array}{r} 841 \\ \times 7 \\ \hline \end{array} \quad \begin{array}{r} 841^0 \\ \times 7 \\ \hline \end{array} \quad \begin{array}{r} 841^{20} \\ \times 7 \\ \hline \end{array} \quad \begin{array}{r} 841^5 \\ \times 7 \\ \hline \end{array}$$

- ▶ Entrées, résultat ;
- ▶ une boucle ;
- ▶ propagation d'une retenue.

# Multiplication : $n$ et $p$ chiffres

## Exercice

Calculer  $841 \times 57$

8 4 1	8 4 1	8 4 1	
x 5 7	x 5 7	x 5 7	
-----	-----	-----	-----
? ? ?	5 8 8 7	5 8 8 7	8 4 1
			x 5 7
			-----
			5 8 8 7
		0 4 2	
			-----
			5 8 8 7

- ▶ Une boucle...
- ▶ ou deux ;
- ▶ choisir le niveau de découpage du problème.



# Soustraction

## Exercice

Calculer  $857 - 41$

$$\begin{array}{r} 857 \\ - 41 \\ \hline \end{array} \quad \begin{array}{r} 857 \\ - 41 \\ \hline \end{array} \quad \begin{array}{r} 857 \\ - 41 \\ \hline \end{array} \quad \begin{array}{r} 857 \\ - 41 \\ \hline \end{array}$$
$$\begin{array}{r} ? \\ ? \\ ? \end{array} \quad \begin{array}{r} ? \\ ? \\ 6 \end{array} \quad \begin{array}{r} ? \\ 1 \\ 6 \end{array} \quad \begin{array}{r} 8 \\ 1 \\ 6 \end{array}$$

## Exercice

Calculer  $841 - 57$

$$\begin{array}{r} 841 \\ - 57 \\ \hline \end{array} \quad \begin{array}{r} 841 \\ - 57 \\ \hline \end{array} \quad \begin{array}{r} 841 \\ - 57 \\ \hline \end{array} \quad \begin{array}{r} 841 \\ - 57 \\ \hline \end{array}$$
$$\begin{array}{r} ? \\ ? \\ ? \end{array} \quad \begin{array}{r} ? \\ ? \\ 4 \end{array} \quad \begin{array}{r} ? \\ 8 \\ 4 \end{array} \quad \begin{array}{r} 7 \\ 8 \\ 4 \end{array}$$

## Exercice

Formaliser un algorithme de soustraction

Notion d'algorithme

Stéphane Gonnord

Plan

Formaliser des tâches répétitives

Dans la vraie vie

En mathématiques

Des maths très élaborées

Additionner

Multiplier

Mais encore...

Boucles et tests en Python

Syntaxe des boucles

Syntaxe des tests

Exemples et exercices

Reprise des algorithmes précédents

Du rabe

À l'envers !

Exercices

Ce qui reste à faire

Correction et terminaison

Complexité

# Division

## Exercice

Réaliser la division euclidienne de 841 par 3 puis par 57.

```
>>> 841/3, 841%3  
(280, 1)  
>>> 841/57, 841%57  
(14, 43)
```

## Exercice (difficile)

Formaliser un algorithme de division euclidienne d'entiers.

*On pourra utiliser une boîte noire prenant  $a$  et  $b$  avec  $0 \leq a \leq 99$  et  $1 \leq b \leq 9$ , renvoyant le plus grand entier  $q$  tel que  $bq \leq a$ .*

# Boucles énumératives : syntaxe

L'indentation marque le bloc exécuté.

- Entiers décrivant un domaine régulier :

```
for i in range(10):  
    ...  
    ...  
for i in range(5, 10):  
for i in range(20, 30, 3):
```

- Énumération sur une liste ou un ensemble :

```
for x in [1, 20, 1, -5]:  
    ...  
for x in {-1, 15, 10}:
```

- Et même sur les lignes d'un fichier !

```
toto = open('un_fichier.txt', 'r')  
for ch in toto:  
    ...
```

# Boucles énumératives : exemples

## ► Sur un range :

```
>>> for i in range(10, 1, -3):  
    print(i**2)  
  
100  
49  
16
```

## ► Sur des listes/ensembles :

```
>>> for x in [1, 3, 1]:  
    print(x**2)  
  
1  
9  
1  
>>> for x in {1, 3, 1}:  
    print(x**2)  
  
1  
9
```

# Boucles conditionnelles

« *Tant qu'il reste des M&M's, on en reprend* »

- ▶ L'indentation marque le bloc ;
- ▶ Syntaxe :

```
while <condition> :  
    <instr 1>  
    ...  
    <instr n>  
<ensuite>
```

- ▶ Exemple moisi :

```
i = 10  
while i < 20:  
    i = i+3  
    print(i)  
print("Maintenant, i vaut "+str(i))
```

## Exercice

Qu'est-ce qui va être affiché ? Pourquoi cet exemple est moisi ?

### Plan

#### Formaliser des tâches répétitives

Dans la vraie vie

En mathématiques

#### Des maths très élaborées

Additionner

Multiplier

Mais encore...

#### Boucles et tests en Python

Syntaxe des boucles

Syntaxe des tests

#### Exemples et exercices

Reprise des algorithmes précédents

Du rabe

À l'envers !

Exercices

#### Ce qui reste à faire

Correction et terminaison

Complexité

# Boucles conditionnelles

- Encore un exemple moisi :

```
i, j = 10, 20
while j < 20:
    i = i+3
    print(i+j)
print("Maintenant, i vaut "+str(i))
```

## Exercice

Qu'est-ce qui va être affiché ?

- Plus intéressant :

```
a, cpt = 2020, 0
while a % 2 == 0:
    cpt = cpt+1
    a = a/2
```

## Exercice

Que vaut  $i$  à la sortie de boucle ? Et  $a$  ?

# Tests : syntaxe

- L'indentation gnagnagna

- Syntaxe :

```
if <condition> :  
    <instr 1>  
    ...  
    <instr n>  
<ensuite>
```

- Possibilité (mais non obligation !) d'une alternative :

```
if <condition> :  
    <instr 1>  
    ...  
else:  
    <instr 1>  
    ...  
<ensuite>
```

# Tests : exemples

- ▶ S'il pleut, alors je prends un parapluie.
- ▶ Si j'ai colle de physique demain alors je bosse mon cours de physique ; sinon je bosse mon cours de maths.
- ▶ Si j'ai colle de physique demain alors je bosse mon cours de physique ; sinon : si j'ai colle de maths demain, alors je bosse mon cours de maths, et sinon je vais courir au parc.
- ▶ Si j'ai colle de physique demain alors je bosse mon cours de physique, et si j'ai colle de maths demain, alors je bosse mon cours de maths.

## Exercice

Que se passe t-il dans les trois derniers exemples selon les colles qu'on a le lendemain ? (il y a donc 12 réponses attendues !).

*On fera l'hypothèse que les conditions suffisantes énoncées sont nécessaires...*

## Plan

### Formaliser des tâches répétitives

Dans la vraie vie

En mathématiques

### Des maths très élaborées

Additionner

Multiplier

Mais encore...

### Boucles et tests en Python

Syntaxe des boucles

Syntaxe des tests

### Exemples et exercices

Reprise des algorithmes précédents

Du rabe

À l'envers !

Exercices

### Ce qui reste à faire

Correction et terminaison

Complexité



# Nombre et somme de premiers

## Exercice

Déterminer le nombre d'entiers inférieurs à 1000 qui sont premiers, ainsi que leur somme.

### ► Algorithme :

$c \leftarrow 0$  ;  $s \leftarrow 0$  # le compteur et le sommeur

**pour**  $i$  allant de 1 à 1000 **faire**

**si**  $i$  est premier **alors**

$c \leftarrow c + 1$

$s \leftarrow s + i$

### ► Programme :

```
compteur, somme = 0, 0
for n in range(1000):
    if est_premier(n):
        compteur = compteur+1
        somme = somme+n
```

### ► Résultat :

```
>>> compteur, somme
(168, 76127)
```

## Plan

### Formaliser des tâches répétitives

Dans la vraie vie

En mathématiques

### Des maths très élaborées

Additionner

Multiplier

Mais encore...

### Boucles et tests en Python

Syntaxe des boucles

Syntaxe des tests

### Exemples et exercices

Reprise des algorithmes précédents

Du rabe

À l'envers !

Exercices

### Ce qui reste à faire

Correction et terminalison

Complexité

# Calcul de $841^{42}$

## Exercice

Comment calculer  $841^{42}$  avec seulement des multiplications ?

► Algorithme :

$r \leftarrow 1$

**pour**  $i$  allant de 0 à 41 **faire**

$r \leftarrow r \times 841$

► Programme :

```
r = 1
for i in range(42):
    r *= 841 # Rappel : c'est comme r = r*841
```

► Résultat :

```
>>> r - 841**42
0L
```

# Calcul de 841!

## Exercice

Comment calculer 841! avec seulement des multiplications ?

► Algorithme :

$r \leftarrow 1$

**pour**  $i$  allant de 1 à 841 **faire**

$r \leftarrow r \times i$

► Programme :

```
r = 1
```

```
for i in range(1, 842):
```

```
    r *= i
```

► Résultat :

```
>>> r - math.factorial(841)
```

```
0L
```

## Plan

### Formaliser des tâches répétitives

Dans la vraie vie

En mathématiques

### Des maths très élaborées

Additionner

Multiplier

Mais encore...

### Boucles et tests en Python

Syntaxe des boucles

Syntaxe des tests

### Exemples et exercices

Reprise des algorithmes précédents

Du rabe

À l'envers !

Exercices

### Ce qui reste à faire

Correction et terminaison

Complexité

# Un dépassement

## Exercice

Déterminer le plus petit entier  $n$  tel que  $n! \geq 10^{10}$

► Algorithme :

$n \leftarrow 1 ; f \leftarrow 1$  #  $f$  vaudra  $n!$  à (presque) tout moment

**tant que**  $f < 10^{10}$  **faire**

```
|    $n \leftarrow n + 1$   
|    $f \leftarrow f \times n$ 
```

► Programme :

```
n, facto = 1, 1  
while facto < 10**10:  
    n += 1  
    facto *= n
```

► Résultat :

```
>>> n, facto  
(14, 87178291200L)  
>>> math.factorial(14)  
87178291200L
```

## Plan

### Formaliser des tâches répétitives

Dans la vraie vie

En mathématiques

### Des maths très élaborées

Additionner

Multiplier

Mais encore...

### Boucles et tests en Python

Syntaxe des boucles

Syntaxe des tests

### Exemples et exercices

Reprise des algorithmes précédents

Du rabe

À l'envers !

Exercices

### Ce qui reste à faire

Correction et terminaison

Complexité

# Addition de deux entiers

```
def addition(a,b):  
    """ calcul de c=a+b; représentation avec une liste de  
    On suppose que len(a) = len(b) """  
    n = len(a)  
    c = [] # on va adjoindre successivement les décimales  
    r = 0  
    for i in range(n):  
        s = a[i]+b[i]+r  
        if s>9:  
            c.append(s-10)  
            r = 1  
        else:  
            c.append(s)  
            r = 0  
    if r == 1:  
        c.append(1)  
    return c
```

```
>>> addition([7,8,9,3,2,1],[6,5,4,6,5,4])  
[3, 4, 4, 0, 8, 5]
```

# Recherche de diviseurs

## Exercice

Déterminer les diviseurs de 9991

### ► Algorithme :

**pour** *d allant de 1 à 9991* **faire**

**si** *d divise 9991* **alors**

        Afficher *d*

### ► Programme :

```
for d in range(1, 9992):  
    if 9991%d ==0:  
        print(d)
```

### ► Résultat :

```
1  
97  
103  
9991
```

### ► Et pour les mettre dans une liste ?

## Plan

### Formaliser des tâches répétitives

Dans la vraie vie

En mathématiques

### Des maths très élaborées

Additionner

Multiplier

Mais encore...

### Boucles et tests en Python

Syntaxe des boucles

Syntaxe des tests

### Exemples et exercices

Reprise des algorithmes précédents

**Du rabe**

À l'envers !

Exercices

### Ce qui reste à faire

Correction et terminalison

Complexité

# Primalité (1/2)

## Exercice

$N = 10002200057$  est-il premier ?

- ▶ Algorithme 1 :

**pour**  $d$  allant de 2 à  $N - 1$  **faire**

**si**  $d$  divise  $N$  **alors**

        Afficher « Oulala,  $N$  n'est pas premier »

- ▶ Bof...

- ▶ Algorithme 2 :

`Premier <- True # jusqu'à preuve du contraire !`

**pour**  $d$  allant de 2 à  $N - 1$  **faire**

**si**  $d$  divise  $N$  **alors**

`Premier <- False`

- ▶ Améliorations :

- ▶ s'arrêter à  $\sqrt{N}$  ;
- ▶ s'arrêter si `Premier` est passé à `False`

# Primalité (2/2)

- ▶ Algorithme 3 : s'arrêter à  $\sqrt{N}$   
**pour**  $d$  allant de 2 à  $\lfloor \sqrt{N} \rfloor$  **faire**  
    ...
- ▶ Algorithme 4 : s'arrêter si Premier est passé à False  
     $d \leftarrow 2$   
    **tant que**  $d \leq \sqrt{N}$  **et** Premier **faire**  
        **si**  $d$  **divise**  $N$  **alors**  
            Premier  $\leftarrow$  False  
             $d \leftarrow d + 1$

- ▶ Programme :

```
n, premier, d = 10002200057, True, 2
while d<=sqrt(n) and premier:
    if n % d == 0:
        premier = False
    d += 1
```

- ▶ Résultat :

```
>>> d
100004
>>> 1000003*1000019
1000022000057L
```

## Plan

### Formaliser des tâches répétitives

Dans la vraie vie

En mathématiques

### Des maths très élaborées

Additionner

Multiplier

Mais encore...

### Boucles et tests en Python

Syntaxe des boucles

Syntaxe des tests

### Exemples et exercices

Reprise des algorithmes précédents

#### Du rabe

À l'envers !

Exercices

### Ce qui reste à faire

Correction et terminalison

Complexité



# Reverse engineering

- ▶ Après l'exécution du programme suivant, que vaut  $s$  ?

```
s = 0
for i in range(101):
    for j in range(101):
        s = s+(i+j)**2
```

- ▶ Montrer que le programme suivant va terminer :

```
i, cpt = 5000, 0
while i>0:
    i = i//2
    cpt = cpt+1
```

Que valent  $i$  et  $cpt$  après exécution ?

- ▶ Que se passe-t-il à l'exécution de ce programme ?

```
n = 10
while n>1:
    print(n)
    if n%2 ==0:
        n = n/2
    else:
        n = 3*n+1
```

Que se passe-t-il si  $n = 1000$  au départ ? Et pour  $n$  quelconque ?

## Plan

### Formaliser des tâches répétitives

Dans la vraie vie

En mathématiques

### Des maths très élaborées

Additionner

Multiplier

Mais encore...

### Boucles et tests en Python

Syntaxe des boucles

Syntaxe des tests

### Exemples et exercices

Reprise des algorithmes précédents

Du rabe

**À l'envers !**

Exercices

### Ce qui reste à faire

Correction et terminalaison

Complexité

# Quelques exercices supplémentaires

- ▶ Écrire un programme (une fonction) réalisant la soustraction de deux entiers donnés par des listes de décimales (supposées de même taille).
- ▶ Même chose pour la multiplication de deux entiers :
  - ▶ avec  $n$  et 1 décimales ;
  - ▶ avec  $n$  et  $p$  décimales.
- ▶ S'attaquer à la division !

## Plan

### Formaliser des tâches répétitives

Dans la vraie vie

En mathématiques

### Des maths très élaborées

Additionner

Multiplier

Mais encore...

### Boucles et tests en Python

Syntaxe des boucles

Syntaxe des tests

### Exemples et exercices

Reprise des algorithmes précédents

Du rabe

À l'envers !

**Exercices**

### Ce qui reste à faire

Correction et terminaison

Complexité

# Prouver la correction (1/2)

## Exercice

Montrer que le code suivant fait ce qu'on attendait de lui :

```
compteur = 0
for n in range(1001):
    if est_premier(n):
        compteur = compteur+1
```

- ▶ Notion d'*invariant de boucle*.
- ▶ Trouver le bon (délicat !);
- ▶ et le prouver.
- ▶ Ici par exemple : « À la fin de l'exécution du corps de boucle, compteur vaut le nombre d'indices  $k \leq n$  tels que `est_premier(k)` a renvoyé `True` »
- ▶ Il reste à conclure !

## Plan

### Formaliser des tâches répétitives

Dans la vraie vie

En mathématiques

### Des maths très élaborées

Additionner

Multiplier

Mais encore...

### Boucles et tests en Python

Syntaxe des boucles

Syntaxe des tests

### Exemples et exercices

Reprise des algorithmes précédents

Du rabe

À l'envers !

Exercices

### Ce qui reste à faire

Correction et terminaison

Complexité

# Prouver la correction (2/2)

## Exercice

Montrer qu'après l'exécution du script suivant,  $r_1$  et  $r_2$  valent respectivement  $841^{42}$  et  $841!$  :

```
r1, r2 = 1, 1
for i in range(42):
    r1 = r1*841
for i in range(2,842):
    r2 = r2*i
```

Fastoche !

## Exercice

Montrer qu'après l'exécution du script suivant,  $r$  vaut  $841^{42}$

```
r, p, n = 1, 841, 42
while n > 0:
    if n%2 == 1:
        r *= p
    p *= p
    n //= 2
```

Peut-être un peu moins fastoche !

## Plan

### Formaliser des tâches répétitives

Dans la vraie vie

En mathématiques

### Des maths très élaborées

Additionner

Multiplier

Mais encore...

### Boucles et tests en Python

Syntaxe des boucles

Syntaxe des tests

### Exemples et exercices

Reprise des algorithmes précédents

Du rabe

À l'envers !

Exercices

### Ce qui reste à faire

Correction et terminaison

Complexité

# Prouver la terminaison

- ▶ On suppose que  $n$  a été affecté (et vaut un entier strictement positif). Montrer que la boucle suivante va terminer :

```
while n>0:  
    n = n-10
```

- ▶ Même chose ici :

```
while n>0:  
    n = n // 2
```

- ▶ Et encore ici :

```
while n>1:  
    if n%2 ==0:  
        n = n/2  
    else:  
        n = 3*n+1
```

*Attention, il y a un piège :-)*

## Plan

### Formaliser des tâches répétitives

Dans la vraie vie

En mathématiques

### Des maths très élaborées

Additionner

Multiplier

Mais encore...

### Boucles et tests en Python

Syntaxe des boucles

Syntaxe des tests

### Exemples et exercices

Reprise des algorithmes précédents

Du rabe

À l'envers !

Exercices

### Ce qui reste à faire

Correction et terminaison

Complexité

# Évaluer la complexité (1/2)

- ▶ On suppose que  $n$  a été affecté (et vaut un entier strictement positif). Combien d'*opérations élémentaires* le programme suivant va-t-il exécuter ?

```
s = 0
for i in range(n):
    s = s+i**2
```

- ▶ Même chose ici :

```
s = 0
for i in range(n):
    for j in range(i,n):
        s = s+j**2
```

- ▶ Et ici :

```
while n<10**10:
    n = n * 2
```

## Plan

### Formaliser des tâches répétitives

Dans la vraie vie

En mathématiques

### Des maths très élaborées

Additionner

Multiplier

Mais encore...

### Boucles et tests en Python

Syntaxe des boucles

Syntaxe des tests

### Exemples et exercices

Reprise des algorithmes précédents

Du rabe

À l'envers !

Exercices

### Ce qui reste à faire

Correction et terminaison

Complexité

# Évaluer la complexité (2/2)

- On suppose que  $n$  a été affecté (et vaut un entier strictement positif). Combien d'opérations élémentaires le programme suivant va-t-il exécuter ?

```
while n>0:  
    n = n // 2
```

- Et ici ?

```
while n>1:  
    if n%2 ==0:  
        n = n/2  
    else:  
        n = 3*n+1
```

## Plan

### Formaliser des tâches répétitives

Dans la vraie vie

En mathématiques

### Des maths très élaborées

Additionner

Multiplier

Mais encore...

### Boucles et tests en Python

Syntaxe des boucles

Syntaxe des tests

### Exemples et exercices

Reprise des algorithmes précédents

Du rabe

À l'envers !

Exercices

### Ce qui reste à faire

Correction et terminaison

Complexité

# End game

Merci de votre attention !



*XKCD 1247 : « The mother of all suspect files »  
(Better change the URL to `https` before downloading.)*

Notion d'algorithme

Stéphane Gonnord

Plan

Formaliser des  
tâches répétitives

Dans la vraie vie

En mathématiques

Des maths très  
élaborées

Additionner

Multiplier

Mais encore...

Boucles et tests en  
Python

Syntaxe des boucles

Syntaxe des tests

Exemples et  
exercices

Reprise des algorithmes  
précédents

Du rabe

À l'envers !

Exercices

Ce qui reste à faire

Correction et terminaison

Complexité