Programmation et mathématique

INFO1 - semaines 36 à 42

Guillaume CONNAN

septembre 2015

Iut de Nantes - Dpt d'informatique

Sommaire

Sommaire

- Des diapos en amphi : on ne les recopie pas mais on les commente par écrit. Elles sont disponibles sur HTTP://INFORMATHIX.TUXFAMILY.ORG. Pas de poly de cours.

vos travaux écrits et votre investissement en amphi et en TD/TP.

- Des diapos en amphi : on ne les recopie pas mais on les commente par écrit. Elles sont disponibles sur HTTP://INFORMATHIX.TUXFAMILY.ORG. Pas de poly de cours.

vos travaux écrits et votre investissement en amphi et en TD/TP.

- Des diapos en amphi : on ne les recopie pas mais on les commente par écrit. Elles sont disponibles sur HTTP://INFORMATHIX.TUXFAMILY.ORG.
  Pas de poly de cours.
- En TD/TP : un petit poly avec des questions. Travail d'équipe avec l'intervenant pour y répondre.

vos travaux écrits et votre investissement en amphi et en TD/TP.

- Des diapos en amphi : on ne les recopie pas mais on les commente par écrit. Elles sont disponibles sur HTTP://INFORMATHIX.TUXFAMILY.ORG. Pas de poly de cours.
- En TD/TP : un petit poly avec des questions. Travail d'équipe avec l'intervenant pour y répondre.
- Parfois il faudra répondre à un petit QCM en ligne pour obtenir l'énoncé du TP.
- Des petits exos en amphi et en TD, annoncés à l'avance ou pas.

vos travaux écrits et votre investissement en amphi et en TD/TP.

- Des diapos en amphi : on ne les recopie pas mais on les commente par écrit. Elles sont disponibles sur HTTP://INFORMATHIX.TUXFAMILY.ORG. Pas de poly de cours.
- En TD/TP : un petit poly avec des questions. Travail d'équipe avec l'intervenant pour y répondre.
- Parfois il faudra répondre à un petit QCM en ligne pour obtenir l'énoncé du TP.
- Des petits tests en amphi et en TD, annoncés à l'avance ou non.
- Des travaux personnels à rendre par écrit ou sous forme électronique.
- Des devoirs en amphi reprenant ce qui a été étudié. Vous avez droit à vos notes manuscrites : résumé, synthèse, recherches,...
- Puisqu'il en faut une, il y aura une note finale qui tentera de représenter vos travaux écrits et votre investissement en amphi et en TD/TP.

- Des diapos en amphi : on ne les recopie pas mais on les commente par écrit. Elles sont disponibles sur HTTP://INFORMATHIX.TUXFAMILY.ORG. Pas de poly de cours.
- En TD/TP : un petit poly avec des questions. Travail d'équipe avec l'intervenant pour y répondre.
- Parfois il faudra répondre à un petit QCM en ligne pour obtenir l'énoncé du TP.
- Des petits tests en amphi et en TD, annoncés à l'avance ou non.
- Des travaux personnels à rendre par écrit ou sous forme électronique.
- Des devoirs en amphi reprenant ce qui a été étudié. Vous avez droit à vos notes **manuscrites** : résumé, synthèse, recherches,...
- Puisqu'il en faut une, il y aura une note finale qui tentera de représenter vos travaux écrits et votre investissement en amphi et en TD/TP.

- Des diapos en amphi : on ne les recopie pas mais on les commente par écrit. Elles sont disponibles sur `HTTP://INFORMATHIX.TUXFAMILY.ORG`. Pas de poly de cours.
- En TD/TP : un petit poly avec des questions. Travail d'équipe avec l'intervenant pour y répondre.
- Parfois il faudra répondre à un petit QCM en ligne pour obtenir l'énoncé du TP.
- Des petits tests en amphi et en TD, annoncés à l'avance ou non.
- Des travaux personnels à rendre par écrit ou sous forme électronique.
- Des devoirs en amphi reprenant ce qui a été étudié. Vous avez droit à vos notes **manuscrites** : résumé, synthèse, recherches,...
- Puisqu'il en faut une, il y aura une note finale qui tentera de représenter vos travaux écrits et votre investissement en amphi et en TD/TP.

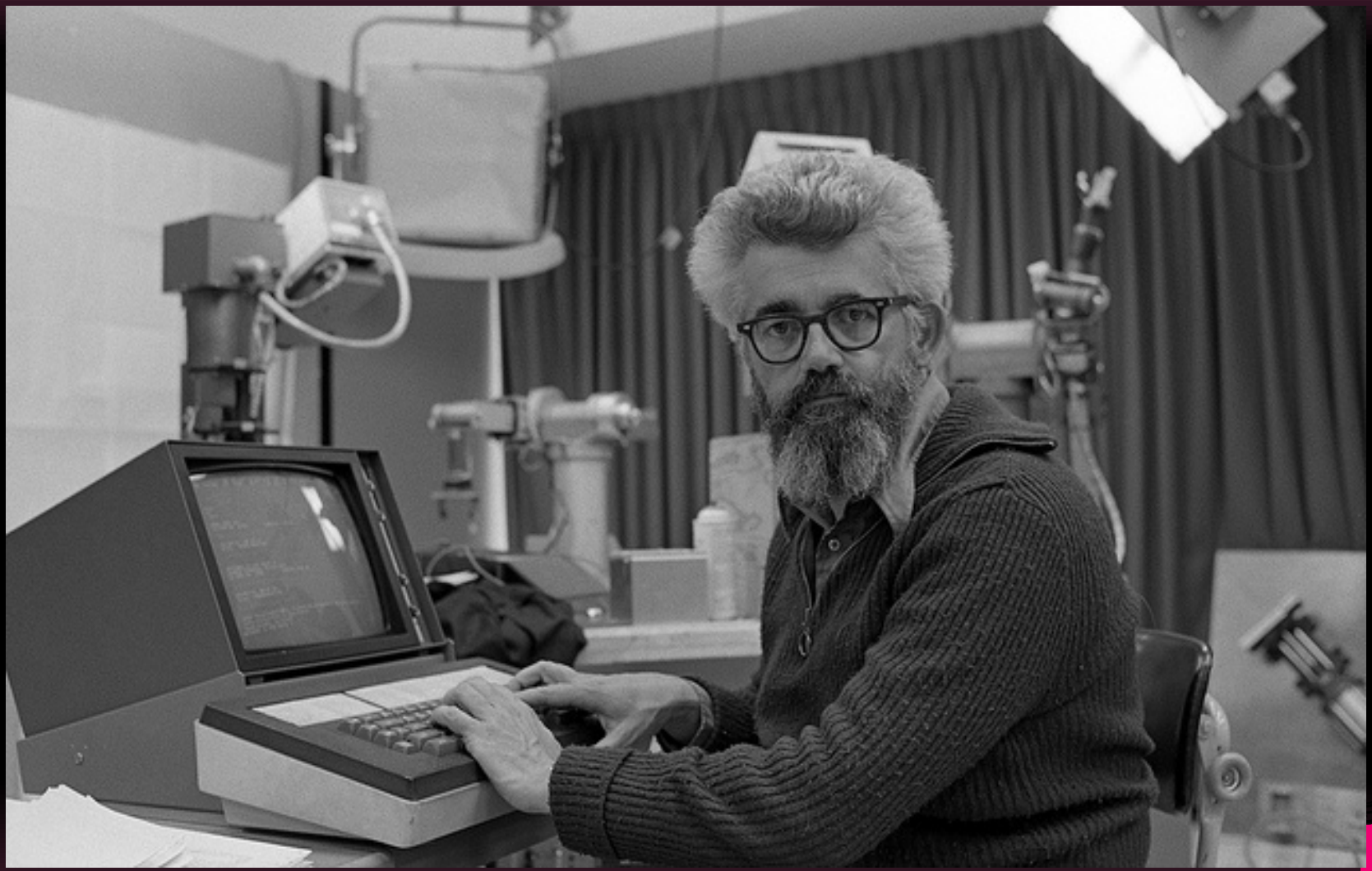
- Des diapos en amphi : on ne les recopie pas mais on les commente par écrit. Elles sont disponibles sur `HTTP://INFORMATHIX.TUXFAMILY.ORG`. Pas de poly de cours.
- En TD/TP : un petit poly avec des questions. Travail d'équipe avec l'intervenant pour y répondre.
- Parfois il faudra répondre à un petit QCM en ligne pour obtenir l'énoncé du TP.
- Des petits tests en amphi et en TD, annoncés à l'avance ou non.
- Des travaux personnels à rendre par écrit ou sous forme électronique.
- Des devoirs en amphi reprenant ce qui a été étudié. Vous avez droit à vos notes **manuscrites** : résumé, synthèse, recherches,...
- Puisqu'il en faut une, il y aura une note finale qui tentera de représenter vos travaux écrits et votre investissement en amphi et en TD/TP.

- Des diapos en amphi : on ne les recopie pas mais on les commente par écrit. Elles sont disponibles sur `HTTP://INFORMATHIX.TUXFAMILY.ORG`. Pas de poly de cours.
- En TD/TP : un petit poly avec des questions. Travail d'équipe avec l'intervenant pour y répondre.
- Parfois il faudra répondre à un petit QCM en ligne pour obtenir l'énoncé du TP.
- Des petits tests en amphi et en TD, annoncés à l'avance ou non.
- Des travaux personnels à rendre par écrit ou sous forme électronique.
- Des devoirs en amphi reprenant ce qui a été étudié. Vous avez droit à vos notes **manuscrites** : résumé, synthèse, recherches,...
- Puisqu'il en faut une, il y aura une note finale qui tentera de représenter vos travaux écrits et votre investissement en amphi et en TD/TP.

- Des diapos en amphi : on ne les recopie pas mais on les commente par écrit. Elles sont disponibles sur `HTTP://INFORMATHIX.TUXFAMILY.ORG`. Pas de poly de cours.
- En TD/TP : un petit poly avec des questions. Travail d'équipe avec l'intervenant pour y répondre.
- Parfois il faudra répondre à un petit QCM en ligne pour obtenir l'énoncé du TP.
- Des petits tests en amphi et en TD, annoncés à l'avance ou non.
- Des travaux personnels à rendre par écrit ou sous forme électronique.
- Des devoirs en amphi reprenant ce qui a été étudié. Vous avez droit à vos notes **manuscrites** : résumé, synthèse, recherches,...
- Puisqu'il en faut une, il y aura une note finale qui tentera de représenter vos travaux écrits et votre investissement en amphi et en TD/TP.

Sommaire

Julien Cohen :



Sébastien Faucou

Olivier Jousselin

Sommaire

- Les rudiments de la programmation avec Python
- Des mathématiques que vous n'avez jamais explorées avant :
  - Des rudiments sur la notion d'ensemble ;
  - Des rudiments sur la notion de relation ;

Ces notions vont être abordées au fur et à mesure de notre exploration de la programmation mais pas forcément dans l'ordre et de manière dispersée.
UN IMPORTANT TRAVAIL DE SYNTHÈSE DES INFORMATIONS SERA INDISPENSABLE !
C'est une exigence quotidienne du travail en entreprise ou à l'université.
On vous demande donc de changer votre approche du travail :
SCOLAIRE $\longrightarrow$ PROFESSIONNEL

- Les rudiments de la programmation avec Python
- Des mathématiques que vous n'avez jamais explorées avant :
  - Des rudiments sur la notion d'ensemble
  - Des rudiments sur la notion de relation
  - Des rudiments sur la notion de structure algébrique
  - Des rudiments d'arithmétique des entiers
  - Des rudiments de logique

Ces notions vont être abordées au fur et à mesure de notre exploration de la programmation mais pas forcément dans l'ordre et de manière dispersée.
UN IMPORTANT TRAVAIL DE SYNTHÈSE DES INFORMATIONS SERA INDISPENSABLE !
C'est une exigence quotidienne du travail en entreprise ou à l'université.
On vous demande donc de changer votre approche du travail :
SCOLAIRE $\longrightarrow$ PROFESSIONNEL

- Les rudiments de la programmation avec Python
- Des mathématiques que vous n'avez jamais explorées avant :
  - Des rudiments sur la notion d'ensemble
  - Des rudiments sur la notion de relation
  - Des rudiments sur la notion de structure algébrique
  - Des rudiments d'arithmétique des entiers
  - Des rudiments de logique

Ces notions vont être abordées au fur et à mesure de notre exploration de la programmation mais pas forcément dans l'ordre et de manière dispersée.

UN IMPORTANT TRAVAIL DE SYNTHÈSE DES INFORMATIONS SERA INDISPENSABLE !

C'est une exigence quotidienne du travail en entreprise ou à l'université.

On vous demande donc de changer votre approche du travail :

SCOLAIRE $\longrightarrow$ PROFESSIONNEL

- Les rudiments de la programmation avec Python
- Des mathématiques que vous n'avez jamais explorées avant :
  - Des rudiments sur la notion d'ensemble
  - Des rudiments sur la notion de relation
  - Des rudiments sur la notion de structure algébrique
  - Des rudiments d'arithmétique des entiers
  - Des rudiments de logique

Ces notions vont être abordées au fur et à mesure de notre exploration de la programmation mais pas forcément dans l'ordre et de manière dispersée.
UN IMPORTANT TRAVAIL DE SYNTHÈSE DES INFORMATIONS SERA INDISPENSABLE !
C'est une exigence quotidienne du travail en entreprise ou à l'université.
On vous demande donc de changer votre approche du travail :
SCOLAIRE $\longrightarrow$ PROFESSIONNEL

- Les rudiments de la programmation avec Python
- Des mathématiques que vous n'avez jamais explorées avant :
  - Des rudiments sur la notion d'ensemble
  - Des rudiments sur la notion de relation
  - Des rudiments sur la notion de structure algébrique
  - Des rudiments d'arithmétique des entiers
  - Des rudiments de logique

Ces notions vont être abordées au fur et à mesure de notre exploration de la programmation mais pas forcément dans l'ordre et de manière dispersée.
UN IMPORTANT TRAVAIL DE SYNTHÈSE DES INFORMATIONS SERA INDISPENSABLE !
C'est une exigence quotidienne du travail en entreprise ou à l'université.
On vous demande donc de changer votre approche du travail :
SCOLAIRE $\longrightarrow$ PROFESSIONNEL

- Les rudiments de la programmation avec Python
- Des mathématiques que vous n'avez jamais explorées avant :
  - Des rudiments sur la notion d'ensemble
  - Des rudiments sur la notion de relation
  - Des rudiments sur la notion de structure algébrique
  - Des rudiments d'arithmétique des entiers
  - Des rudiments de logique

Ces notions vont être abordées au fur et à mesure de notre exploration de la programmation mais pas forcément dans l'ordre et de manière dispersée.
UN IMPORTANT TRAVAIL DE SYNTHÈSE DES INFORMATIONS SERA INDISPENSABLE !
C'est une exigence quotidienne du travail en entreprise ou à l'université.
On vous demande donc de changer votre approche du travail :
SCOLAIRE $\longrightarrow$ PROFESSIONNEL

- Les rudiments de la programmation avec Python
- Des mathématiques que vous n'avez jamais explorées avant :
  - Des rudiments sur la notion d'ensemble
  - Des rudiments sur la notion de relation
  - Des rudiments sur la notion de structure algébrique
  - Des rudiments d'arithmétique des entiers
  - Des rudiments de logique

Ces notions vont être abordées au fur et à mesure de notre exploration de la programmation mais pas forcément dans l'ordre et de manière dispersée.
UN IMPORTANT TRAVAIL DE SYNTHÈSE DES INFORMATIONS SERA INDISPENSABLE !
C'est une exigence quotidienne du travail en entreprise ou à l'université.
On vous demande donc de changer votre approche du travail :
SCOLAIRE $\longrightarrow$ PROFESSIONNEL

- Les rudiments de la programmation avec Python
- Des mathématiques que vous n'avez jamais explorées avant :
  - Des rudiments sur la notion d'ensemble
  - Des rudiments sur la notion de relation
  - Des rudiments sur la notion de structure algébrique
  - Des rudiments d'arithmétique des entiers
  - Des rudiments de logique

Ces notions vont être abordées au fur et à mesure de notre exploration de la programmation mais pas forcément dans l'ordre et de manière dispersée.
UN IMPORTANT TRAVAIL DE SYNTHÈSE DES INFORMATIONS SERA INDISPENSABLE !
C'est une exigence quotidienne du travail en entreprise ou à l'université.
On vous demande donc de changer votre approche du travail :
SCOLAIRE $\longrightarrow$ PROFESSIONNEL

- Les rudiments de la programmation avec Python
- Des mathématiques que vous n'avez jamais explorées avant :
  - Des rudiments sur la notion d'ensemble
  - Des rudiments sur la notion de relation
  - Des rudiments sur la notion de structure algébrique
  - Des rudiments d'arithmétique des entiers
  - Des rudiments de logique

Ces notions vont être abordées au fur et à mesure de notre exploration de la programmation mais pas forcément dans l'ordre et de manière dispersée.

UN IMPORTANT TRAVAIL DE SYNTHÈSE DES INFORMATIONS SERA INDISPENSABLE !

C'est une exigence quotidienne du travail en entreprise ou à l'université.

On vous demande donc de changer votre approche du travail :

SCOLAIRE $\longrightarrow$ PROFESSIONNEL

- Les rudiments de la programmation avec Python
- Des mathématiques que vous n'avez jamais explorées avant :
  - Des rudiments sur la notion d'ensemble
  - Des rudiments sur la notion de relation
  - Des rudiments sur la notion de structure algébrique
  - Des rudiments d'arithmétique des entiers
  - Des rudiments de logique

Ces notions vont être abordées au fur et à mesure de notre exploration de la programmation mais pas forcément dans l'ordre et de manière dispersée.
UN IMPORTANT TRAVAIL DE SYNTHÈSE DES INFORMATIONS SERA INDISPENSABLE !
C'est une exigence quotidienne du travail en entreprise ou à l'université.
On vous demande donc de changer votre approche du travail :
SCOLAIRE $\longrightarrow$ PROFESSIONNEL

- Les rudiments de la programmation avec Python
- Des mathématiques que vous n'avez jamais explorées avant :
  - Des rudiments sur la notion d'ensemble
  - Des rudiments sur la notion de relation
  - Des rudiments sur la notion de structure algébrique
  - Des rudiments d'arithmétique des entiers
  - Des rudiments de logique

Ces notions vont être abordées au fur et à mesure de notre exploration de la programmation mais pas forcément dans l'ordre et de manière dispersée.
UN IMPORTANT TRAVAIL DE SYNTHÈSE DES INFORMATIONS SERA INDISPENSABLE !
C'est une exigence quotidienne du travail en entreprise ou à l'université.
On vous demande donc de changer votre approche du travail :
SCOLAIRE $\longrightarrow$ PROFESSIONNEL

- Les rudiments de la programmation avec Python
- Des mathématiques que vous n'avez jamais explorées avant :
  - Des rudiments sur la notion d'ensemble
  - Des rudiments sur la notion de relation
  - Des rudiments sur la notion de structure algébrique
  - Des rudiments d'arithmétique des entiers
  - Des rudiments de logique

Ces notions vont être abordées au fur et à mesure de notre exploration de la programmation mais pas forcément dans l'ordre et de manière dispersée.
UN IMPORTANT TRAVAIL DE SYNTHÈSE DES INFORMATIONS SERA INDISPENSABLE !
C'est une exigence quotidienne du travail en entreprise ou à l'université.
On vous demande donc de changer votre approche du travail :
SCOLAIRE $\longrightarrow$ PROFESSIONNEL

- Les rudiments de la programmation avec Python
- Des mathématiques que vous n'avez jamais explorées avant :
  - Des rudiments sur la notion d'ensemble
  - Des rudiments sur la notion de relation
  - Des rudiments sur la notion de structure algébrique
  - Des rudiments d'arithmétique des entiers
  - Des rudiments de logique

Ces notions vont être abordées au fur et à mesure de notre exploration de la programmation mais pas forcément dans l'ordre et de manière dispersée.
UN IMPORTANT TRAVAIL DE SYNTHÈSE DES INFORMATIONS SERA INDISPENSABLE !
C'est une exigence quotidienne du travail en entreprise ou à l'université.
On vous demande donc de changer votre approche du travail :
SCOLAIRE $\longrightarrow$ PROFESSIONNEL

Avant

Pendant

Après

À la fin de ce module, il faudra donc être capable de mobiliser vos connaissances pour résoudre de petits problèmes informatiques.

Ce travail se prolongera lors du module suivant qui explorera un peu plus profondément les structures algébriques afin d'aborder des rudiments sur la spécification et la vérification de programmes, l'application de ce qui a été vu aux bases de données, des rudiments de calcul matriciel pour aborder des structures de données plus complexes et la manipulation d'images par exemple.

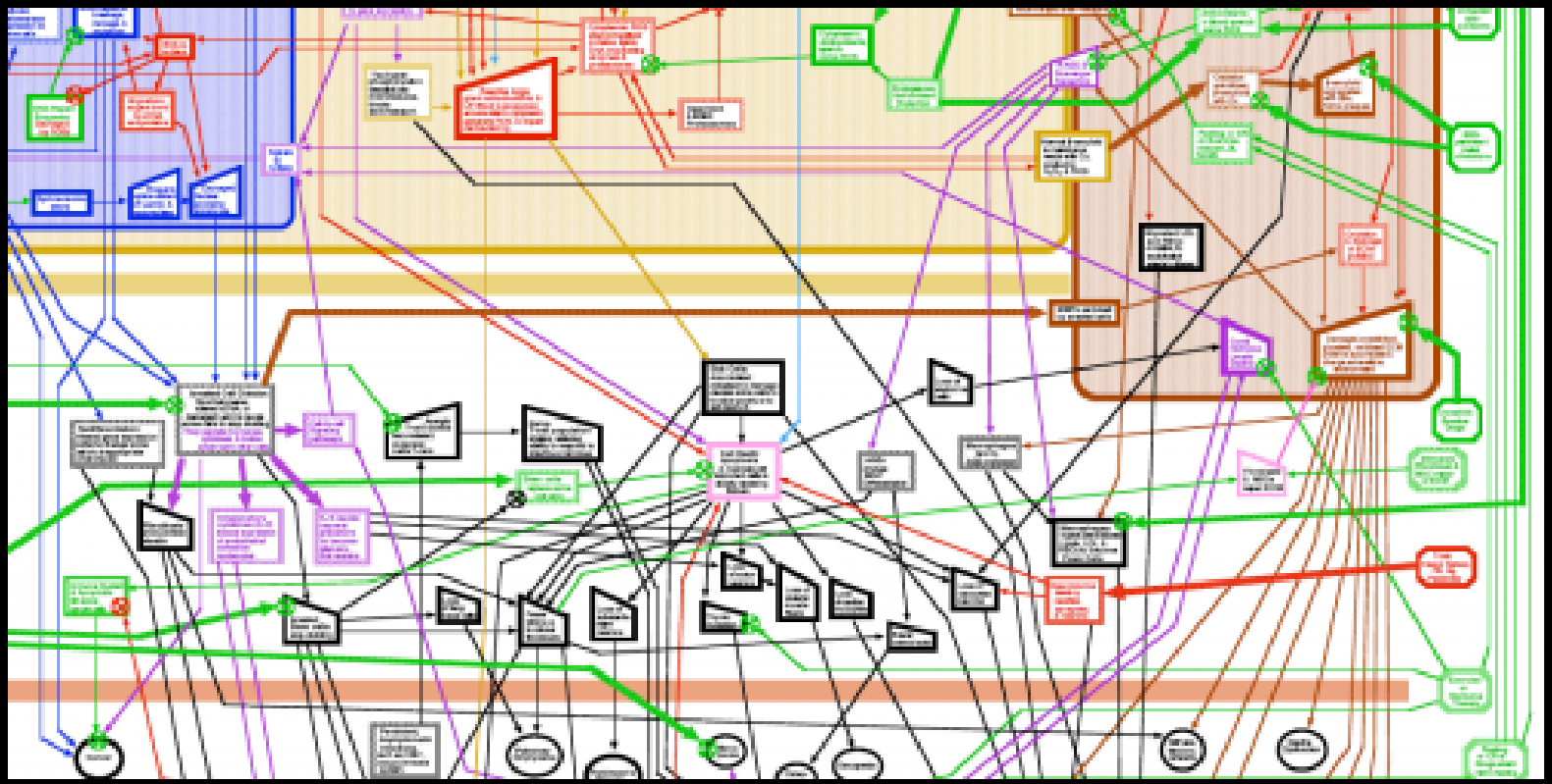
À la fin de ce module, il faudra donc être capable de mobiliser vos connaissances pour résoudre de petits problèmes informatiques.
Ce travail se prolongera lors du module suivant qui explorera un peu plus profondément les structures algébriques afin d'aborder des rudiments sur la spécification et la vérification de programmes, l'application de ce qui a été vu aux bases de données, des rudiments de calcul matriciel pour aborder des structures de données plus complexes et la manipulation d'images par exemple.

Sommaire

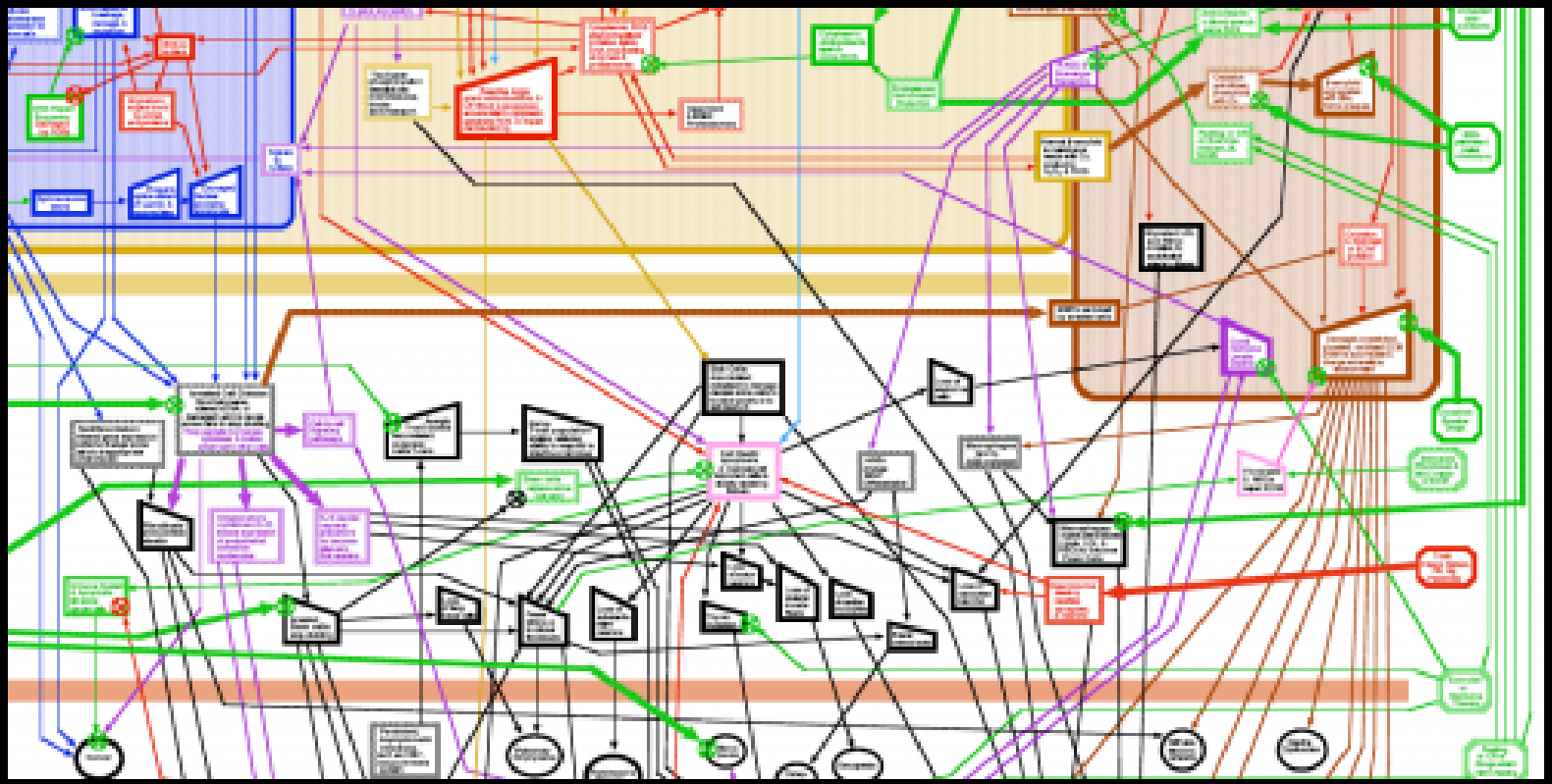
Sommaire

On veut faire la liste des URL qui apparaissent dans deux pages web.

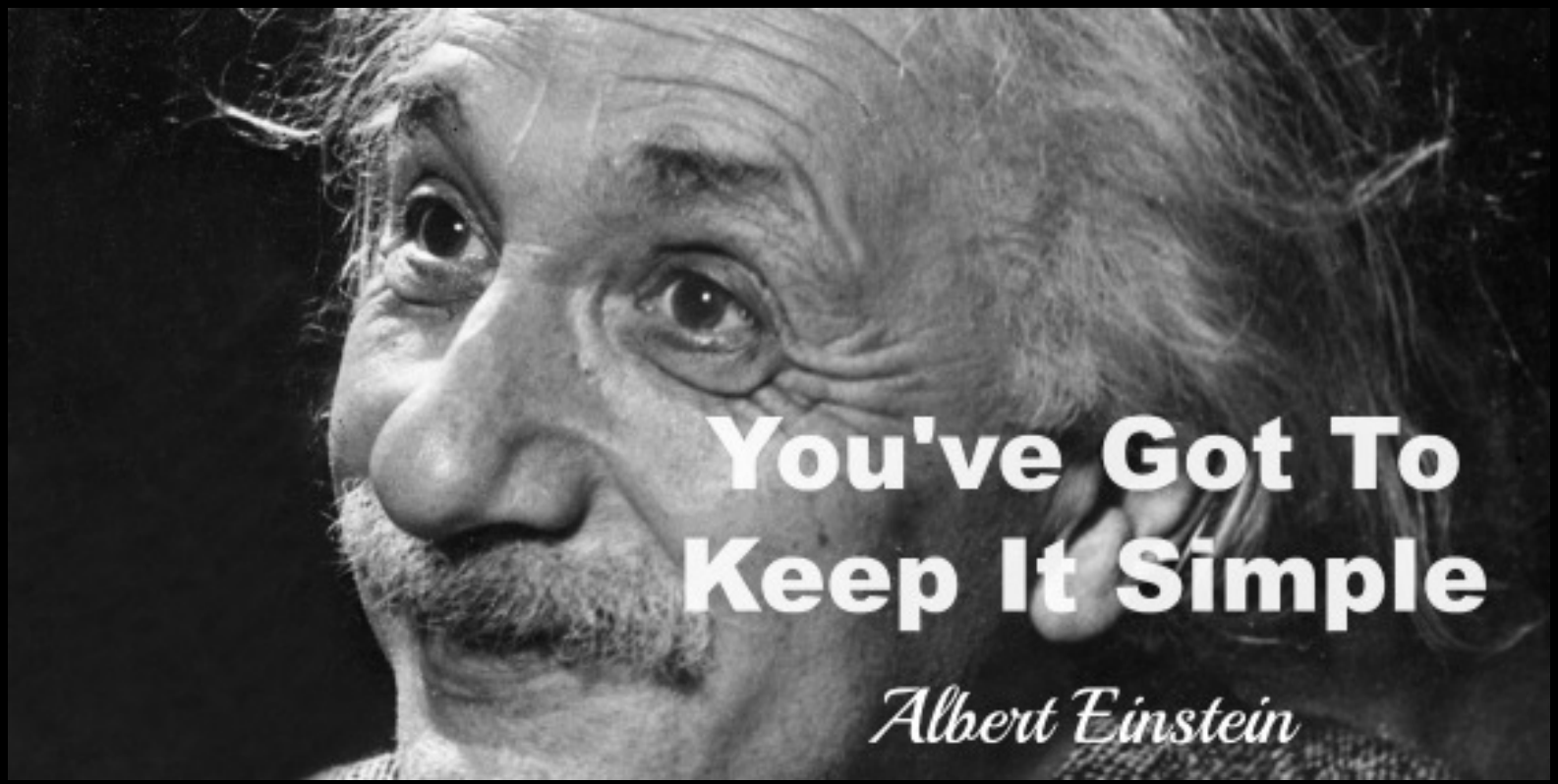
Des propositions ?



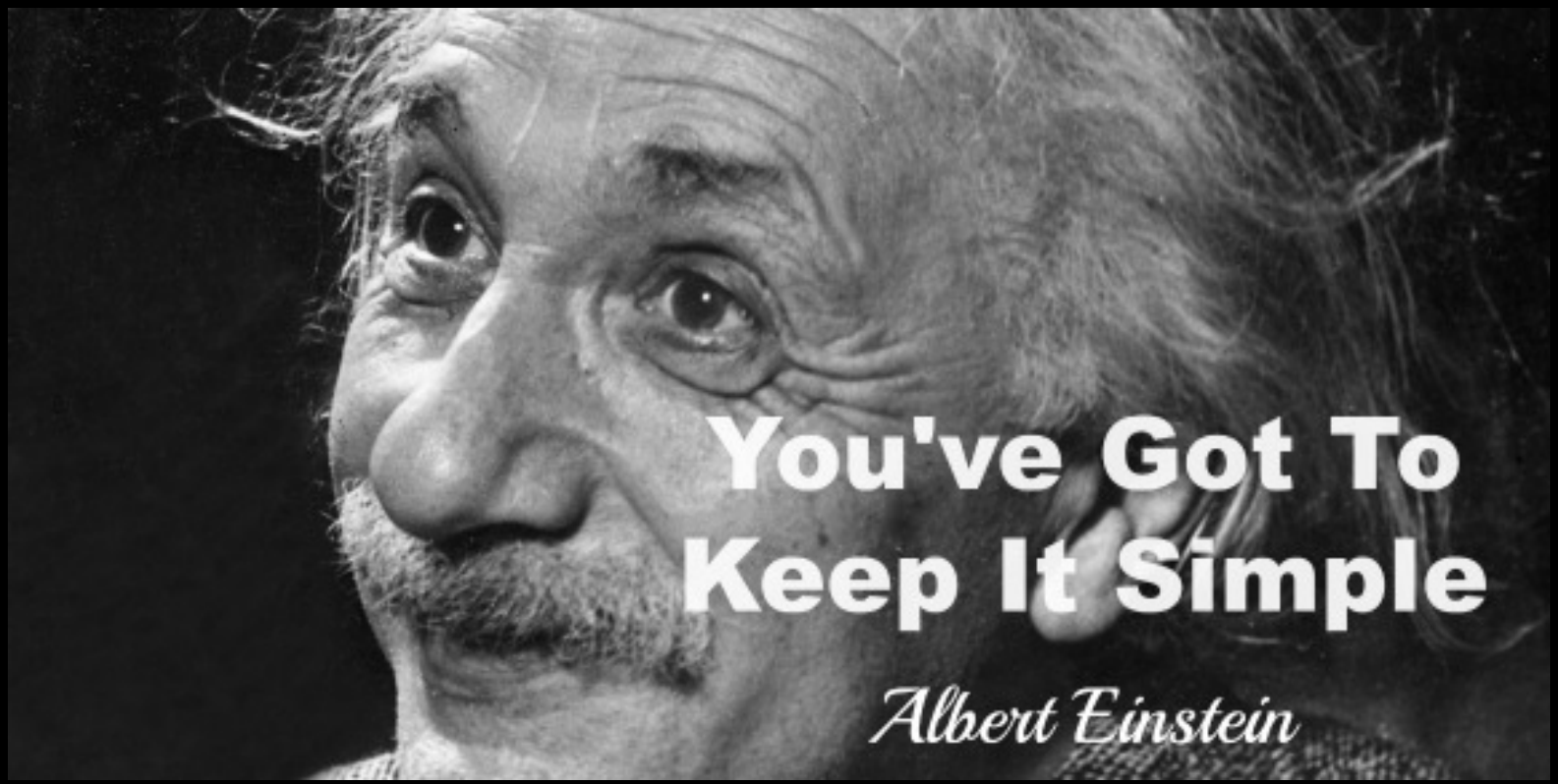
Des propositions ?



Et si on parlait maths ?



Et si on parlait maths ?



Python

- Bas niveau
- Haut niveau

- Bas niveau
- Haut niveau

Premier exemple

Recherche

On veut rechercher dans un texte les mots de quatre lettres qui apparaissent à la fois dans les deux sens de lecture et éventuellement compter leurs occurrences.

MARCEL PROUST

A LA RECHERCHE DU TEMPS PERDU

TOME I

DU CÔTÉ DE CHEZ SWANN
A L'OMBRE DES JEUNES FILLES EN FLEURS

TEXTE EN PARTIE INÉDIT ÉTABLI
SUR LES MANUSCRITS AUTOGRAPHES
VARIANTES, NOTES CRITIQUES, INTRODUCTION,
RÉSUMÉ DE CHAQUE PARTIE DE L'ŒUVRE
INDEX DES NOMS DE PERSONNAGES
ET DES NOMS DE LIEUX
CHRONOLOGIE DE MARCEL PROUST

par

PIERRE CLARAC et ANDRÉ FERRÉ

Préface d'ANDRÉ MAUROIS, de l'Académie française

BIBLIOTHÈQUE nrf DE LA PLÉIADE


CBS 3832
DU COTE DE CHEZ SWANN
Dave
FAIS-MOI L'AMOUR
DAVE


Le Projet Gutenberg offre 49 697 ebooks libres à télécharger.
Recherche
Derniers
Conditions d'utilisation
Faire un don?
Mobile
Recherche Projet Gutenber
Projet Gutenberg › 49 697 ebook libres › 10 par Marcel Proust
Du côté de chez Swann by Marcel Proust
Aucune image disponible
Télécharger
Bibrec
Télécharger cet ebook
Format
Taille
Read this book online: HTML
1.0 MB
EPUB (avec images)
405 kB
EPUB (sans images)
405 kB
Kindle (avec images)
1.6 MB
Kindle (sans images)
1.6 MB
Texte brut UTF-8
1.0 MB

```
swann = urlopen( 'https://www.gutenberg.org/files/2650/2650-0.txt' )

mots = set( swann.read().decode().split() )

verlan = { mot for mot in mots if len(mot) == 4 and mot[::-1] in mots }

print( 'Il y a ' + str(len(verlan)) + ' mots qui ont la propriété demandée : ' )

for mot in verlan :
    print(mot)
```

```
In [7]: Il y a 10 mots qui ont la propriété :
bons
snob
trot
tort
seul
lues
trop
port
elle
alla
```

Deuxième exemple

Quel est le rôle de nomTel ?

```
assoNantesBrut = urlopen(
'http://data.nantes.fr/api/publication/24440040400129_VDN_VDN_00132/ANNUAIRE_ASSOCIA
        ↪   )

assoNantes = assoNantesBrut.read().decode().split('\n')

def nomTel(mot) :
    annuaire = set()
    for ligne in assoNantes[1:-2] :
        donnees  = ligne.split(',')
        nom, tel = donnees[1], donnees[9]
        if mot in nom :
            annuaire.add( (nom,tel) )
    return annuaire
```

Pour vous aider, voici la première ligne :

```
In [8]: assoNantes[0]
Out[8]: ['"ID","NOM","SIGLE","SIEGE_NUM","SIEGE_CPLT","SIEGE_VOIE",
"SIEGE_CP","SIEGE_COMMUNE","SIEGE_CODCOM","TEL","LIEN1","LIEN2",
"LIEN3","DATE_CONST","DATE_DERN_MAJ","SIRET","OBJET","LIB_THEME",
"LIB_ACTIVITE","LIEU_ACT_NUM","LIEU_ACT_VOIE","LIEU_ACT_CP",
"LIEU_ACT_COMMUNE","LIEU_ACT_QUARTIER","PUB_ENFANTS","PUB_JEUNES",
"PUB_ADULTES","PUB_FAMILLES","PUB_SENIORS","PUB_ASSOCIATIONS"']
```

et si on avait directement écrit ceci :

```
return {(ligne.split(',')[1], ligne.split(',')[9]) for ligne in assoNantes[1:-2]
        ↪   if mot in ligne.split(',')[1]}
```

```
In [54]: nomTel('Informatique')
Out[54]:
{('"Association Libre Informatique et Solidaire"', '"0951112145"'),
 ('"Association des Jeunes Chercheurs en Informatique de Nantes - LOGIN"',
  '"0251125842"')}
```

```
In [61]: nomTel('Vélo')
Out[61]:
{('"Association Place au Vélo - Cheviré Cyclable"', '"0240200400"'),
 ('"Nantes Doulon Vélo Sport"', '"0240761292"'),
 ('"Terrain Rugby du Vélodrome du Petit Breton"', '"0240464819"'),
 ('"Vélo Sport Nantais Rugby"', '"0683857995"'),
 ('"Vélo Tout Terrain Nantais"', '"0671540551"'),
 ('"Vélocampus"', '"0240162645"'),
 ('"Véloce Sport Couëronnais"', '"0240383975"')}
```

```
In [75]: nomTel('Sourd')
Out[75]:
{('"Association Régionale des Pays de la Loire pour l\'Intégration de l\'Enfant
        ↪  Sourd"',
  '"0251838384"'),
 ('"Association des Sourds de la Loire Atlantique"', '""'),
 ('"Centre Socio-Culturel des Sourds de Loire-Atlantique"', '"0240351412"'),
 ('"Club Sportif des Sourds de Nantes"', '"0240754370"'),
 ('"Seniors Sourds de la Loire Atlantique"', '""')}
```

Sommaire


SALT-N-PEPA
LET'S TALK
ABOUT SETS
LONDON

De manière très sommaire (nous ferons mieux plus tard) :

- Un ensemble (`set`) est une collection non ordonnée d'éléments différents...plus quelques particularités informatiques en Python dues à la manière d'implémenter les ensembles mais nous ne l'évoquerons pas pour l'instant. Dans certains langages (mais pas Python), les éléments doivent être de même type.
- Une suite (en anglais de spécialité « *sequence* ») est une collection ordonnée d'élements non nécessairement différents. En informatique il y a de nombreuses structures qui appartiennent à cette catégorie.

De manière très sommaire (nous ferons mieux plus tard) :

- Un ensemble (`set`) est une collection non ordonnée d'éléments différents...plus quelques particularités informatiques en Python dues à la manière d'implémenter les ensembles mais nous ne l'évoquerons pas pour l'instant. Dans certains langages (mais pas Python), les éléments doivent être de même type.
- Une suite (en anglais de spécialité « *sequence* ») est une collection ordonnée d'élements non nécessairement différents. En informatique il y a de nombreuses structures qui appartiennent à cette catégorie.

De manière très sommaire (nous ferons mieux plus tard) :

- Un ensemble (`set`) est une collection non ordonnée d'éléments différents...plus quelques particularités informatiques en Python dues à la manière d'implémenter les ensembles mais nous ne l'évoquerons pas pour l'instant. Dans certains langages (mais pas Python), les éléments doivent être de même type.
- Une suite (en anglais de spécialité « *sequence* ») est une collection ordonnée d'élements non nécessairement différents. En informatique il y a de nombreuses structures qui appartiennent à cette catégorie.

De manière très sommaire (nous ferons mieux plus tard) :

- Un ensemble (`set`) est une collection non ordonnée d'éléments différents...plus quelques particularités informatiques en Python dues à la manière d'implémenter les ensembles mais nous ne l'évoquerons pas pour l'instant. Dans certains langages (mais pas Python), les éléments doivent être de même type.
- Une suite (en anglais de spécialité « *sequence* ») est une collection ordonnée d'élements non nécessairement différents. En informatique il y a de nombreuses structures qui appartiennent à cette catégorie.

Un coup d'œil :

```
In [113]: seq1 = tuple((1,2,3))

In [114]: seq2 = tuple((2,1,3))

In [115]: seq1 == seq2
Out[115]: False
```

```
In [116]: set1 = set((1,2,3))

In [117]: set2 = set((2,1,3))

In [118]: set1 == set2
Out[118]: True
```

```
In [119]: set1 == set((1,3,2,1,2,3,3,3,2,1,1,1,1))
Out[119]: True
```

Un coup d'œil :

```
In [113]: seq1 = tuple((1,2,3))

In [114]: seq2 = tuple((2,1,3))

In [115]: seq1 == seq2
Out[115]: False
```

```
In [116]: set1 = set((1,2,3))

In [117]: set2 = set((2,1,3))

In [118]: set1 == set2
Out[118]: True
```

```
In [119]: set1 == set((1,3,2,1,2,3,3,3,2,1,1,1,1))
Out[119]: True
```

Un coup d'œil :

```
In [113]: seq1 = tuple((1,2,3))

In [114]: seq2 = tuple((2,1,3))

In [115]: seq1 == seq2
Out[115]: False
```

```
In [116]: set1 = set((1,2,3))

In [117]: set2 = set((2,1,3))

In [118]: set1 == set2
Out[118]: True
```

```
In [119]: set1 == set((1,3,2,1,2,3,3,3,2,1,1,1,1))
Out[119]: True
```

```
swann = urlopen( 'https://www.gutenberg.org/files/2650/2650-0.txt' )
```

Pourquoi :

```
mots = set( swann.read().decode().split() )
```

et pas :

```
mots = swann.read().decode().split()
```

Ensemble ou suite ?
Conséquences en programmation ?

```
swann = urlopen( 'https://www.gutenberg.org/files/2650/2650-0.txt' )
```

Pourquoi :

```
mots = set( swann.read().decode().split() )
```

et pas :

```
mots = swann.read().decode().split()
```

Ensemble ou suite ?

Conséquences en programmation ?

```
swann = urlopen( 'https://www.gutenberg.org/files/2650/2650-0.txt' )
```

Pourquoi :

```
mots = set( swann.read().decode().split() )
```

et pas :

```
mots = swann.read().decode().split()
```

Ensemble ou suite ?
Conséquences en programmation ?

Quels sont vos commentaires en terme de suite/ensemble sur la fonction suivante :

```
def nomTel(mot) :
    annuaire = set()
    for ligne in assoNantes[1:-2] :
        donnees  = ligne.split(',')
        nom, tel = donnees[1], donnees[9]
        if mot in nom :
            annuaire.add( (nom,tel) )
    return annuaire
```

Comment décrire annuaire ? Ses éléments ?

- À chaque nom on associe un numéro de téléphone
- Chaque numéro de téléphone est associé à un nom

Mais certaines associations n'ont pas de numéro de téléphone (`('"Association des Sourds de la Loire Atlantique"', '""')`) et on peut imaginer que dans un autre contexte, une personne puisse avoir plusieurs numéros.
Quelles conséquences informatiques ?
Quelle notion mathématique permet de clarifier tout ça ?

- À chaque nom on associe un numéro de téléphone
- Chaque numéro de téléphone est associé à un nom

Mais certaines associations n'ont pas de numéro de téléphone (`('"Association des Sourds de la Loire Atlantique"', '""')`) et on peut imaginer que dans un autre contexte, une personne puisse avoir plusieurs numéros.
Quelles conséquences informatiques ?
Quelle notion mathématique permet de clarifier tout ça ?

- À chaque nom on associe un numéro de téléphone
- Chaque numéro de téléphone est associé à un nom

Mais certaines associations n'ont pas de numéro de téléphone (`('"Association des Sourds de la Loire Atlantique"', '""')`) et on peut imaginer que dans un autre contexte, une personne puisse avoir plusieurs numéros.
Quelles conséquences informatiques ?
Quelle notion mathématique permet de clarifier tout ça ?

- À chaque nom on associe un numéro de téléphone
- Chaque numéro de téléphone est associé à un nom

Mais certaines associations n'ont pas de numéro de téléphone (`('"Association des Sourds de la Loire Atlantique"', '""')`) et on peut imaginer que dans un autre contexte, une personne puisse avoir plusieurs numéros.
Quelles conséquences informatiques ?
Quelle notion mathématique permet de clarifier tout ça ?

- relation
- fonction
- application
- surjection
- injection
- bijection

- relation
- fonction
- application
- surjection
- injection
- bijection

- relation
- fonction
- application
- surjection
- injection
- bijection

- relation
- fonction
- application
- surjection
- injection
- bijection

- relation
- fonction
- application
- surjection
- injection
- bijection

- relation
- fonction
- application
- surjection
- injection
- bijection

Troisième exemple

```
swann = urlopen( 'https://www.gutenberg.org/files/2650/2650-0.txt' )

texte = swann.read().decode().split()

mots = set( texte )

verlan = { mot for mot in mots if len(mot) == 4 and mot[::-1] in mots }

def nb_occu(mot) :
    cpt = 0
    for w in texte :
        cpt += w == mot
    return cpt

dico_occu = { mot:nb_occu(mot) for mot in verlan }
```

Sommaire

Sommaire

- Python 3.4
- Shell interactif IPython version 4
- Des produits clés en main comme PyZo

- Python 3.4
- Shell interactif IPython version 4
- Des produits clés en main comme PyZo

- Python 3.4
- Shell interactif IPython version 4
- Des produits clés en main comme PyZo

Sommaire

Pas trop de détails informatiques pour l'instant.

- valeur
- nom
- type

Pas trop de détails informatiques pour l'instant.

- valeur
- nom
- type

Pas trop de détails informatiques pour l'instant.

- valeur
- nom
- type

Type : « ensemble » de valeurs ...mais pas seulement.

C'est un ensemble de valeurs et des opérations associées et une certaine façon de les stocker en mémoire.

Cela va nous occuper un peu plus tard quand nous aurons un peu plus de maturité...

Présentons cependant quelques types basiques.

Type : « ensemble » de valeurs...mais pas seulement.

C'est un ensemble de valeurs et des opérations associées et une certaine façon de les stocker en mémoire.

Cela va nous occuper un peu plus tard quand nous aurons un peu plus de maturité...

Présentons cependant quelques types basiques.

Type : « ensemble » de valeurs...mais pas seulement.
C'est un ensemble de valeurs et des opérations associées et une certaine façon de les stocker en mémoire.
Cela va nous occuper un peu plus tard quand nous aurons un peu plus de maturité...
Présentons cependant quelques types basiques.

Type : « ensemble » de valeurs...mais pas seulement.
C'est un ensemble de valeurs et des opérations associées et une certaine façon de les stocker en mémoire.
Cela va nous occuper un peu plus tard quand nous aurons un peu plus de maturité...
Présentons cependant quelques types basiques.

Type : « ensemble » de valeurs...mais pas seulement.
C'est un ensemble de valeurs et des opérations associées et une certaine façon de les stocker en mémoire.
Cela va nous occuper un peu plus tard quand nous aurons un peu plus de maturité...
Présentons cependant quelques types basiques.

Sommaire

L'ensemble des booléens est assez limité puisqu'il n'a que 2 éléments : `True` et `False`.

```
In [16]: type(True)
Out[16]: bool

In [17]: type(False)
Out[17]: bool
```

L'ensemble des booléens est assez limité puisqu'il n'a que 2 éléments : `True` et `False`.

```
In [16]: type(True)
Out[16]: bool

In [17]: type(False)
Out[17]: bool
```

```
In [18]: type( (1,2) == (2,1) )
Out[18]: bool
```

- opérandes : `(1,2)` et `(2,1)`
- opérateur binaire : `==`
- expression : `(1,2) == (2,1)`
- évaluation : `False`

```
In [18]: type( (1,2) == (2,1) )
Out[18]: bool
```

- opérandes : `(1,2)` et `(2,1)`
- opérateur binaire : `==`
- expression : `(1,2) == (2,1)`
- évaluation : `False`

```
In [18]: type( (1,2) == (2,1) )
Out[18]: bool
```

- opérandes : (1,2) et (2,1)
- opérateur binaire : ==
- expression : (1,2) == (2,1)
- évaluation : False

```
In [18]: type( (1,2) == (2,1) )
Out[18]: bool
```

- opérandes : (1,2) et (2,1)
- opérateur binaire : ==
- expression : (1,2) == (2,1)
- évaluation : False

```
In [18]: type( (1,2) == (2,1) )
Out[18]: bool
```

- opérandes : (1,2) et (2,1)
- opérateur binaire : ==
- expression : (1,2) == (2,1)
- évaluation : False

```
In [18]: type( (1,2) == (2,1) )
Out[18]: bool
```

- opérandes : (1,2) et (2,1)
- opérateur binaire : ==
- expression : (1,2) == (2,1)
- évaluation : False

```
In [18]: type( (1,2) == (2,1) )
Out[18]: bool
```

- opérandes : (1,2) et (2,1)
- opérateur binaire : ==
- expression : (1,2) == (2,1)
- évaluation : False

```
In [18]: type( (1,2) == (2,1) )
Out[18]: bool
```

- opérandes : `(1,2)` et `(2,1)`
- opérateur binaire : `==`
- expression : `(1,2) == (2,1)`
- évaluation : `False`

```
In [18]: type( (1,2) == (2,1) )
Out[18]: bool
```

- opérandes : `(1,2)` et `(2,1)`
- opérateur binaire : `==`
- expression : `(1,2) == (2,1)`
- évaluation : `False`

```
In [19]: A = { 1, 2 }

In [20]: B = { 2, 1 }

In [21]: ( A <= B ) and ( B <= A )
Out[21]: True
```

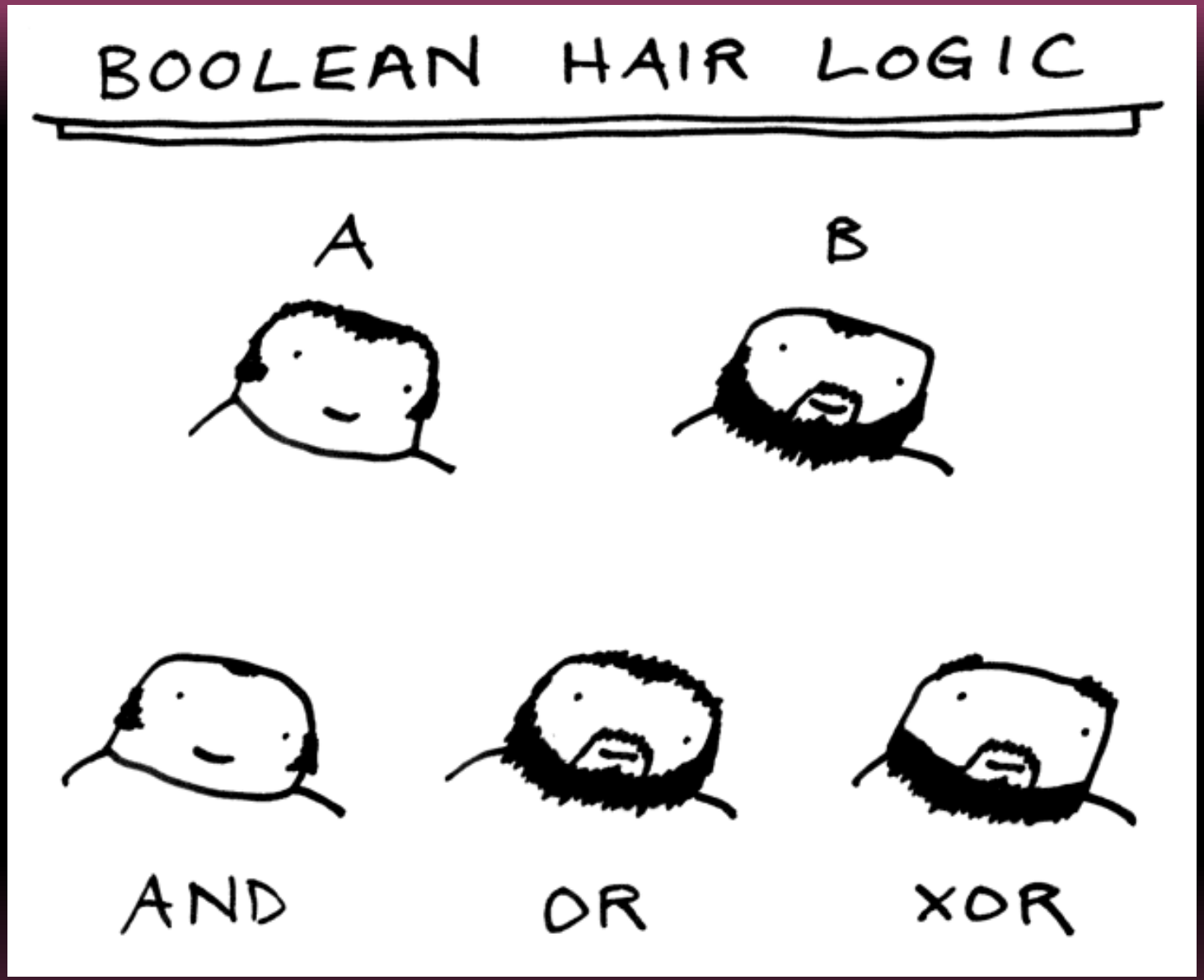
```
In [22]: ( ( A <= B ) and ( B <= A ) == ( 2 + 2 == 4 ) )
Out[22]: True
```

```
In [19]: A = { 1, 2 }

In [20]: B = { 2, 1 }

In [21]: ( A <= B ) and ( B <= A )
Out[21]: True
```

```
In [22]: ( ( A <= B ) and ( B <= A ) == ( 2 + 2 == 4 ) )
Out[22]: True
```


BOOLEAN HAIR LOGIC
A
B
AND
OR
XOR

Extrait de la Doc

These are the Boolean operations, ordered by ascending priority :

Operation	Result	Notes
x or y	if x is false, then y, else x	(1)
x and y	if x is false, then x, else y	(2)
not x	if x is false, then True, else False	(3)

Notes :
(1) This is a short-circuit operator, so it only evaluates the second argument if the first one is False.
(2) This is a short-circuit operator, so it only evaluates the second argument if the first one is True.
(3) not has a lower priority than non-Boolean operators, so not a == b is interpreted as not (a == b), and a == not b is a syntax error.

Notations

- and se note généralement $\wedge$
- or se note généralement $\vee$
- not se note généralement $\neg$

Notations

- `and` se note généralement $\wedge$
- `or` se note généralement $\vee$
- `not` se note généralement $\neg$

Notations

- `and` se note généralement $\wedge$
- `or` se note généralement $\vee$
- `not` se note généralement $\neg$

Une spécification des booléens

- **Sorte :** Boul
- Opérations :
  - Vrai : : Boul
  - Faux ∧ b = Faux

Une spécification des booléens

- **Sorte :** Boul
- **Opérations :**
  - Vrai : : Boul
  - Faux ∧ b = Faux

Une spécification des booléens

- **Sorte :** Boul
- **Opérations :**
  - Vrai : : Boul { constructeur }

- Faux ∧ b = Faux

Une spécification des booléens

- **Sorte :** Boul
- **Opérations :**
  - Vrai : : Boul { constructeur }
  - Faux : : Boul { constructeur }

- Faux ∧ b = Faux

Une spécification des booléens

- **Sorte :** Boul
- **Opérations :**
  - Vrai : : Boul { constructeur }
  - Faux : : Boul { constructeur }

- Faux $\wedge$ b = Faux

Une spécification des booléens

- **Sorte :** Boul
- **Opérations :**
  - Vrai : : Boul { constructeur }
  - Faux : : Boul { constructeur }

- Faux ∧ b = Faux

Une spécification des booléens

- **Sorte :** Boul
- **Opérations :**
  - Vrai : : Boul { constructeur }
  - Faux : : Boul { constructeur }
  - ... : Boul -> Boul

- Faux ∧ b = Faux

Une spécification des booléens

- **Sorte :** Boul
- **Opérations :**
  - Vrai : : Boul { constructeur }
  - Faux : : Boul { constructeur }
  - $\neg$: : Boul $\longrightarrow$ Boul
  - $\wedge$: : Boul Boul $\longrightarrow$ Boul
- **Axiomes :**
  - Faux $\wedge$ b = Faux

Une spécification des booléens

- **Sorte :** Boul
- **Opérations :**
  - Vrai : : Boul { constructeur }
  - Faux : : Boul { constructeur }
  - $\neg$: : Boul $\longrightarrow$ Boul
  - $\wedge$: : Boul Boul $\longrightarrow$ Boul
- **Axiomes**
  - Faux $\wedge$ b = Faux

Une spécification des booléens

- **Sorte :** Boul
- **Opérations :**
  - Vrai : : Boul { constructeur }
  - Faux : : Boul { constructeur }
  - $\neg$: : Boul $\longrightarrow$ Boul
  - $\wedge$: : Boul Boul $\longrightarrow$ Boul
- **Axiomes**
  - Variable : b : : Boul
  - Equations :
    - Faux $\wedge$ b = Faux

Une spécification des booléens

- **Sorte :** Boul
- **Opérations :**
  - Vrai : : Boul { constructeur }
  - Faux : : Boul { constructeur }
  - $\neg$: : Boul $\longrightarrow$ Boul
  - $\wedge$: : Boul Boul $\longrightarrow$ Boul
- **Axiomes**
  - **Variable :** b : : Boul
  - **Équations :**
    - Faux $\wedge$ b = Faux

Une spécification des booléens

- **Sorte :** Boul
- **Opérations :**
  - Vrai : : Boul { constructeur }
  - Faux : : Boul { constructeur }
  - $\neg$: : Boul $\longrightarrow$ Boul
  - $\wedge$: : Boul Boul $\longrightarrow$ Boul
- **Axiomes**
  - **Variable :** b : : Boul
  - **Équations :**
    - $\neg$ Vrai = Faux
    - $\neg$ Faux = Vrai
    - Vrai $\wedge$ b = b
    - Faux $\wedge$ b = Faux

Une spécification des booléens

- **Sorte :** Boul
- **Opérations :**
  - Vrai : : Boul { constructeur }
  - Faux : : Boul { constructeur }
  - $\neg$: : Boul $\longrightarrow$ Boul
  - $\wedge$: : Boul Boul $\longrightarrow$ Boul
- **Axiomes**
  - **Variable :** b : : Boul
  - **Équations :**
    - $\neg$ Vrai = Faux
    - $\neg$ Faux = Vrai
    - Vrai $\wedge$ b = b
    - Faux $\wedge$ b = Faux

Une spécification des booléens

- **Sorte :** Boul
- **Opérations :**
  - Vrai : : Boul { constructeur }
  - Faux : : Boul { constructeur }
  - $\neg$: : Boul $\longrightarrow$ Boul
  - $\wedge$: : Boul Boul $\longrightarrow$ Boul
- **Axiomes**
  - **Variable :** b : : Boul
  - **Équations :**
    - $\neg$ Vrai = Faux
    - $\neg$ Faux = Vrai
    - Vrai $\wedge$ b = b
    - Faux $\wedge$ b = Faux

Une spécification des booléens

- **Sorte :** Boul
- **Opérations :**
  - Vrai : : Boul { constructeur }
  - Faux : : Boul { constructeur }
  - $\neg$: : Boul $\longrightarrow$ Boul
  - $\wedge$: : Boul Boul $\longrightarrow$ Boul
- **Axiomes**
  - **Variable :** b : : Boul
  - **Équations :**
    - $\neg$ Vrai = Faux
    - $\neg$ Faux = Vrai
    - Vrai $\wedge$ b = b
    - Faux $\wedge$ b = Faux

Une spécification des booléens

- **Sorte :** Boul
- **Opérations :**
  - Vrai : : Boul { constructeur }
  - Faux : : Boul { constructeur }
  - $\neg$: : Boul $\longrightarrow$ Boul
  - $\wedge$: : Boul Boul $\longrightarrow$ Boul
- **Axiomes**
  - **Variable :** b : : Boul
  - **Équations :**
    - $\neg$ Vrai = Faux
    - $\neg$ Faux = Vrai
    - Vrai $\wedge$ b = b
    - Faux $\wedge$ b = Faux

Une spécification des booléens

- **Sorte :** Boul
- **Opérations :**
  - Vrai : : Boul { constructeur }
  - Faux : : Boul { constructeur }
  - $\neg$: : Boul $\longrightarrow$ Boul
  - $\wedge$: : Boul Boul $\longrightarrow$ Boul
- **Axiomes**
  - **Variable :** b : : Boul
  - **Équations :**
    - $\neg$ Vrai = Faux
    - $\neg$ Faux = Vrai
    - Vrai $\wedge$ b = b
    - Faux $\wedge$ b = Faux

Et les autres opérations ? ($\vee$, $\Longrightarrow$, $\Longleftrightarrow$ et $\oplus$)

- $b \vee c = \neg((\neg b) \wedge (\neg c))$
- $b \Longrightarrow c = (\neg b) \vee c$
- $b \Longleftrightarrow c = (b \Longrightarrow c) \wedge (c \Longrightarrow b)$
- $b \oplus c = \neg(b \Longleftrightarrow c)$

Et les autres opérations ? ($\vee$, $\Longrightarrow$, $\Longleftrightarrow$ et $\oplus$)

- $b \vee c = \neg((\neg b) \wedge (\neg c))$
- $b \Longrightarrow c = (\neg b) \vee c$
- $b \Longleftrightarrow c = (b \Longrightarrow c) \wedge (c \Longrightarrow b)$
- $b \oplus c = \neg(b \Longleftrightarrow c)$

Et les autres opérations ? ($\vee$, $\Longrightarrow$, $\Longleftrightarrow$ et $\oplus$)

- $b \vee c = \neg((\neg b) \wedge (\neg c))$
- $b \Longrightarrow c = (\neg b) \vee c$
- $b \Longleftrightarrow c = (b \Longrightarrow c) \wedge (c \Longrightarrow b)$
- $b \oplus c = \neg(b \Longleftrightarrow c)$

Et les autres opérations ? ($\vee$, $\Longrightarrow$, $\Longleftrightarrow$ et $\oplus$)

- $b \vee c = \neg((\neg b) \wedge (\neg c))$
- $b \Longrightarrow c = (\neg b) \vee c$
- $b \Longleftrightarrow c = (b \Longrightarrow c) \wedge (c \Longrightarrow b)$
- $b \oplus c = \neg(b \Longleftrightarrow c)$

Et les autres opérations ? ($\vee$, $\Longrightarrow$, $\Longleftrightarrow$ et $\oplus$)

- $b \vee c = \neg((\neg b) \wedge (\neg c))$
- $b \Longrightarrow c = (\neg b) \vee c$
- $b \Longleftrightarrow c = (b \Longrightarrow c) \wedge (c \Longrightarrow b)$
- $b \oplus c = \neg(b \Longleftrightarrow c)$

```
In [23]: True == 1
Out[23]: True

In [24]: True == 2
Out[24]: False

In [25]: True == 0
Out[25]: False

In [26]: False == 0
Out[26]: True
```

```
In [27]: ( A <= B ) + ( B <= A ) + ( 2 + 2 == 4)
```

```
Out[27]: 3
```

```
In [23]: True == 1
Out[23]: True

In [24]: True == 2
Out[24]: False

In [25]: True == 0
Out[25]: False

In [26]: False == 0
Out[26]: True
```

```
In [27]: ( A <= B ) + ( B <= A ) + ( 2 + 2 == 4)
```

```
Out[27]: 3
```

```
In [23]: True == 1
Out[23]: True

In [24]: True == 2
Out[24]: False

In [25]: True == 0
Out[25]: False

In [26]: False == 0
Out[26]: True
```

```
In [27]: ( A <= B ) + ( B <= A ) + ( 2 + 2 == 4)
```

```
Out[27]: 3
```

PEP 285

PEP : Python Enhancement Proposals

Python's Booleans were added with the primary goal of making code clearer. For example, if you're reading a function and encounter the statement return 1, you might wonder whether the 1 represents a Boolean truth value, an index, or a coefficient that multiplies some other quantity. If the statement is return True, however, the meaning of the return value is quite clear.

Python's Booleans were not added for the sake of strict type-checking. A very strict language such as Pascal would also prevent you performing arithmetic with Booleans, and would require that the expression in an if statement always evaluate to a Boolean result. Python is not this strict and never will be, as PEP 285 explicitly says. This means you can still use any expression in an if statement, even ones that evaluate to a list or tuple or some random object. The Boolean type is a subclass of the int class so that arithmetic using a Boolean still works.

PEP 285

PEP : Python Enhancement Proposals

Python's Booleans were added with the primary goal of making code clearer. For example, if you're reading a function and encounter the statement return 1, you might wonder whether the 1 represents a Boolean truth value, an index, or a coefficient that multiplies some other quantity. If the statement is return True, however, the meaning of the return value is quite clear.

Python's Booleans were not added for the sake of strict type-checking. A very strict language such as Pascal would also prevent you performing arithmetic with Booleans, and would require that the expression in an if statement always evaluate to a Boolean result. Python is not this strict and never will be, as PEP 285 explicitly says. This means you can still use any expression in an if statement, even ones that evaluate to a list or tuple or some random object. The Boolean type is a subclass of the int class so that arithmetic using a Boolean still works.

PEP 285

PEP : Python Enhancement Proposals
Python's Booleans were added with the primary goal of making code clearer. For example, if you're reading a function and encounter the statement return 1, you might wonder whether the 1 represents a Boolean truth value, an index, or a coefficient that multiplies some other quantity. If the statement is return True, however, the meaning of the return value is quite clear.
Python's Booleans were not added for the sake of strict type-checking. A very strict language such as Pascal would also prevent you performing arithmetic with Booleans, and would require that the expression in an if statement always evaluate to a Boolean result. Python is not this strict and never will be, as PEP 285 explicitly says. This means you can still use any expression in an if statement, even ones that evaluate to a list or tuple or some random object. The Boolean type is a subclass of the int class so that arithmetic using a Boolean still works.

PEP 285

PEP : Python Enhancement Proposals
Python's Booleans were added with the primary goal of making code clearer. For example, if you're reading a function and encounter the statement return 1, you might wonder whether the 1 represents a Boolean truth value, an index, or a coefficient that multiplies some other quantity. If the statement is return True, however, the meaning of the return value is quite clear.
Python's Booleans were not added for the sake of strict type-checking. A very strict language such as Pascal would also prevent you performing arithmetic with Booleans, and would require that the expression in an if statement always evaluate to a Boolean result. Python is not this strict and never will be, as PEP 285 explicitly says. This means you can still use any expression in an if statement, even ones that evaluate to a list or tuple or some random object. The Boolean type is a subclass of the int class so that arithmetic using a Boolean still works.

```python
def nb_occu(mot) :
    cpt = 0
    for w in texte :
        cpt += w == mot
    return cpt
```

Sommaire

Vous avez appris au collège qu'il y avait des entiers positifs (naturels) qu'on regroupe dans un ensemble appelé $\mathbb{N}$ et un plus grand ensemble, $\mathbb{Z}$ qui regroupe tous les zentiers, positifs comme négatifs.
Tous les éléments de $\mathbb{N}$ sont des éléments de $\mathbb{Z}$: on dit que $\mathbb{N}$ *est inclus* dans $\mathbb{Z}$ ou que $\mathbb{Z}$ *contient* $\mathbb{N}$.

Définition 1 (inclusion)

L'ensemble X est inclus dans l'ensemble Y si, et seulement si, tous les éléments de X sont des éléments de Y. Cela se note

$$(X \subseteq Y) \iff ((z \in X) \Rightarrow (z \in Y))$$

On dira indifféremment « X *est contenu dans* Y », « Y *contient* X », « X *est un sous-ensemble de* Y », « X *est une partie de* Y ».

Danger

Il faudra bien distinguer $\subset$, $\subseteq$ et $\not\subseteq$.
$B \not\subseteq A$ signifie $(B \subseteq A$ et $B \neq A)$.
Il ne faut pas confondre avec $B \not\subseteq A$ qui exprime que B n'est pas inclus dans A.
On dit que B est une partie propre de A si, et seulement si, $B \subseteq A$ avec $B \neq A$ et $B \neq \emptyset$ (il est nécessaire que A ne soit pas vide et ne soit pas un singleton).

Danger

Il faudra bien distinguer $\subset$, $\subseteq$ et $\not\subseteq$.
$B \subsetneq A$ signifie $(B \subseteq A$ et $B \neq A)$.
Il ne faut pas confondre avec $B \not\subseteq A$ qui exprime que B n'est pas inclus dans A.
On dit que B est une partie propre de A si, et seulement si, $B \subseteq A$ avec $B \neq A$ et $B \neq \emptyset$ (il est nécessaire que A ne soit pas vide et ne soit pas un singleton).

Danger

Il faudra bien distinguer $\subset$, $\subseteq$ et $\nsubseteq$.
$B \subsetneq A$ signifie $(B \subseteq A$ et $B \neq A)$.
Il ne faut pas confondre avec $B \nsubseteq A$ qui exprime que B n'est pas inclus dans A.
On dit que B est une partie propre de A si, et seulement si, $B \subseteq A$ avec $B \neq A$ et $B \neq \emptyset$ (il est nécessaire que A ne soit pas vide et ne soit pas un singleton).

Danger

Il faudra bien distinguer $\subset$, $\subseteq$ et $\not\subseteq$.
$B \subsetneq A$ signifie $(B \subseteq A$ et $B \neq A)$.
Il ne faut pas confondre avec $B \not\subseteq A$ qui exprime que B n'est pas inclus dans A.
On dit que B est une partie propre de A si, et seulement si, $B \subseteq A$ avec $B \neq A$ et $B \neq \emptyset$ (il est nécessaire que A ne soit pas vide et ne soit pas un singleton).

Danger

Il faudra bien distinguer $\subset$, $\subseteq$ et $\nsubseteq$.
$B \subsetneq A$ signifie $(B \subseteq A$ et $B \neq A)$.
Il ne faut pas confondre avec $B \nsubseteq A$ qui exprime que B n'est pas inclus dans A.
On dit que B est une partie propre de A si, et seulement si, $B \subseteq A$ avec $B \neq A$ et $B \neq \varnothing$ (il est nécessaire que A ne soit pas vide et ne soit pas un singleton).

```
In [11]: A = { 1, 2, 3 }

In [12]: B = { 2, 3 }

In [13]: B <= A
Out[13]: True

In [14]: C = { 1, 4 }

In [15]: C <= A
Out[15]: False
```

Théorème 2 (Axiome d'extensionalité)

$$(\forall X)(\forall Y)(((X \subseteq Y) \wedge (Y \subseteq X)) \Longrightarrow (X == Y))$$

??????????????

Théorème 2 (Axiome d'extensionalité)

$$(\forall X)(\forall Y)(((X \subseteq Y) \wedge (Y \subseteq X)) \Longrightarrow (X == Y))$$

??????????????

Théorème 2 (Axiome d'extensionalité)

$$(\forall X)(\forall Y)(((X \subseteq Y) \wedge (Y \subseteq X)) \Longrightarrow (X == Y))$$

??????????????

Sommaire

« *est un entier pair* » est une proposition qui ne peut être que vraie ou fausse et qui est définie sur l'ensemble $\mathbb{N}$ des entiers naturels que nous noterons P. Alors 2 vérifie P mais 3 non.
On notera aussi $2 \in \texttt{pairs}$ ou encore $2 \in \{x | (x \in \mathbb{N}) \wedge P(x)\}$ ou encore $2 \in \{t | (t \in \mathbb{N}) \wedge P(t)\}$ ou encore $P(2)$ ou encore $2 \in \{x | x \text{ est un entier naturel et } x \text{ est pair}\}$.

```
def est_pair(n) :
    return n % 2 == 0

def est_petit(n) :
    return n < 20

A = { n for n in range(20) if est_pair(n) }

B = { n for n in range(1000) if est_pair(n) and est_petit(n) }
```

```
In [10]: A == B
Out[10]: True
```

« *est un entier pair* » est une proposition qui ne peut être que vraie ou fausse et qui est définie sur l'ensemble $\mathbb{N}$ des entiers naturels que nous noterons P. Alors 2 vérifie P mais 3 non.
On notera aussi $2 \in$ `pairs` ou encore $2 \in \{x | (x \in \mathbb{N}) \wedge P(x)\}$ ou encore $2 \in \{t | (t \in \mathbb{N}) \wedge P(t)\}$ ou encore $P(2)$ ou encore $2 \in \{x | x \text{ est un entier naturel et } x \text{ est pair}\}$.

```
def est_pair(n) :
    return n % 2 == 0

def est_petit(n) :
    return n < 20

A = { n for n in range(20) if est_pair(n) }

B = { n for n in range(1000) if est_pair(n) and est_petit(n) }
```

```
In [10]: A == B
Out[10]: True
```

« *est un entier pair* » est une proposition qui ne peut être que vraie ou fausse et qui est définie sur l'ensemble $\mathbb{N}$ des entiers naturels que nous noterons P. Alors 2 vérifie P mais 3 non.
On notera aussi $2 \in$ `pairs` ou encore $2 \in \{x | (x \in \mathbb{N}) \wedge P(x)\}$ ou encore $2 \in \{t | (t \in \mathbb{N}) \wedge P(t)\}$ ou encore $P(2)$ ou encore $2 \in \{x | x$ est un entier naturel et x est pair$\}$.

```
def est_pair(n) :
    return n % 2 == 0

def est_petit(n) :
    return n < 20

A = { n for n in range(20) if est_pair(n) }

B = { n for n in range(1000) if est_pair(n) and est_petit(n) }
```

```
In [10]: A == B
Out[10]: True
```

« *est un entier pair* » est une proposition qui ne peut être que vraie ou fausse et qui est définie sur l'ensemble $\mathbb{N}$ des entiers naturels que nous noterons P. Alors 2 vérifie P mais 3 non.
On notera aussi $2 \in$ `pairs` ou encore $2 \in \{x | (x \in \mathbb{N}) \wedge P(x)\}$ ou encore $2 \in \{t | (t \in \mathbb{N}) \wedge P(t)\}$ ou encore $P(2)$ ou encore $2 \in \{x | x$ est un entier naturel et x est pair$\}$.

```
def est_pair(n) :
    return n % 2 == 0

def est_petit(n) :
    return n < 20

A = { n for n in range(20) if est_pair(n) }

B = { n for n in range(1000) if est_pair(n) and est_petit(n) }
```

```
In [10]: A == B
Out[10]: True
```

Ainsi $P(x)$ signifie que l'objet x vérifie la propriété P. Pour signifier que x ne vérifie pas P, on notera $\neg P(x)$ (« non P de x »).
On dit que P est un **prédicat** : c'est une fonction à valeurs dans l'ensemble des booléens.

```
In [14]: est_pair(52)
Out[14]: True

In [15]: est_pair(57)
Out[15]: False

In [16]: 52 in A
Out[16]: False

In [17]: 16 in A
Out[17]: True
```

Ainsi $P(x)$ signifie que l'objet x vérifie la propriété P. Pour signifier que x ne vérifie pas P, on notera $\neg P(x)$ (« non P de x »).
On dit que P est un **prédicat** : c'est une fonction à valeurs dans l'ensemble des booléens.

```
In [14]: est_pair(52)
Out[14]: True

In [15]: est_pair(57)
Out[15]: False

In [16]: 52 in A
Out[16]: False

In [17]: 16 in A
Out[17]: True
```

Ainsi $P(x)$ signifie que l'objet x vérifie la propriété P. Pour signifier que x ne vérifie pas P, on notera $\neg P(x)$ (« non P de x »).
On dit que P est un **prédicat** : c'est une fonction à valeurs dans l'ensemble des booléens.

```
In [14]: est_pair(52)
Out[14]: True

In [15]: est_pair(57)
Out[15]: False

In [16]: 52 in A
Out[16]: False

In [17]: 16 in A
Out[17]: True
```

On notera donc aussi $3 \notin \texttt{pairs}$ ou encore $3 \in \left\{x | (x \in \mathbb{N}) \wedge \neg P(x)\right\}$ ou encore $3 \notin \left\{t | (t \in \mathbb{N}) \wedge P(t)\right\}$ (est-ce équivalent :-) ?) ou encore $\neg P(3)$.

Considérons une autre propriété R : un nombre x vérifie R si, et seulement si, son carré vaut 2.
Quelque soit l'entier x, on a $x^2 \neq 2$. On notera cette propriété par

$$\forall x(x \in \mathbb{N} \wedge \neg R(x))$$

$\forall$ est un *quantificateur universel* (c'est un A à l'envers, car en allemand, ce symbole se lit *für Alle* et a été introduit en 1934 par Gerhard Gentzel...).

Considérons une autre propriété R : un nombre x vérifie R si, et seulement si, son carré vaut 2. Quelque soit l'entier x, on a $x^2 \neq 2$. On notera cette propriété par

$$\forall x(x \in \mathbb{N} \wedge \neg R(x))$$

$\forall$ est un *quantificateur universel* (c'est un A à l'envers, car en allemand, ce symbole se lit *für Alle* et a été introduit en 1934 par Gerhard Gentzel...).

Considérons une autre propriété R : un nombre x vérifie R si, et seulement si, son carré vaut 2. Quelque soit l'entier x, on a $x^2 \neq 2$. On notera cette propriété par

$$\forall x(x \in \mathbb{N} \wedge \neg R(x))$$

$\forall$ est un *quantificateur universel* (c'est un A à l'envers, car en allemand, ce symbole se lit *für Alle* et a été introduit en 1934 par Gerhard Gentzel...).

```python
def nR(x) :
    return x*x != 2
```

```python
def compte_vrai(coll_de_bool):
    """ on utilise le fait que True <-> 1 """
    return sum(coll_de_bool)
```

```python
def pour_tout(pred, ens) :
    return compte_vrai(( pred(e) for e in ens )) == len(ens)
```

```
In [26]: pour_tout(nR, {1,2,3,4,5,6,7,8,9})
Out[26]: True
```

```
In [27]: pour_tout(est_pair, {1,4,6,8})
Out[27]: False
```

```python
def nR(x) :
    return x*x != 2
```

```python
def compte_vrai(coll_de_bool):
    """ on utilise le fait que True <-> 1 """
    return sum(coll_de_bool)
```

```python
def pour_tout(pred, ens) :
    return compte_vrai(( pred(e) for e in ens )) == len(ens)
```

```
In [26]: pour_tout(nR, {1,2,3,4,5,6,7,8,9})
Out[26]: True
```

```
In [27]: pour_tout(est_pair, {2,4,6,7})
Out[27]: False
```

```python
def nR(x) :
    return x*x != 2
```

```python
def compte_vrai(coll_de_bool):
    """ on utilise le fait que True <-> 1 """
    return sum(coll_de_bool)
```

```python
def pour_tout(pred, ens) :
    return compte_vrai(( pred(e) for e in ens )) == len(ens)
```

```
In [26]: pour_tout(nR, {1,2,3,4,5,6,7,8,9})
Out[26]: True
```

```
In [27]: pour_tout(est_pair, {2,4,6,7})
Out[27]: False
```

```python
def nR(x) :
    return x*x != 2
```

```python
def compte_vrai(coll_de_bool):
    """ on utilise le fait que True <-> 1 """
    return sum(coll_de_bool)
```

```python
def pour_tout(pred, ens) :
    return compte_vrai(( pred(e) for e in ens )) == len(ens)
```

```
In [26]: pour_tout(nR, {1,2,3,4,5,6,7,8,9})
Out[26]: True
```

```
In [27]: pour_tout(est_pair, {2,4,6,7})
Out[27]: False
```

```python
def nR(x) :
    return x*x != 2
```

```python
def compte_vrai(coll_de_bool):
    """ on utilise le fait que True <-> 1 """
    return sum(coll_de_bool)
```

```python
def pour_tout(pred, ens) :
    return compte_vrai(( pred(e) for e in ens )) == len(ens)
```

```
In [26]: pour_tout(nR, {1,2,3,4,5,6,7,8,9})
Out[26]: True
```

```
In [27]: pour_tout(est_pair, {2,4,6,7})
Out[27]: False
```

Considérons une autre propriété S : un nombre x vérifie S si, et seulement si, son carré vaut 4.
Il existe au moins un entier qui vérifie S : il s'agit de 2. On note alors

$$\exists x(x \in \mathbb{N} \wedge S(x))$$

$\exists$ est l'autre quantificateur universel introduit par Giuseppe Peano en 1897 (un E à l'envers comme dans *Esiste almeno uno*).
On peut même préciser que cet entier est *unique* en faisant suivre le quantificateur d'un point d'exclamation : $\exists! x(x \in \mathbb{N} \wedge S(x))$

Considérons une autre propriété S : un nombre x vérifie S si, et seulement si, son carré vaut 4.
Il existe au moins un entier qui vérifie S : il s'agit de 2. On note alors

$$\exists x(x \in \mathbb{N} \wedge S(x))$$

$\exists$ est l'autre quantificateur universel introduit par Giuseppe Peano en 1897 (un E à l'envers comme dans *Esiste almeno uno*).
On peut même préciser que cet entier est *unique* en faisant suivre le quantificateur d'un point d'exclamation : $\exists! x(x \in \mathbb{N} \wedge S(x))$

Considérons une autre propriété S : un nombre x vérifie S si, et seulement si, son carré vaut 4.
Il existe au moins un entier qui vérifie S : il s'agit de 2. On note alors

$$\exists x(x \in \mathbb{N} \wedge S(x))$$

$\exists$ est l'autre quantificateur universel introduit par Giuseppe Peano en 1897 (un E à l'envers comme dans *Esiste almeno uno*).
On peut même préciser que cet entier est *unique* en faisant suivre le quantificateur d'un point d'exclamation : $\exists! x(x \in \mathbb{N} \wedge S(x))$

Considérons une autre propriété S : un nombre x vérifie S si, et seulement si, son carré vaut 4.
Il existe au moins un entier qui vérifie S : il s'agit de 2. On note alors

$$\exists x(x \in \mathbb{N} \wedge S(x))$$

$\exists$ est l'autre quantificateur universel introduit par Giuseppe Peano en 1897 (un E à l'envers comme dans *Esiste almeno uno*).
On peut même préciser que cet entier est *unique* en faisant suivre le quantificateur d'un point d'exclamation : $\exists! x(x \in \mathbb{N} \wedge S(x))$

`il_existe(pred,ens)` ???... En TD...

Implication

Recherche

Soit un entier naturel n inférieur à 20. On considère la proposition « si n est pair, alors son successeur est premier ». Quels sont les entiers qui rendent cette proposition vraie ?

Implication

- SI A ALORS B
- IF A THEN B
- A IMPLIQU*E* B
- $A \Longrightarrow B$
- SI je me baigne ALORS je suis mouillé
- et si je ne me baigne pas ?
-

A	B	$A \Longrightarrow B$
Vrai	Vrai	Vrai
Vrai	Faux	Faux
Faux	Vrai	Vrai
Faux	Faux	Vrai

- $\neg A \vee B$
- Cours de M. Attiogbe

Implication

```
def implique(A,B) :
    return (not A) or B
```

```
inf20 = set( range(21) )

prem20 = { 2, 3, 5, 7, 11, 13, 17, 19 }

question = { n for n in inf20 if implique(est_pair(n), n + 1 in prem20) }
```

```
In [18]: question
Out[18]: {1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19}
```

Implication

```
def implique(A,B) :
    return (not A) or B
```

```
inf20 = set( range(21) )

prem20 = { 2, 3, 5, 7, 11, 13, 17, 19 }

question = { n for n in inf20 if implique(est_pair(n), n + 1 in prem20) }
```

```
In [18]: question
Out[18]: {1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19}
```

Implication

```
def implique(A,B) :
    return (not A) or B
```

```
inf20 = set( range(21) )

prem20 = { 2, 3, 5, 7, 11, 13, 17, 19 }

question = { n for n in inf20 if implique(est_pair(n), n + 1 in prem20) }
```

```
In [18]: question
Out[18]: {1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19}
```

Équivalence

```
def equiv(A, B) :
    return implique(A, B) and implique(B, A)
```

Sommaire

Observons :

```
In [8]: 37 == 37.0
Out[8]: True

In [9]: type(37)
Out[9]: int

In [10]: type(37.0)
Out[10]: float
```

Nous consacrerons quelques semaines à l'étude des nombres à virgule flottante au printemps prochain et nous allons juste aujourd'hui aborder quelques notions indispensables.
Ces notions seront également vues en ISI avec M. Jézequel.

Observons :

```
In [8]: 37 == 37.0
Out[8]: True

In [9]: type(37)
Out[9]: int

In [10]: type(37.0)
Out[10]: float
```

Nous consacrerons quelques semaines à l'étude des nombres à virgule flottante au printemps prochain et nous allons juste aujourd'hui aborder quelques notions indispensables.
Ces notions seront également vues en ISI avec M. Jézequel.

Observons :

```
In [8]: 37 == 37.0
Out[8]: True

In [9]: type(37)
Out[9]: int

In [10]: type(37.0)
Out[10]: float
```

Nous consacrerons quelques semaines à l'étude des nombres à virgule flottante au printemps prochain et nous allons juste aujourd'hui aborder quelques notions indispensables.
Ces notions seront également vues en ISI avec M. Jézequel.

Base de numération

On rappelle que l'écriture en base β d'un nombre n est définie par la donnée de l'unique famille $(a_0, a_1, ..., a_k)$ vérifiant :

$$n = a_k\beta^k + a_{k-1}\beta^{k-1} + \cdots + a_2\beta^2 + a_1\beta + a_0 = \sum_{i=0}^{i=k} a_i \times \beta^i$$

avec $a_k \neq 0$ et $0 \leqslant a_i < \beta$ pour tout $i \in \{0, 1, .., k\}$.

Base de numération

On rappelle que l'écriture en base β d'un nombre n est définie par la donnée de l'unique famille $(a_0, a_1, ..., a_k)$ vérifiant :

$$n = a_k\beta^k + a_{k-1}\beta^{k-1} + \cdots + a_2\beta^2 + a_1\beta + a_0 = \sum_{i=0}^{i=k} a_i \times \beta^i$$

avec $a_k \neq 0$ et $0 \leqslant a_i < \beta$ pour tout $i \in \{0, 1, .., k\}$.

Par exemple, 100101 en base 2 est le nombre :

$$1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 32 + 4 + 1 = 37$$

et en base 8 ?

$$1 \times 8^5 + 0 \times 8^4 + 0 \times 8^3 + 1 \times 8^2 + 0 \times 8^1 + 1 \times 2^0 = 32768 + 64 + 1 = 32833$$

et en base 10 ?

$$1 \times 10^5 + 0 \times 10^4 + 0 \times 10^3 + 1 \times 10^2 + 0 \times 10^1 + 1 \times 2^0 = 100000 + 100 + 1 = 100101$$

Par exemple, 100101 en base 2 est le nombre :

$$1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 32 + 4 + 1 = 37$$

et en base 8 ?

$$1 \times 8^5 + 0 \times 8^4 + 0 \times 8^3 + 1 \times 8^2 + 0 \times 8^1 + 1 \times 2^0 = 32768 + 64 + 1 = 32833$$

et en base 10 ?

$$1 \times 10^5 + 0 \times 10^4 + 0 \times 10^3 + 1 \times 10^2 + 0 \times 10^1 + 1 \times 2^0 = 100000 + 100 + 1 = 100101$$

Par exemple, 100101 en base 2 est le nombre :

$$1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 32 + 4 + 1 = 37$$

et en base 8 ?

$$1 \times 8^5 + 0 \times 8^4 + 0 \times 8^3 + 1 \times 8^2 + 0 \times 8^1 + 1 \times 2^0 = 32768 + 64 + 1 = 32833$$

et en base 10 ?

$$1 \times 10^5 + 0 \times 10^4 + 0 \times 10^3 + 1 \times 10^2 + 0 \times 10^1 + 1 \times 2^0 = 100000 + 100 + 1 = 100101$$

Par exemple, 100101 en base 2 est le nombre :

$$1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 32 + 4 + 1 = 37$$

et en base 8 ?

$$1 \times 8^5 + 0 \times 8^4 + 0 \times 8^3 + 1 \times 8^2 + 0 \times 8^1 + 1 \times 2^0 = 32768 + 64 + 1 = 32833$$

et en base 10 ?

$$1 \times 10^5 + 0 \times 10^4 + 0 \times 10^3 + 1 \times 10^2 + 0 \times 10^1 + 1 \times 2^0 = 100000 + 100 + 1 = 100101$$

Par exemple, 100101 en base 2 est le nombre :

$$1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 32 + 4 + 1 = 37$$

et en base 8 ?

$$1 \times 8^5 + 0 \times 8^4 + 0 \times 8^3 + 1 \times 8^2 + 0 \times 8^1 + 1 \times 2^0 = 32768 + 64 + 1 = 32833$$

et en base 10 ?

$$1 \times 10^5 + 0 \times 10^4 + 0 \times 10^3 + 1 \times 10^2 + 0 \times 10^1 + 1 \times 2^0 = 100000 + 100 + 1 = 100101$$

Par exemple, 100101 en base 2 est le nombre :

$$1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 32 + 4 + 1 = 37$$

et en base 8 ?

$$1 \times 8^5 + 0 \times 8^4 + 0 \times 8^3 + 1 \times 8^2 + 0 \times 8^1 + 1 \times 2^0 = 32768 + 64 + 1 = 32833$$

et en base 10 ?

$$1 \times 10^5 + 0 \times 10^4 + 0 \times 10^3 + 1 \times 10^2 + 0 \times 10^1 + 1 \times 2^0 = 100000 + 100 + 1 = 100101$$

Par exemple, 100101 en base 2 est le nombre :

$$1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 32 + 4 + 1 = 37$$

et en base 8 ?

$$1 \times 8^5 + 0 \times 8^4 + 0 \times 8^3 + 1 \times 8^2 + 0 \times 8^1 + 1 \times 2^0 = 32768 + 64 + 1 = 32833$$

et en base 10 ?

$$1 \times 10^5 + 0 \times 10^4 + 0 \times 10^3 + 1 \times 10^2 + 0 \times 10^1 + 1 \times 2^0 = 100000 + 100 + 1 = 100101$$

Par exemple, 100101 en base 2 est le nombre :

$$1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 32 + 4 + 1 = 37$$

et en base 8 ?

$$1 \times 8^5 + 0 \times 8^4 + 0 \times 8^3 + 1 \times 8^2 + 0 \times 8^1 + 1 \times 2^0 = 32768 + 64 + 1 = 32833$$

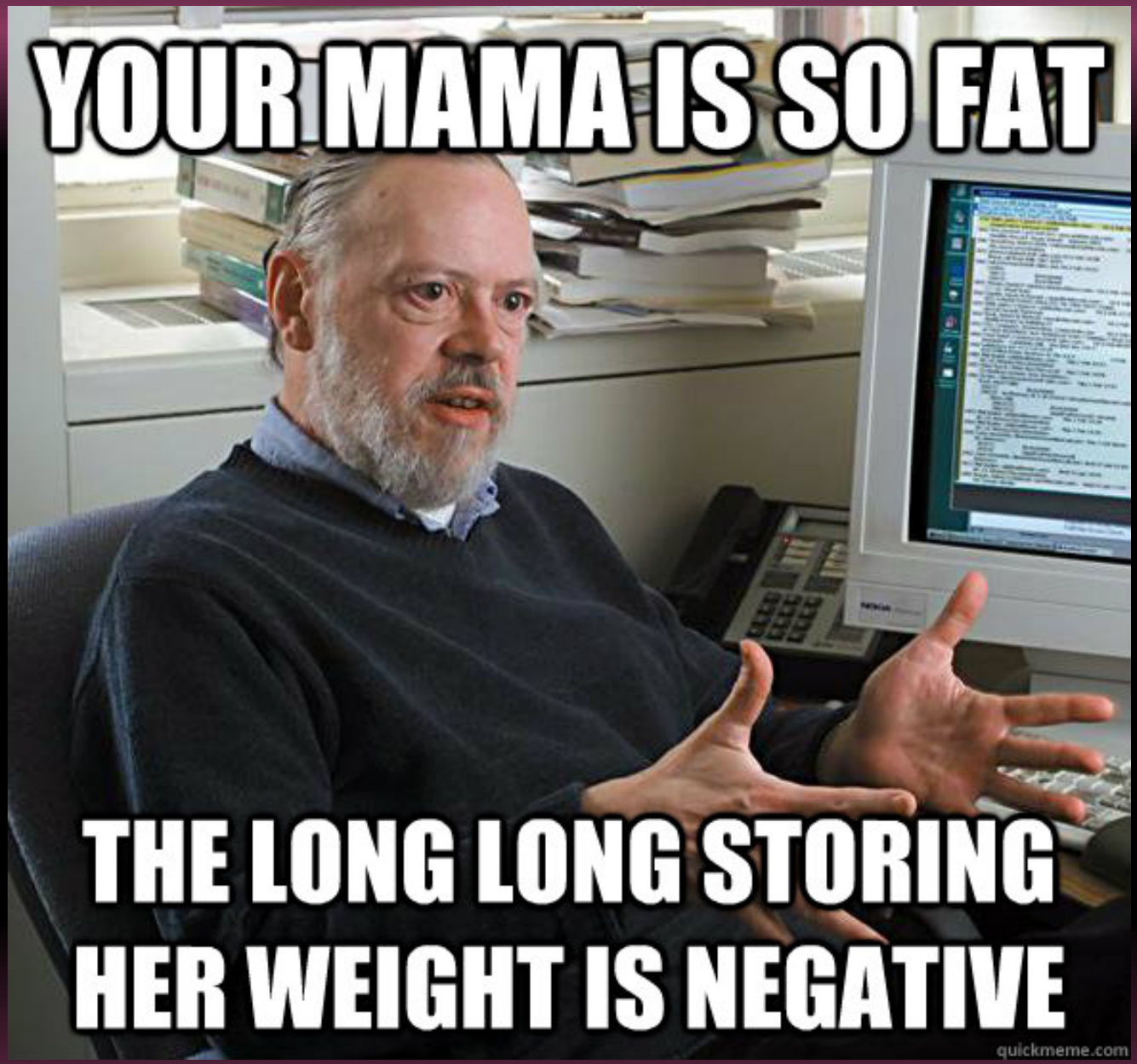
et en base 10 ?

$$1 \times 10^5 + 0 \times 10^4 + 0 \times 10^3 + 1 \times 10^2 + 0 \times 10^1 + 1 \times 2^0 = 100000 + 100 + 1 = 100101$$

Sur machine, les entiers peuvent être représentés de différentes manières.
En C par exemple il y a les types `int`, `signed int`, `unsigned int`, `long`, `long long`, etc.
Vous verrez ça avec M. Morin.

Sur machine, les entiers peuvent être représentés de différentes manières.
En C par exemple il y a les types `int`, `signed int`, `unsigned int`, `long`, `long long`, etc.
Vous verrez ça avec M. Morin.

Sur machine, les entiers peuvent être représentés de différentes manières.
En C par exemple il y a les types `int`, `signed int`, `unsigned int`, `long`, `long long`, etc.
Vous verrez ça avec M. Morin.


YOUR MAMA IS SO FAT
THE LONG LONG STORING
HER WEIGHT IS NEGATIVE
quickmeme.com

Pour comprendre cet exemple d'humour geek qui fera votre succès lors du tonus des infirmières, il faut savoir qu'un nombre ou tout autre objet est stocké sur un nombre fini de bits.
Par exemple, les `long long` sont stockés sur 64 bits, sachant que le premier bit indique le signe : 0 pour positif et 1 pour négatif.
Quel est le plus grand `long long` ?

0	1	1	...	1	1

$$1 \times 2^{62} + 1 \times 2^{61} + \cdots + 1 = 2^{63} - 1 = 9\,223\,372\,036\,854\,775\,807$$

et le suivant ?

1	0	0	...	0	0

qui est donc négatif.
C'est ce genre de problème qui a fait exploser la fusée Ariane V lors de son vol inaugural.

Pour comprendre cet exemple d'humour geek qui fera votre succès lors du tonus des infirmières, il faut savoir qu'un nombre ou tout autre objet est stocké sur un nombre fini de bits.
Par exemple, les `long long` sont stockés sur 64 bits, sachant que le premier bit indique le signe : 0 pour positif et 1 pour négatif.
Quel est le plus grand `long long` ?

0	1	1	...	1	1

$$1 \times 2^{62} + 1 \times 2^{61} + \cdots + 1 = 2^{63} - 1 = 9\,223\,372\,036\,854\,775\,807$$

et le suivant ?

1	0	0	...	0	0

qui est donc négatif.
C'est ce genre de problème qui a fait exploser la fusée Ariane V lors de son vol inaugural.

Pour comprendre cet exemple d'humour geek qui fera votre succès lors du tonus des infirmières, il faut savoir qu'un nombre ou tout autre objet est stocké sur un nombre fini de bits.
Par exemple, les `long long` sont stockés sur 64 bits, sachant que le premier bit indique le signe : 0 pour positif et 1 pour négatif.
Quel est le plus grand `long long` ?

$$1 \times 2^{62} + 1 \times 2^{61} + \cdots + 1 = 2^{63} - 1 = 9\,223\,372\,036\,854\,775\,807$$

et le suivant ?

qui est donc négatif.
C'est ce genre de problème qui a fait exploser la fusée Ariane V lors de son vol inaugural.

Pour comprendre cet exemple d'humour geek qui fera votre succès lors du tonus des infirmières, il faut savoir qu'un nombre ou tout autre objet est stocké sur un nombre fini de bits.
Par exemple, les `long long` sont stockés sur 64 bits, sachant que le premier bit indique le signe : 0 pour positif et 1 pour négatif.
Quel est le plus grand `long long` ?

$$1 \times 2^{62} + 1 \times 2^{61} + \cdots + 1 = 2^{63} - 1 = 9\,223\,372\,036\,854\,775\,807$$

et le suivant ?

qui est donc négatif.
C'est ce genre de problème qui a fait exploser la fusée Ariane V lors de son vol inaugural.

Pour comprendre cet exemple d'humour geek qui fera votre succès lors du tonus des infirmières, il faut savoir qu'un nombre ou tout autre objet est stocké sur un nombre fini de bits.
Par exemple, les `long long` sont stockés sur 64 bits, sachant que le premier bit indique le signe : 0 pour positif et 1 pour négatif.
Quel est le plus grand `long long` ?

0	1	1	⋯	1	1

$$1 \times 2^{62} + 1 \times 2^{61} + \cdots + 1 = 2^{63} - 1 = 9\,223\,372\,036\,854\,775\,807$$

et le suivant ?

1	0	0	⋯	0	0

qui est donc négatif.
C'est ce genre de problème qui a fait exploser la fusée Ariane V lors de son vol inaugural.

Pour comprendre cet exemple d'humour geek qui fera votre succès lors du tonus des infirmières, il faut savoir qu'un nombre ou tout autre objet est stocké sur un nombre fini de bits.
Par exemple, les `long long` sont stockés sur 64 bits, sachant que le premier bit indique le signe : 0 pour positif et 1 pour négatif.
Quel est le plus grand `long long` ?

0	1	1	⋯	1	1

$$1 \times 2^{62} + 1 \times 2^{61} + \cdots + 1 = 2^{63} - 1 = 9\,223\,372\,036\,854\,775\,807$$

et le suivant ?

1	0	0	⋯	0	0

qui est donc négatif.
C'est ce genre de problème qui a fait exploser la fusée Ariane V lors de son vol inaugural.

Pour comprendre cet exemple d'humour geek qui fera votre succès lors du tonus des infirmières, il faut savoir qu'un nombre ou tout autre objet est stocké sur un nombre fini de bits.
Par exemple, les `long long` sont stockés sur 64 bits, sachant que le premier bit indique le signe : 0 pour positif et 1 pour négatif.
Quel est le plus grand `long long` ?

0	1	1	⋯	1	1

$$1 \times 2^{62} + 1 \times 2^{61} + \cdots + 1 = 2^{63} - 1 = 9\,223\,372\,036\,854\,775\,807$$

et le suivant ?

1	0	0	⋯	0	0

qui est donc négatif.
C'est ce genre de problème qui a fait exploser la fusée Ariane V lors de son vol inaugural.

Pour comprendre cet exemple d'humour geek qui fera votre succès lors du tonus des infirmières, il faut savoir qu'un nombre ou tout autre objet est stocké sur un nombre fini de bits.
Par exemple, les `long long` sont stockés sur 64 bits, sachant que le premier bit indique le signe : 0 pour positif et 1 pour négatif.
Quel est le plus grand `long long` ?

0	1	1	⋯	1	1

$$1 \times 2^{62} + 1 \times 2^{61} + \cdots + 1 = 2^{63} - 1 = 9\,223\,372\,036\,854\,775\,807$$

et le suivant ?

1	0	0	⋯	0	0

qui est donc négatif.
C'est ce genre de problème qui a fait exploser la fusée Ariane V lors de son vol inaugural.

Pour comprendre cet exemple d'humour geek qui fera votre succès lors du tonus des infirmières, il faut savoir qu'un nombre ou tout autre objet est stocké sur un nombre fini de bits.
Par exemple, les `long long` sont stockés sur 64 bits, sachant que le premier bit indique le signe : 0 pour positif et 1 pour négatif.
Quel est le plus grand `long long` ?

0	1	1	⋯	1	1

$$1 \times 2^{62} + 1 \times 2^{61} + \cdots + 1 = 2^{63} - 1 = 9\,223\,372\,036\,854\,775\,807$$

et le suivant ?

1	0	0	⋯	0	0

qui est donc négatif.
C'est ce genre de problème qui a fait exploser la fusée Ariane V lors de son vol inaugural.

Pour comprendre cet exemple d'humour geek qui fera votre succès lors du tonus des infirmières, il faut savoir qu'un nombre ou tout autre objet est stocké sur un nombre fini de bits.
Par exemple, les `long long` sont stockés sur 64 bits, sachant que le premier bit indique le signe : 0 pour positif et 1 pour négatif.
Quel est le plus grand `long long` ?

0	1	1	⋯	1	1

$$1 \times 2^{62} + 1 \times 2^{61} + \cdots + 1 = 2^{63} - 1 = 9\,223\,372\,036\,854\,775\,807$$

et le suivant ?

1	0	0	⋯	0	0

qui est donc négatif.
C'est ce genre de problème qui a fait exploser la fusée Ariane V lors de son vol inaugural.

En C :

```c
#include <stdio.h>

int
main(void)
{
  long long a = 92233720368547758O7 ;
  long long b = 1 ;

  long long d ;  d = a + b ;
  long long e ;  e = a + b + b ;

  printf("2**63 - 1 = %lli\n2**63   = %lli\n2**63 + 1 = %lli\n", a, d, e );

  return 0;
}
```

```
2**63 - 1 =  9223372036854775807
2**63     = -9223372036854775808
2**63 + 1 = -9223372036854775807
```

En C :

```c
#include <stdio.h>

int
main(void)
{
  long long a = 92233720368547758O7 ;
  long long b = 1 ;

  long long d ;  d = a + b ;
  long long e ;  e = a + b + b ;

  printf("2**63 - 1 = %lli\n2**63   = %lli\n2**63 + 1 = %lli\n", a, d, e );

  return 0;
}
```

```
2**63 - 1 =  9223372036854775807
2**63     = -9223372036854775808
2**63 + 1 = -9223372036854775807
```

Pourtant sur Python :

```
In [24]: 2**63 + 1
Out[24]: 9223372036854775809

In [25]: 2**63 + 10
Out[25]: 9223372036854775818
```

C'est que Python passe, sans nous le dire, en précision infinie lorsque la taille limite est atteinte. La taille limite d'un entier n'est plus 64 bits mais la taille de la mémoire disponible !
Voici un extrait de la PEP 237 :

> *Python currently distinguishes between two kinds of integers (ints) : regular or short ints, limited by the size of a C long (typically 32 or 64 bits), and long ints, which are limited only by available memory. When operations on short ints yield results that don't fit in a C long, they raise an error. There are some other distinctions too. This PEP proposes to do away with most of the differences in semantics, unifying the two types from the perspective of the Python user.*

Pourtant sur Python :

```
In [24]: 2**63 + 1
Out[24]: 9223372036854775809

In [25]: 2**63 + 10
Out[25]: 9223372036854775818
```

C'est que Python passe, sans nous le dire, en précision infinie lorsque la taille limite est atteinte. La taille limite d'un entier n'est plus 64 bits mais la taille de la mémoire disponible !

Voici un extrait de la PEP 237 :

> *Python currently distinguishes between two kinds of integers (ints) : regular or short ints, limited by the size of a C long (typically 32 or 64 bits), and long ints, which are limited only by available memory. When operations on short ints yield results that don't fit in a C long, they raise an error. There are some other distinctions too. This PEP proposes to do away with most of the differences in semantics, unifying the two types from the perspective of the Python user.*

Pourtant sur Python :

```
In [24]: 2**63 + 1
Out[24]: 9223372036854775809

In [25]: 2**63 + 10
Out[25]: 9223372036854775818
```

C'est que Python passe, sans nous le dire, en précision infinie lorsque la taille limite est atteinte. La taille limite d'un entier n'est plus 64 bits mais la taille de la mémoire disponible !
Voici un extrait de la PEP 237 :

> *Python currently distinguishes between two kinds of integers (ints) : regular or short ints, limited by the size of a C long (typically 32 or 64 bits), and long ints, which are limited only by available memory. When operations on short ints yield results that don't fit in a C long, they raise an error. There are some other distinctions too. This PEP proposes to do away with most of the differences in semantics, unifying the two types from the perspective of the Python user.*

Passons aux flottants :

```
In [26]: 2**63 + 1.0
Out[26]: 9.223372036854776e+18

In [27]: 2**63 + 10.0
Out[27]: 9.223372036854776e+18

In [28]: 2**63 + 10.0 == 2**63 + 1.0
Out[28]: True
```

Argh....

```
In [29]: int(2**63 + 10.0)
Out[29]: 9223372036854775808

In [30]: 2**63 + 10
Out[30]: 9223372036854775818
```

gloups...

```
In [31]: 3*0.1 == 0.3
Out[31]: False
```

Garp...

```
In [26]: 2**63 + 1.0
Out[26]: 9.223372036854776e+18

In [27]: 2**63 + 10.0
Out[27]: 9.223372036854776e+18

In [28]: 2**63 + 10.0 == 2**63 + 1.0
Out[28]: True
```

Argh....

```
In [29]: int(2**63 + 10.0)
Out[29]: 9223372036854775808

In [30]: 2**63 + 10
Out[30]: 9223372036854775818
```

gloups...

```
In [31]: 3*0.1 == 0.3
Out[31]: False
```

Garp...

```
In [26]: 2**63 + 1.0
Out[26]: 9.223372036854776e+18

In [27]: 2**63 + 10.0
Out[27]: 9.223372036854776e+18

In [28]: 2**63 + 10.0 == 2**63 + 1.0
Out[28]: True
```

Argh....

```
In [29]: int(2**63 + 10.0)
Out[29]: 9223372036854775808

In [30]: 2**63 + 10
Out[30]: 9223372036854775818
```

gloups...

```
In [31]: 3*0.1 == 0.3
Out[31]: False
```

Garp...

```
In [26]: 2**63 + 1.0
Out[26]: 9.223372036854776e+18

In [27]: 2**63 + 10.0
Out[27]: 9.223372036854776e+18

In [28]: 2**63 + 10.0 == 2**63 + 1.0
Out[28]: True
```

Argh....

```
In [29]: int(2**63 + 10.0)
Out[29]: 9223372036854775808

In [30]: 2**63 + 10
Out[30]: 9223372036854775818
```

gloups...

```
In [31]: 3*0.1 == 0.3
Out[31]: False
```

Garp...

Nous en reparlerons calmement au printemps, chaque chose en son temps...
Nous nous contenterons de travailler sur des entiers le plus possible d'ici-là.

Nous en reparlerons calmement au printemps, chaque chose en son temps...
Nous nous contenterons de travailler sur des entiers le plus possible d'ici-là.

Sommaire

Les chaînes de caractères en Python sont des suites de caractères...
Les caractères sont ceux du clavier mais aussi tous ceux du grand ensemble de caractères qu'on appelle *Unicode*.
Nous nous contenterons de ceux de notre clavier pour l'instant.

Les chaînes de caractères en Python sont des suites de caractères...
Les caractères sont ceux du clavier mais aussi tous ceux du grand ensemble de caractères qu'on appelle *Unicode*.
Nous nous contenterons de ceux de notre clavier pour l'instant.

Les chaînes de caractères en Python sont des suites de caractères...
Les caractères sont ceux du clavier mais aussi tous ceux du grand ensemble de caractères qu'on appelle *Unicode*.
Nous nous contenterons de ceux de notre clavier pour l'instant.

```
In [33]: type('é')
Out[33]: str

In [34]: "Bonjour"
Out[34]: 'Bonjour'

In [35]: 'Bonjour'
Out[35]: 'Bonjour'

In [36]: 'Bonjour' == "Bonjour"
Out[36]: True

In [37]: ' Bonjour "Le Monde" '
Out[37]: ' Bonjour "Le Monde" '

In [38]: " Bonjour 'Le Monde' "
Out[38]: " Bonjour 'Le Monde' "

In [39]: " Bonjour l'ami "
Out[39]: " Bonjour l'ami "
```

```
In [44]: "J'aime " + "les gâteaux"
Out[44]: "J'aime les gâteaux"

In [45]: "J'aime " * 3
Out[45]: "J'aime J'aime J'aime "

In [46]: "J'aime " * 3 + "les gâteaux"
Out[46]: "J'aime J'aime J'aime les gâteaux"
```

```
In [47]: a, b = "J'aime ", "les gâteaux"

In [48]: 3*a + b
Out[48]: "J'aime J'aime J'aime les gâteaux"
```

```
In [50]: a, b = 2, 10

In [51]: 3*a + b
Out[51]: 16
```

```
In [52]: a, b = [2], [10]

In [53]: 3*a + b
Out[53]: [2, 2, 2, 10]
```

```
In [47]: a, b = "J'aime ", "les gâteaux"

In [48]: 3*a + b
Out[48]: "J'aime J'aime J'aime les gâteaux"
```

```
In [50]: a, b = 2, 10

In [51]: 3*a + b
Out[51]: 16
```

```
In [52]: a, b = [2], [10]

In [53]: 3*a + b
Out[53]: [2, 2, 2, 10]
```

```
In [47]: a, b = "J'aime ", "les gâteaux"

In [48]: 3*a + b
Out[48]: "J'aime J'aime J'aime les gâteaux"
```

```
In [50]: a, b = 2, 10

In [51]: 3*a + b
Out[51]: 16
```

```
In [52]: a, b = [2], [10]

In [53]: 3*a + b
Out[53]: [2, 2, 2, 10]
```

```
In [54]: a, b = 2, "les gâteaux"

In [55]: 3*a + b
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-55-61f87e631998> in <module>()
----> 1 3*a + b

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Sommaire

Départ DOMAINE

Arrivée CODOMAINE

Une fonction est une relation

C'est donc une partie (sous-ensemble) de DOMAINE ⊗ CODOMAINE

...mais il n'y a pas deux couples ayant le même premier élément

Départ DOMAINE

Arrivée CODOMAINE

Une fonction est une relation

C'est donc une partie (sous-ensemble) de DOMAINE ⊗ CODOMAINE

...mais il n'y a pas deux couples ayant le même premier élément

Départ DOMAINE
Arrivée CODOMAINE
Une fonction est une relation
C'est donc une partie (sous-ensemble) de DOMAINE ⊗ CODOMAINE
...mais il n'y a pas deux couples ayant le même premier élément

Départ DOMAINE
Arrivée CODOMAINE
Une fonction est une relation
C'est donc une partie (sous-ensemble) de DOMAINE ⊗ CODOMAINE
...mais il n'y a pas deux couples ayant le même premier élément

Départ DOMAINE
Arrivée CODOMAINE
Une fonction est une relation
C'est donc une partie (sous-ensemble) de DOMAINE ⊗ CODOMAINE
...mais il n'y a pas deux couples ayant le même premier élément

Départ DOMAINE
Arrivée CODOMAINE
Une fonction est une relation
C'est donc une partie (sous-ensemble) de DOMAINE ⊗ CODOMAINE
...mais il n'y a pas deux couples ayant le même premier élément

Départ DOMAINE
Arrivée CODOMAINE
Une fonction est une relation
C'est donc une partie (sous-ensemble) de DOMAINE $\otimes$ CODOMAINE
...mais il n'y a pas deux couples ayant le même premier élément

Départ DOMAINE
Arrivée CODOMAINE
Une fonction est une relation
C'est donc une partie (sous-ensemble) de DOMAINE ⊗ CODOMAINE
...mais il n'y a pas deux couples ayant le même premier élément

Départ DOMAINE
Arrivée CODOMAINE
Une fonction est une relation
C'est donc une partie (sous-ensemble) de DOMAINE $\otimes$ CODOMAINE
...mais il n'y a pas deux couples ayant le même premier élément

Donc la fonction *double* de domaine $\{1, 2, 3, ...\}$ est

$$\{(1, 2), (2, 4), (3, 6), ...\}$$

Donc la fonction *double* de domaine $\{1, 2, 3, ...\}$ est

$$\{(1, 2), (2, 4), (3, 6), ...\}$$

La fonction *addition* de domaine $\{1,2,3,...\} \otimes \{1,2,3,...\}$ est

$$\{((1,1),2),((1,2),3),...((5,7),12),((5,8),13),...\}$$

La fonction *addition* de domaine $\{1,2,3,...\} \otimes \{1,2,3,...\}$ est

$$\{((1,1),2),((1,2),3),...((5,7),12),((5,8),13),...\}$$

Mais bon, ça manque un peu de dynamisme.

Départ $\longrightarrow$ Arrivée

Départ $\longrightarrow$ Arrivée

Exemple 3

On veut mettre une chaîne de caractères en majuscules

- On commence par créer une fonction qui met une chaîne de longueur 1 en majuscule.
- Cela ne concerne que les lettres minuscules.
- On manipule les unicodes correspondant.
- Les lettres minuscules ont un code entre 97 et 97 + 26
- La majuscule correspondant a un code diminué de 2^5
- La fonction doit retourner une chaîne de longueur 1

Exemple 3

On veut mettre une chaîne de caractères en majuscules

- On commence par créer une fonction qui met une chaîne de longueur 1 en majuscule.
- Cela ne concerne que les lettres minuscules.
- On manipule les unicodes correspondant.
- Les lettres minuscules ont un code entre 97 et $97 + 26$
- La majuscule correspondant a un code diminué de 2^5
- La fonction doit retourner une chaîne de longueur 1

Exemple 3

On veut mettre une chaîne de caractères en majuscules

- On commence par créer une fonction qui met une chaîne de longueur 1 en majuscule.
- Cela ne concerne que les lettres minuscules.
- On manipule les unicodes correspondant.
- Les lettres minuscules ont un code entre 97 et $97 + 26$
- La majuscule correspondant a un code diminué de 2^5
- La fonction doit retourner une chaîne de longueur 1

Exemple 3

On veut mettre une chaîne de caractères en majuscules

- On commence par créer une fonction qui met une chaîne de longueur 1 en majuscule.
- Cela ne concerne que les lettres minuscules.
- On manipule les unicodes correspondant.
- Les lettres minuscules ont un code entre 97 et $97 + 26$
- La majuscule correspondant a un code diminué de 2^5
- La fonction doit retourner une chaîne de longueur 1

Exemple 3

On veut mettre une chaîne de caractères en majuscules

- On commence par créer une fonction qui met une chaîne de longueur 1 en majuscule.
- Cela ne concerne que les lettres minuscules.
- On manipule les unicodes correspondant.
- Les lettres minuscules ont un code entre 97 et $97 + 26$
- La majuscule correspondant a un code diminué de 2^5
- La fonction doit retourner une chaîne de longueur 1

Exemple 3

On veut mettre une chaîne de caractères en majuscules

- On commence par créer une fonction qui met une chaîne de longueur 1 en majuscule.
- Cela ne concerne que les lettres minuscules.
- On manipule les unicodes correspondant.
- Les lettres minuscules ont un code entre 97 et $97 + 26$
- La majuscule correspondant a un code diminué de 2^5
- La fonction doit retourner une chaîne de longueur 1

Exemple 3

On veut mettre une chaîne de caractères en majuscules

- On commence par créer une fonction qui met une chaîne de longueur 1 en majuscule.
- Cela ne concerne que les lettres minuscules.
- On manipule les unicodes correspondant.
- Les lettres minuscules ont un code entre 97 et $97 + 26$
- La majuscule correspondant a un code diminué de 2^5
- La fonction doit retourner une chaîne de longueur 1

```
In [62]: ?chr
Docstring:
chr(i) -> Unicode character

Return a Unicode string of one character with ordinal i; 0 <= i <= 0x10ffff.
Type:      builtin_function_or_method
```

```
In [63]: ?ord
Docstring:
ord(c) -> integer

Return the integer ordinal of a one-character string.
Type:      builtin_function_or_method
```

```
Prelude Data.Char> :t chr
chr :: Int -> Char
Prelude Data.Char> :t ord
ord :: Char -> Int
```

```
In [62]: ?chr
Docstring:
chr(i) -> Unicode character

Return a Unicode string of one character with ordinal i; 0 <= i <= 0x10ffff.
Type:      builtin_function_or_method
```

```
In [63]: ?ord
Docstring:
ord(c) -> integer

Return the integer ordinal of a one-character string.
Type:      builtin_function_or_method
```

```
Prelude Data.Char> :t chr
chr :: Int -> Char
Prelude Data.Char> :t ord
ord :: Char -> Int
```

```
In [62]: ?chr
Docstring:
chr(i) -> Unicode character

Return a Unicode string of one character with ordinal i; 0 <= i <= 0x10ffff.
Type:      builtin_function_or_method
```

```
In [63]: ?ord
Docstring:
ord(c) -> integer

Return the integer ordinal of a one-character string.
Type:      builtin_function_or_method
```

```
Prelude Data.Char> :t chr
chr :: Int -> Char
Prelude Data.Char> :t ord
ord :: Char -> Int
```

```python
def car_en_maj(car) :
    ascii = ord(car)
    if 97 <= ascii <= 97 + 26 :
        return chr(ascii - 32)
    else :
        return car
```

```python
def car_en_maj(car) :
    ascii = ord(car)
    if 97 <= ascii <= 97 + 26 :
        return chr(ascii - 32)
    return car
```

```python
def car_en_maj(car) :
    ascii = ord(car)
    if 97 <= ascii <= 97 + 26 :
        return chr(ascii - 32)
    else :
        return car
```

```python
def car_en_maj(car) :
    ascii = ord(car)
    if 97 <= ascii <= 97 + 26 :
        return chr(ascii - 32)
    return car
```

Print

Et pourquoi pas **`print`** ?
`ord` prend un argument de type **`string`**, retourne un argument de type **`int`** ET NE FAIT QUE ÇA.
C'est une fonction PURE

Print

Et pourquoi pas **print** ?
ord prend un argument de type **string**, retourne un argument de type **int** ET NE FAIT QUE ÇA.
C'est une fonction PURE

Print

Et pourquoi pas **print** ?
ord prend un argument de type **string**, retourne un argument de type **int** ET NE FAIT QUE ÇA.
C'est une fonction PURE

Print

Et pourquoi pas `print` ?
`ord` prend un argument de type `string`, retourne un argument de type `int` ET NE FAIT QUE ÇA.
C'est une fonction PURE


8
VALID
MORROW, JEROME
011010100-09564
GNQ9.8612 = VALIDITY JE7342DAN

Print

```
In [34]: type(print(2))
2
Out[34]: NoneType

In [35]: print(print(2))
2
None

In [36]: print(print(2),print('GNU'))
2
GNU
None None
```

```
In [37]: [ print(a) for a in range(5) ]
0
1
2
3
4
Out[37]: [None, None, None, None, None]
```

```
In [38]: b = [ print(a) for a in range(5) ]
0
1
2
3
4

In [39]: b
Out[39]: [None, None, None, None, None]
```

`print` est IMPURE !

```
In [38]: b = [ print(a) for a in range(5) ]
0
1
2
3
4

In [39]: b
Out[39]: [None, None, None, None, None]
```

`print` est IMPURE !


IN-VALID
010011001-28253
TACATGACTAAGTTAC MYOPIA: TACCTGTCATT
GQ 3.4071 = DEFICIENCY LI
*SUSP. DE-GENE-ERATE

`print` renvoie toujours `None` mais a un EFFET SECONDAIRE (*side effect*) qui affiche quelque chose dans le monde extérieur.

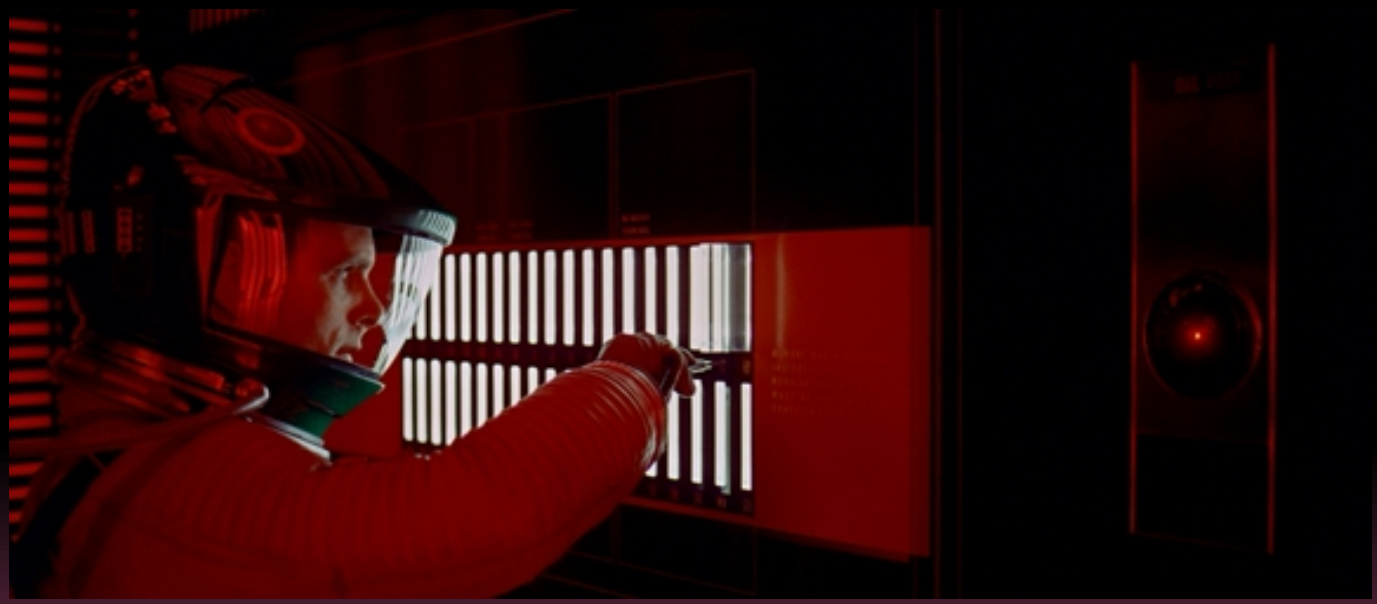
```python
def plus_chaine(x) :
    global a
    return x + ' ' + a
```

```
In [102]: a = 'Dave'

In [103]: plus_chaine('Bonjour')
Out[103]: 'Bonjour Dave'

In [104]: a = 'HAL'

In [105]: plus_chaine('Bonjour')
Out[105]: 'Bonjour HAL'
```

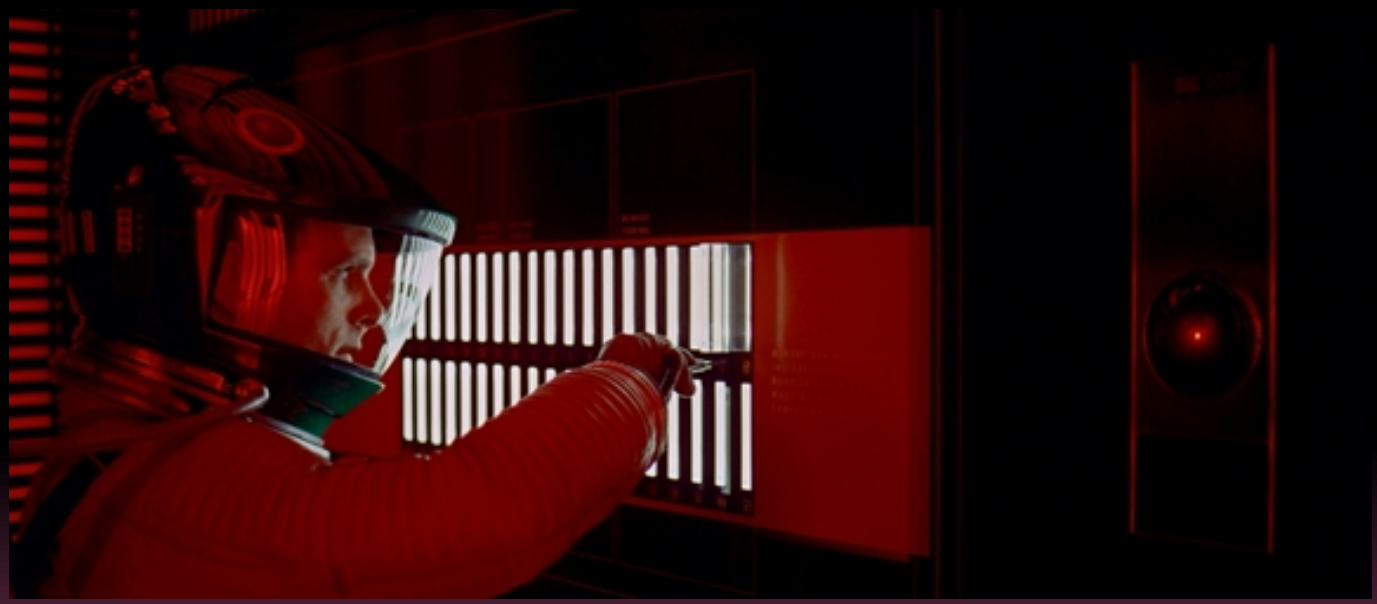


```
In [102]: a = 'Dave'

In [103]: plus_chaine('Bonjour')
Out[103]: 'Bonjour Dave'

In [104]: a = 'HAL'

In [105]: plus_chaine('Bonjour')
Out[105]: 'Bonjour HAL'
```




IN-VALID
010011001-28253
TACATGACTAAGTTAC MYOPIA: TACCTGTCATT
GQ 3.4071 = DEFICIENCY LI
*SUSP. DE-GENE-ERATE

http://golang.org

Rust

mozilla

Les fonctions sont des expressions comme les autres.

```
def transforme_une_lettre(transformation, lettre) :
    return transformation(lettre)
```

```
In [119]: transforme_une_lettre(car_en_maj, 'g')
Out[119]: 'G'
```

```
def cesar(lettre) :
    return chr( ord(lettre) + 3 )
```

```
In [120]: transforme_une_lettre(cesar, 'g')
Out[120]: 'j'
```

```
def transforme_une_lettre(transformation, lettre) :
    return transformation(lettre)
```

```
In [119]: transforme_une_lettre(car_en_maj, 'g')
Out[119]: 'G'
```

```
def cesar(lettre) :
    return chr( ord(lettre) + 3 )
```

```
In [120]: transforme_une_lettre(cesar, 'g')
Out[120]: 'j'
```

```
def transforme_une_lettre(transformation, lettre) :
    return transformation(lettre)
```

```
In [119]: transforme_une_lettre(car_en_maj, 'g')
Out[119]: 'G'
```

```
def cesar(lettre) :
    return chr( ord(lettre) + 3 )
```

```
In [120]: transforme_une_lettre(cesar, 'g')
Out[120]: 'j'
```

```
def transforme_une_lettre(transformation, lettre) :
    return transformation(lettre)
```

```
In [119]: transforme_une_lettre(car_en_maj, 'g')
Out[119]: 'G'
```

```
def cesar(lettre) :
    return chr( ord(lettre) + 3 )
```

```
In [120]: transforme_une_lettre(cesar, 'g')
Out[120]: 'j'
```

λ-expressions

$$x \mapsto 2x + 1$$

$$\lambda x \cdot 2x + 1$$

```
lambda x : 2*x + 1
```

```
In [121]: (lambda x : 2*x + 1)(5)
Out[121]: 11
```

$$x \mapsto 2x + 1$$

$$\lambda x \cdot 2x + 1$$

```
lambda x : 2*x + 1
```

```
In [121]: (lambda x : 2*x + 1)(5)
Out[121]: 11
```

$$x \mapsto 2x + 1$$

$$\lambda x \cdot 2x + 1$$

```
lambda x : 2*x + 1
```

```
In [121]: (lambda x : 2*x + 1)(5)
Out[121]: 11
```

$$x \mapsto 2x + 1$$

$$\lambda x \cdot 2x + 1$$

```
lambda x : 2*x + 1
```

```
In [121]: (lambda x : 2*x + 1)(5)
Out[121]: 11
```

```
In [1]: transforme_une_lettre(lambda c : chr( ord(c) + 5 ), 'a')
Out[1]: 'f'
```

```
def cesar(decalage) :
    return lambda lettre : chr( ord(lettre) + decalage )
```

```
In [127]: cesar(3)
Out[127]: <function __main__.cesar.<locals>.<lambda>>
```

```
In [128]: cesar(3)('a')
Out[128]: 'd'

In [129]: cesar(5)('a')
Out[129]: 'f'
```

```
def cesar(decalage) :
    return lambda lettre : chr( ord(lettre) + decalage )
```

```
In [127]: cesar(3)
Out[127]: <function __main__.cesar.<locals>.<lambda>>
```

```
In [128]: cesar(3)('a')
Out[128]: 'd'

In [129]: cesar(5)('a')
Out[129]: 'f'
```

```
def cesar(decalage) :
    return lambda lettre : chr( ord(lettre) + decalage )
```

```
In [127]: cesar(3)
Out[127]: <function __main__.cesar.<locals>.<lambda>>
```

```
In [128]: cesar(3)('a')
Out[128]: 'd'

In [129]: cesar(5)('a')
Out[129]: 'f'
```

Composition de fonctions

Défi ISI n° 1 :
On attendra dimanche 27 septembre ;-)

Dans un vrai langage de programmation :

```haskell
(.) :: (b -> c) -> (a -> b) -> a -> c
f . g = \x -> f (g x)
```

Défi

Écrire une fonction qui renvoie la longueur du mot le plus long d'une chaîne de caractères donnée en argument.

```
In [1]: long_mot_plus_long("Vos beaux yeux d'amour mourir me font Belle
          ↪  Marquise")
Out[1]: 8
```