

Compléments d'informatique

L1 MIASHS

Guillaume CONNAN

Université Catholique de l'Ouest - Angers

Novembre-Décembre 2018

Sommaire

Descriptif

Modélisation

Vision statique : diagramme de Classes

Python

Juste assez de POO pour traduire notre

UML en actes

Diagramme de cas d'utilisation

Diagramme d'activité

Diagramme de séquence

GIT

Dictionnaires

Découverte

Travail à faire

Lecture / Écriture de fichiers

Sommaire

Descriptif

Modélisation

Vision statique : diagramme de Classes

Python

Juste assez de POO pour traduire notre

UML en actes

Diagramme de cas d'utilisation

Diagramme d'activité

Diagramme de séquence

GIT

Dictionnaires

Découverte

Travail à faire

Lecture / Écriture de fichiers

1. Modélisation d'une application

1. Modélisation d'une application
2. Découverte et utilisation des objets

1. Modélisation d'une application
2. Découverte et utilisation des objets
3. Documentation d'une application

1. Modélisation d'une application
2. Découverte et utilisation des objets
3. Documentation d'une application
4. Compléments d'algorithmique

1. Modélisation d'une application
2. Découverte et utilisation des objets
3. Documentation d'une application
4. Compléments d'algorithmique
5. Réalisation d'un projet en binôme

Sommaire

Descriptif

Modélisation

Vision statique : diagramme de Classes

Python

Juste assez de POO pour traduire notre

UML en actes

Diagramme de cas d'utilisation

Diagramme d'activité

Diagramme de séquence

GIT

Dictionnaires

Découverte

Travail à faire

Lecture / Écriture de fichiers

*Avant donc que d'écrire, apprenez à penser.
Selon que notre idée est plus ou moins obscure,
L'expression la suit, ou moins nette, ou plus pure.
Ce que l'on conçoit bien s'énonce clairement.*

...

*Hâtez-vous lentement, et, sans perdre courage,
Vingt fois sur le métier remettez votre ouvrage
Polissez-le sans cesse et le repolissez;
Ajoutez quelquefois, et souvent effacez.*

*Avant donc que d'écrire, apprenez à penser.
Selon que notre idée est plus ou moins obscure,
L'expression la suit, ou moins nette, ou plus pure.
Ce que l'on conçoit bien s'énonce clairement.*

...

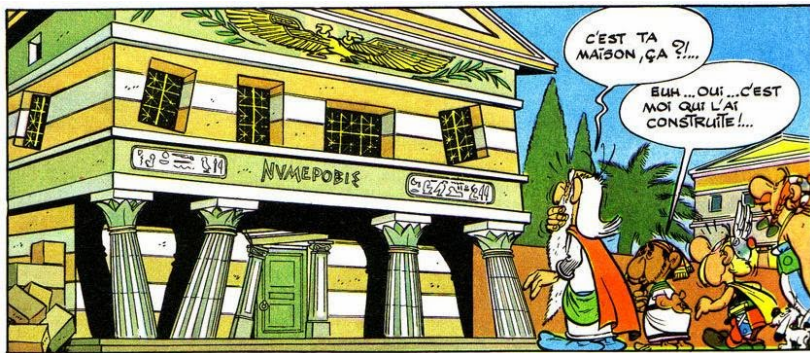
*Hâtez-vous lentement, et, sans perdre courage,
Vingt fois sur le métier remettez votre ouvrage
Polissez-le sans cesse et le repolissez;
Ajoutez quelquefois, et souvent effacez.*

*Avant donc que d'écrire, apprenez à penser.
Selon que notre idée est plus ou moins obscure,
L'expression la suit, ou moins nette, ou plus pure.
Ce que l'on conçoit bien s'énonce clairement.*

...

*Hâtez-vous lentement, et, sans perdre courage,
Vingt fois sur le métier remettez votre ouvrage
Polissez-le sans cesse et le repolissez;
Ajoutez quelquefois, et souvent effacez.*

Nicolas BOILEAU in *L'Art poétique* – 1674



Un cas simple d'étude

Le service d'état-civil de la mairie d'Angers demande un outil pour enregistrer des personnes et pour mettre à jour des changements dans leur situation (mariage, décès, etc.)

Un cas simple d'étude

Le service d'état-civil de la mairie d'Angers demande un outil pour enregistrer des personnes et pour mettre à jour des changements dans leur situation (mariage, décès, etc.)

Nous allons nous restreindre dans un premier temps au cas de l'enregistrement d'une personne et d'un éventuel mariage.

- ▶ UML est le langage de modélisation orienté objet le plus connu et le plus utilisé ;

- ▶ UML est le langage de modélisation orienté objet le plus connu et le plus utilisé ;
- ▶ Objectif : cadre unifié dans un contexte industriel ;

- ▶ UML est le langage de modélisation orienté objet le plus connu et le plus utilisé ;
- ▶ Objectif : cadre unifié dans un contexte industriel ;
- ▶ Succès : simplicité ! structures intuitives ;

- ▶ UML est le langage de modélisation orienté objet le plus connu et le plus utilisé ;
- ▶ Objectif : cadre unifié dans un contexte industriel ;
- ▶ Succès : simplicité ! structures intuitives ;
- ▶ UML 2.0 est apparu en 2002.

UML : un problème \longrightarrow un **diagramme**

- ▶ diagramme de classe

UML : un problème \longrightarrow un **diagramme**

- ▶ diagramme de classe
- ▶ diagramme de cas d'utilisation

UML : un problème \longrightarrow un **diagramme**

- ▶ diagramme de classe
- ▶ diagramme de cas d'utilisation
- ▶ diagramme de déploiement

UML : un problème \longrightarrow un **diagramme**

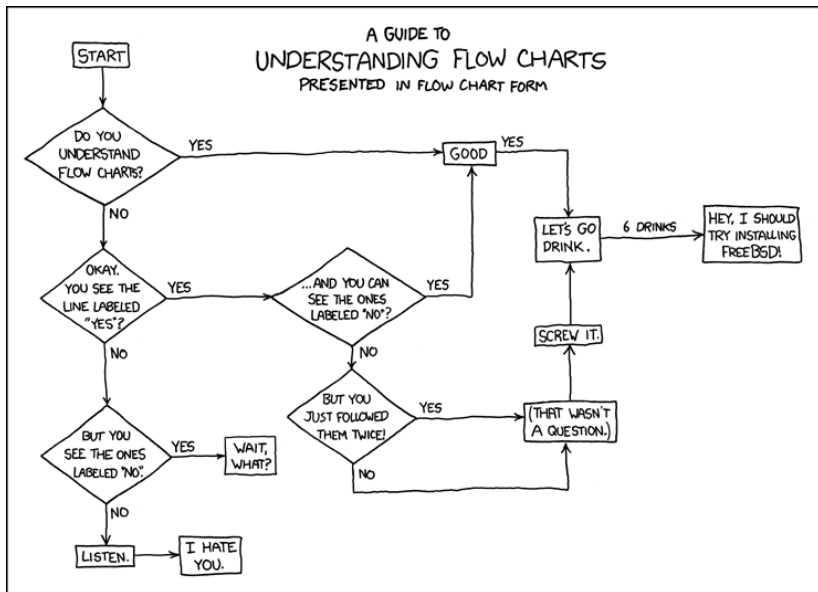
- ▶ diagramme de classe
- ▶ diagramme de cas d'utilisation
- ▶ diagramme de déploiement
- ▶ diagramme de séquence

UML : un problème \longrightarrow un **diagramme**

- ▶ diagramme de classe
- ▶ diagramme de cas d'utilisation
- ▶ diagramme de déploiement
- ▶ diagramme de séquence
- ▶ diagramme d'activité

UML : un problème \longrightarrow un **diagramme**

- ▶ diagramme de classe
- ▶ diagramme de cas d'utilisation
- ▶ diagramme de déploiement
- ▶ diagramme de séquence
- ▶ diagramme d'activité
- ▶ ... une vingtaine en tout !



1. Réfléchir

1. Réfléchir
2. Définir la structure « gros grain »

1. Réfléchir
2. Définir la structure « gros grain »
3. Documenter

1. Réfléchir
2. Définir la structure « gros grain »
3. Documenter
4. Guider le développement

1. Réfléchir
2. Définir la structure « gros grain »
3. Documenter
4. Guider le développement
5. Tester

MAIS

- ▶ les logiciels sont de plus en plus complexes et ont besoin d'être sécurisés ;

MAIS

- ▶ les logiciels sont de plus en plus complexes et ont besoin d'être sécurisés ;
- ▶ il faut de la rigueur et de la précision ;

MAIS

- ▶ les logiciels sont de plus en plus complexes et ont besoin d'être sécurisés ;
- ▶ il faut de la rigueur et de la précision ;
- ▶ techniques fiables à base mathématique ;

MAIS

- ▶ les logiciels sont de plus en plus complexes et ont besoin d'être sécurisés ;
- ▶ il faut de la rigueur et de la précision ;
- ▶ techniques fiables à base mathématique ;
- ▶ **Méthodes formelles** : méthode B, Esterel, Coq,...

	Méthodes semi-formelles	Méthodes formelles
Formalisme	Textuel ou graphique (Merise, SADT, UML, ...)	Mathématique (Z, VDM, B, ...)
Syntaxe du langage	Précise	Précise
Sémantique du langage	Assez faible	Précise
Validation	Syntaxique + Expertise humaine	Preuves Démonstration de théorèmes Model checking Animation et test
Outils	Ateliers de Génie Logiciel (AGL)	Prouveurs + Animateurs
Domaine applicatif	Se veulent généralistes	Systèmes sûrs ou critiques
Objectif	Systèmes bien structurés	Systèmes fiables

Sommaire

Descriptif

Modélisation

Vision statique : diagramme de Classes

Python

Juste assez de POO pour traduire notre

UML en actes

Diagramme de cas d'utilisation

Diagramme d'activité

Diagramme de séquence

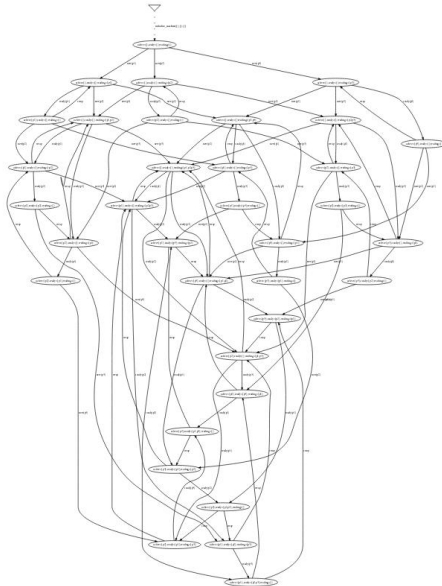
GIT

Dictionnaires

Découverte

Travail à faire

Lecture / Écriture de fichiers



Classes

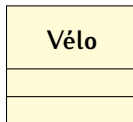
- ▶ Une diagramme de classes donne une vue graphique de la structure statique d'un système ;

Classes

- ▶ Une diagramme de classes donne une vue graphique de la structure statique d'un système ;
- ▶ Une classe représente la structure commune d'un ensemble d'objets : personne, vélo, roue, cadre...

Classes

- ▶ Une diagramme de classes donne une vue graphique de la structure statique d'un système ;
- ▶ Une classe représente la structure commune d'un ensemble d'objets : personne, vélo, roue, cadre...
- ▶ Une classe est représentée par un rectangle, son nom commence par une majuscule et le rectangle peut contenir trois parties



Attributs

- Un attribut est une caractéristique, une information contenue dans une classe ;

Attributs

- ▶ Un attribut est une caractéristique, une information contenue dans une classe ;
- ▶ la taille, la marque, le modèle, l'année de fabrication sont des attributs de vélo

Attributs

- ▶ Un attribut est une caractéristique, une information contenue dans une classe ;
- ▶ la taille, la marque, le modèle, l'année de fabrication sont des attributs de vélo
- ▶ Moins bricolo : un attribut est une relation binaire d'un ensemble d'objets vers un ensemble de valeurs :

Attributs

- ▶ Un attribut est une caractéristique, une information contenue dans une classe ;
- ▶ la taille, la marque, le modèle, l'année de fabrication sont des attributs de vélo
- ▶ Moins bricolo : un attribut est une relation binaire d'un ensemble d'objets vers un ensemble de valeurs :

Attributs

- ▶ Un attribut est une caractéristique, une information contenue dans une classe ;
- ▶ la taille, la marque, le modèle, l'année de fabrication sont des attributs de vélo
- ▶ Moins bricolo : un attribut est une relation binaire d'un ensemble d'objets vers un ensemble de valeurs :

Voiture numéro 3

Attributs

- ▶ Un attribut est une caractéristique, une information contenue dans une classe ;
- ▶ la taille, la marque, le modèle, l'année de fabrication sont des attributs de vélo
- ▶ Moins bricolo : un attribut est une relation binaire d'un ensemble d'objets vers un ensemble de valeurs :

Voiture numéro 3 (*Objet*)

Attributs

- ▶ Un attribut est une caractéristique, une information contenue dans une classe ;
- ▶ la taille, la marque, le modèle, l'année de fabrication sont des attributs de vélo
- ▶ Moins bricolo : un attribut est une relation binaire d'un ensemble d'objets vers un ensemble de valeurs :

Voiture numéro 3 (*Objet*) a_pour_couleur

Attributs

- ▶ Un attribut est une caractéristique, une information contenue dans une classe ;
- ▶ la taille, la marque, le modèle, l'année de fabrication sont des attributs de vélo
- ▶ Moins bricolo : un attribut est une relation binaire d'un ensemble d'objets vers un ensemble de valeurs :

Voiture numéro 3 (*Objet*) a_pour_couleur (*Attribut*)

Attributs

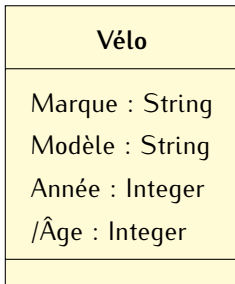
- ▶ Un attribut est une caractéristique, une information contenue dans une classe ;
- ▶ la taille, la marque, le modèle, l'année de fabrication sont des attributs de vélo
- ▶ Moins bricolo : un attribut est une relation binaire d'un ensemble d'objets vers un ensemble de valeurs :

Voiture numéro 3 (*Objet*) a_pour_couleur (*Attribut*) Jaune

Attributs

- ▶ Un attribut est une caractéristique, une information contenue dans une classe ;
- ▶ la taille, la marque, le modèle, l'année de fabrication sont des attributs de vélo
- ▶ Moins bricolo : un attribut est une relation binaire d'un ensemble d'objets vers un ensemble de valeurs :

Voiture numéro 3 (*Objet*) a_pour_couleur (*Attribut*) Jaune (*Valeur*)

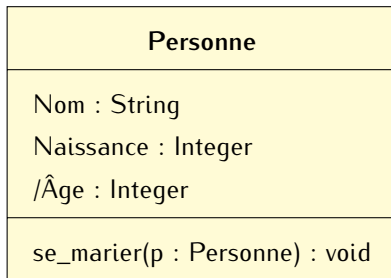


Opérations

- Une opération est un service qu'un objet (une *instance* de la classe) peut exécuter ;

Opérations

- ▶ Une opération est un service qu'un objet (une *instance* de la classe) peut exécuter ;
- ▶ fonctionner, en_réparation

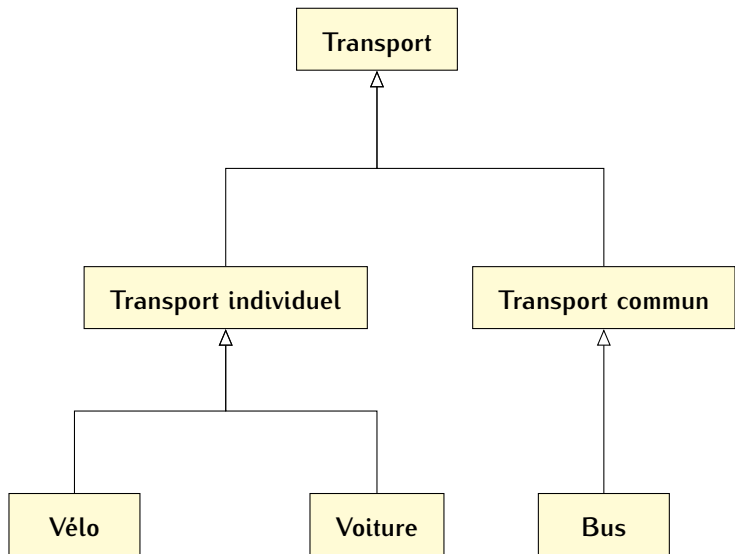


Héritage

- ▶ c'est une relation entre une classe et une classe plus générale ;

Héritage

- ▶ c'est une relation entre une classe et une classe plus générale ;
- ▶ un vélo est un moyen de transport individuel, la voiture aussi, le tram est un moyen transport en commun, un moyen de transport en commun est un moyen de transport...



Associations

- une association sera de manière privilégiée binaire : elle connecte deux éléments (classes le plus souvent)

Associations

- ▶ une association sera de manière privilégiée binaire : elle connecte deux éléments (classes le plus souvent)
- ▶ une associations binaire a deux *association ends*

Associations

- ▶ une association sera de manière privilégiée binaire : elle connecte deux éléments (classes le plus souvent)
- ▶ une associations binaire a deux *association ends*
- ▶ une association est paramétrée par au moins un des éléments suivants :

Associations

- ▶ une association sera de manière privilégiée binaire : elle connecte deux éléments (classes le plus souvent)
- ▶ une associations binaire a deux *association ends*
- ▶ une association est paramétrée par au moins un des éléments suivants :
 - ▶ un nom (minuscule) décrivant le rôle joué par une entité par rapport à l'autre. Si l'association est bidirectionnelle, un verbe à l'actif dans un sens est lu au passif dans l'autre sens ;

Associations

- ▶ une association sera de manière privilégiée binaire : elle connecte deux éléments (classes le plus souvent)
- ▶ une associations binaire a deux *association ends*
- ▶ une association est paramétrée par au moins un des éléments suivants :
 - ▶ un nom (minuscule) décrivant le rôle joué par une entité par rapport à l'autre. Si l'association est bidirectionnelle, un verbe à l'actif dans un sens est lu au passif dans l'autre sens ;
 - ▶ Une multiplicité (0,1, *, 1..*, 0..2, etc.)

Associations

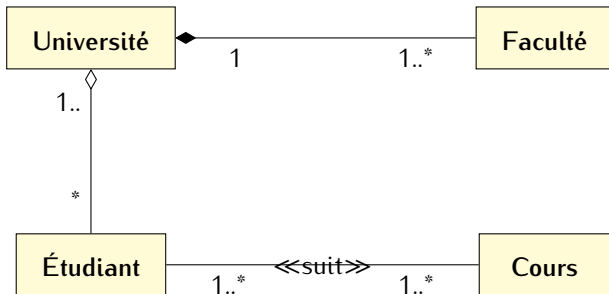
- ▶ une association sera de manière privilégiée binaire : elle connecte deux éléments (classes le plus souvent)
- ▶ une associations binaire a deux *association ends*
- ▶ une association est paramétrée par au moins un des éléments suivants :
 - ▶ un nom (minuscule) décrivant le rôle joué par une entité par rapport à l'autre. Si l'association est bidirectionnelle, un verbe à l'actif dans un sens est lu au passif dans l'autre sens ;
 - ▶ Une multiplicité (0,1, *, 1..*, 0..2, etc.)
 - ▶ un genre d'agrégation dans le cas où l'association n'est pas symétrique :

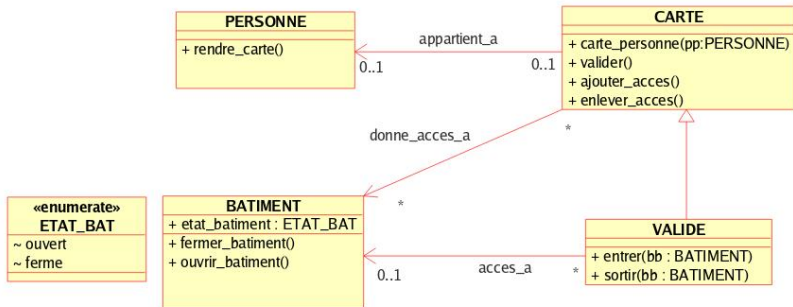
Associations

- ▶ une association sera de manière privilégiée binaire : elle connecte deux éléments (classes le plus souvent)
- ▶ une associations binaire a deux *association ends*
- ▶ une association est paramétrée par au moins un des éléments suivants :
 - ▶ un nom (minuscule) décrivant le rôle joué par une entité par rapport à l'autre. Si l'association est bidirectionnelle, un verbe à l'actif dans un sens est lu au passif dans l'autre sens ;
 - ▶ Une multiplicité (0,1, *, 1..*, 0..2, etc.)
 - ▶ un genre d'agrégation dans le cas où l'association n'est pas symétrique :
 - ▶ *composite*(◆) : la destruction d'une entité entraîne la destruction de l'autre. Un élément ne peut appartenir qu'à un seul agrégat *composite* (chat / pattes)

Associations

- ▶ une association sera de manière privilégiée binaire : elle connecte deux éléments (classes le plus souvent)
- ▶ une associations binaire a deux *association ends*
- ▶ une association est paramétrée par au moins un des éléments suivants :
 - ▶ un nom (minuscule) décrivant le rôle joué par une entité par rapport à l'autre. Si l'association est bidirectionnelle, un verbe à l'actif dans un sens est lu au passif dans l'autre sens ;
 - ▶ Une multiplicité (0,1, *, 1..*, 0..2, etc.)
 - ▶ un genre d'agrégation dans le cas où l'association n'est pas symétrique :
 - ▶ *composite*(◆) : la destruction d'une entité entraîne la destruction de l'autre. Un élément ne peut appartenir qu'à un seul agrégat *composite* (chat / pattes)
 - ▶ simple (◇) (voiture/roue)





Contraintes et notes

- Pour arriver à un semblant de sérieux par rapport aux méthodes formelles, on doit compléter le diagramme de classe par des indications de contraintes.

Contraintes et notes

- ▶ Pour arriver à un semblant de sérieux par rapport aux méthodes formelles, on doit compléter le diagramme de classe par des indications de contraintes.
- ▶ Trop souvent, ces contraintes sont écrites en langage naturel.

Contraintes et notes

- ▶ Pour arriver à un semblant de sérieux par rapport aux méthodes formelles, on doit compléter le diagramme de classe par des indications de contraintes.
- ▶ Trop souvent, ces contraintes sont écrites en langage naturel.
- ▶ Il faut utiliser le langage OCL (*Object Constraint Language*) pour éviter le bricolage.

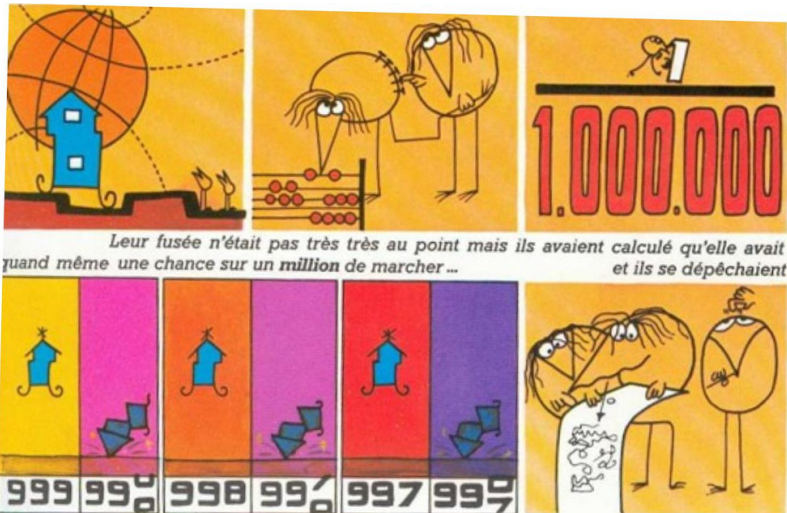
1842 - Ada Lovelace writes the first program. She is hampered in her efforts by the minor inconvenience that she doesn't have any actual computers to run her code. Enterprise architects will later relearn her techniques in order to program in UML.

- ▶ Vous sautez de l'avion ;

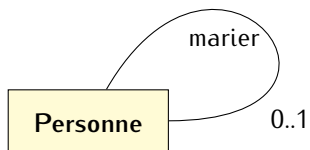
- ▶ Vous sautez de l'avion ;
- ▶ Puis vous lisez la contrainte : « avant de sauter s'assurer d'avoir enfilé un parachute opérationnel »

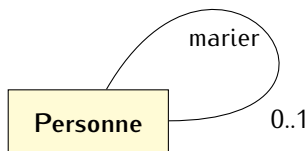
- ▶ Vous sautez de l'avion ;
- ▶ Puis vous lisez la contrainte : « avant de sauter s'assurer d'avoir enfilé un parachute opérationnel »
- ▶ Trop tard...

- ▶ Vous sautez de l'avion ;
- ▶ Puis vous lisez la contrainte : « avant de sauter s'assurer d'avoir enfilé un parachute opérationnel »
- ▶ Trop tard...
- ▶ en passant : interprété/compilé

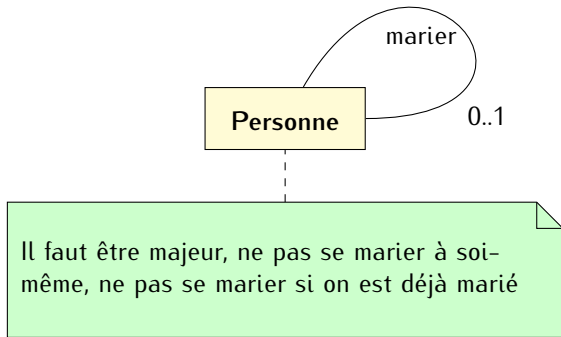


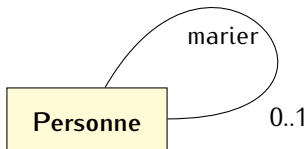
L'administration aura désormais deux mois pour répondre au courrier des usagers : les fonctionnaires ont choisi juin et novembre.





Quels problèmes ?





```
context Personne::marier(p :Personne)
pre pas_marier_soi-meme: not(p = self)
- - aucune des personnes ne doit avoir été mariée
pre pas_deja_marie : self.spouse->size() = 0 and p.spouse->size()=0
pre majeur : self.age >= 18 and p.age >= 18

post : self.spouse = p and p.spouse = self
```

Sommaire

Descriptif

Modélisation

Vision statique : diagramme de Classes

Python

Juste assez de POO pour traduire notre

UML en actes

Diagramme de cas d'utilisation

Diagramme d'activité

Diagramme de séquence

GIT

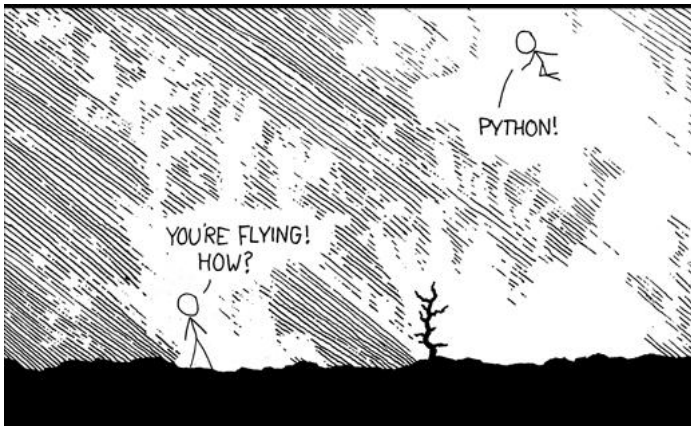
Dictionnaires

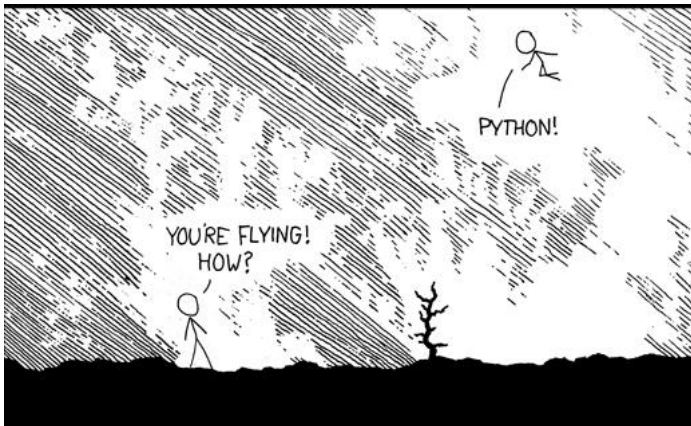
Découverte

Travail à faire

Lecture / Écriture de fichiers

1991 - Dutch programmer Guido van Rossum travels to Argentina for a mysterious operation. He returns with a large cranial scar, invents Python, is declared Dictator for Life by legions of followers, and announces to the world that "There Is Only One Way to Do It." Poland becomes nervous.





I JUST TYPED
`import antigravity`

Sommaire

Descriptif

Modélisation

Vision statique : diagramme de Classes

Python

**Juste assez de POO pour traduire notre
UML en actes**

Diagramme de cas d'utilisation

Diagramme d'activité

Diagramme de séquence

GIT

Dictionnaires

Découverte

Travail à faire

Lecture / Écriture de fichiers

Vous vous êtes sûrement demandés pourquoi les appels de « fonctions » n'ont pas toujours la même syntaxe...

```
Moi[1]: xs = [1,2]
Moi[2]: append(xs, 3)
Traceback (most recent call last):

  File "<ipython-input-2-c65410c1a3c8>", line 1, in <module>
    append(xs, 3)

NameError: name 'append' is not defined
```

```
Moi[3]: xs.append(3)
```

```
Moi[4]: xs
```

```
Python[4]: [1, 2, 3]
```

```
Moi[6]: xs.len()
Traceback (most recent call last):

  File "<ipython-input-6-bedaf6ec6921>", line 1, in <module>
    xs.len()

AttributeError: 'list' object has no attribute 'len'
```

```
Moi[7]: len(xs)  
Python[7]: 3
```

Même les nombres ont ce genre de « fonctions » bizarres qui suivent un point :

Même les nombres ont ce genre de « fonctions » bizarres qui suivent un point :

```
Moi[10]: 1.2.is_integer()  
Python[10]: False  
  
Moi[11]: 1.0.is_integer()  
Python[11]: True
```

Une *classe* correspond à un « moule » permettant de créer un type d'objet.

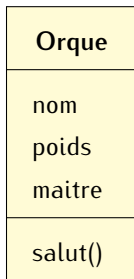
Il ne faut pas confondre la classe avec l'objet (il vaut mieux manger le gâteau que le moule).

Prenons un exemple : vous êtes Saroumane et vous avez un moule à orques. Vous pouvez créer ainsi une infinité d'orques. Informatiquement la classe `Orque` permet d'*instancier* un objet de type orque.

Prenons un exemple : vous êtes Saroumane et vous avez un moule à orques. Vous pouvez créer ainsi une infinité d'orques.

Informatiquement la classe `Orque` permet d'*instancier* un objet de type orque.

Un orque a un nom et un poids : ce sont des *attributs* (ses caractéristiques). Il peut effectuer un salut : c'est une *méthode* (ce que l'objet peut faire).



```
class Orque:
    nom = ""
    poids = 0
    maitre = ""

    def salut(self):
        print("Ash nazg durbatulûk! Mon nom est %s" % self.nom)
```

```
In [76]: a = Orque()
In [77]: a.nom = "Grishnákh"
In [78]: a.salut()
Ash nazg durbatulúk: Mon nom est Grishnákh
In [79]: a.age = 2
In [80]: b = Orque()
In [81]: b.nom = "Uglúk"
In [82]: b.salut()
Ash nazg durbatulúk: Mon nom est Uglúk
In [83]: a.maitre = "Saroumane"
In [84]: b.maitre
Out[84]: 'Sauron'
```



```
class Orque:

    def __init__(self):
        self.nom      = ""
        self.poids    = 0
        self.maitre    = "Sauron"

    def salut(self):
        print("Ash nazg durbatulûk! Mon nom est %s" % self.nom)
```

```
class Orque:

    def __init__(self, nom, poids, maitre = "Sauron"):
        self.nom = nom
        self.poids = poids
        self.maitre = maitre

    def salut(self):
        print("Ash nazg durbatulûk! Mon nom est %s" % self.nom)
```

```
In [106]: a = Orque("Truc",3)

In [107]: a.maitre
Out[107]: 'Sauron'

In [108]: a = Orque("Truc","Sauron")

In [109]: a.maitre
Out[109]: 'Sauron'

In [110]: a.poids
Out[110]: 'Sauron'

In [111]: a = Orque("Truc",100,"Sauron")

In [112]: a.poids
Out[112]: 100

In [113]: a.maitre
Out[113]: 'Sauron'

In [114]: a.nom
Out[114]: 'Truc'
```



```
class Orque:

    def __init__(self, nom:str, poids:int, maitre:str = "Sauron") -> None:
        self.nom      = nom
        self.poids     = poids
        self.maitre    = maitre

    def salut(self)->None:
        print("Ash nazg durbatulûk! Mon nom est %s" % self.nom)
```

```
Moi[22]: a = Orque("Truc", "Sauron")
```

```
Moi[23]: a.poids
```

```
Python[23]: 'Sauron'
```

```
class Orque:

    def __init__(self, nom:str, poids:int, maitre:str = "Sauron") -> None :
        assert type(poids) == int, "le poids est un entier"
        self.nom      = nom
        self.poids     = poids
        self.maitre    = maitre

    def salut(self)->None:
        print("Ash nazg durbatulûk! Mon nom est %s" % self.nom)
```

```
Moi[25]: a = Orque("Truc", "Sauron")
Traceback (most recent call last):

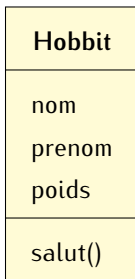
  File "<ipython-input-25-4b1feddb2229>", line 1, in <module>
    a = Orque("Truc", "Sauron")

  File "/home/moi/UCO/L1/ANGERS2018/Python/Orques.py", line 12, in __init__
    assert type(poids) == int, "le poids est un entier"

AssertionError: le poids est un entier
```

Imaginez maintenant que vous êtes JRR TOLKIEN et que vous voulez fabriquer des Hobbits. Vous vous dites qu'il suffit de créer une classe `Hobbit` :

Imaginez maintenant que vous êtes JRR TOLKIEN et que vous voulez fabriquer des Hobbits. Vous vous dites qu'il suffit de créer une classe `Hobbit` :



```
class Hobbit:

    def __init__(self, nom:str, prenom:str, poids:int) -> None:
        assert type(poids) == int, "le poids est un entier"
        assert type(nom) == str and type(prenom) == str, "Le nom et le prénom sont des String"
        self.nom      = nom
        self.prenom    = prenom
        self.poids     = poids

    def salut(self) -> None:
        print("Bonjour! Mon nom est %s %s" % (self.prenom, self.nom))
```

```
Moi[27]: a = Hobbit("Sacquet", "Bilbo", 50)
```

```
Moi[28]: a.salut()
```

```
Bonjour! Mon nom est Bilbo Sacquet
```


Et ensuite il y a les Elfes, les Nains, etc.

On remarque cependant que les Orques comme les Hobbits ont un nom et un poids.

On peut alors créer une « super-classe » Etre dont Hobbit et Orque serait des sous-classes qui *héritent* de ses attributs...comme en UML.

```
class Etre(object):

    def __init__(self, appellation:str, poids:int, bonjour:str) -> None:
        assert type(poids) == int, "le poids est un entier"
        assert type(appellation) == str and type(bonjour) == str, "Le nom et la salutation sont
            des String"
        self.appellation = appellation
        self.poids       = poids
        self.bonjour      = bonjour

    def salut(self) -> None:
        print("%s ! Mon nom est %s" % (self.bonjour, self.appellation))

    def masse(self) -> None:
        print("Je pèse %f kg" % self.poids)

class Orque(Etre):

    def __init__(self, nom:str, poids:int, maitre = "Sauron"):
        Etre.__init__(self,nom,poids,"Ash nazg durbatulûk")
        self.maitre = maitre

class Hobbit(Etre):

    def __init__(self, nom:str, prenom:str, poids:int):
        Etre.__init__(self,prenom + ' ' + nom, poids, "Bonjour")
        self.prenom = prenom
```

```
Moi[31]: a = Hobbit("Sacquet", "Bilbo", 50)

Moi[32]: a.masse()
Je pèse 50.000000 kg

Moi[33]: a.salut()
Bonjour ! Mon nom est Bilbo Sacquet

Moi[34]: b = Orque("Uglúk",100)

Moi[35]: b.salut()
Ash nazg durbatulúk ! Mon nom est Uglúk

Moi[36]: c = Orque("Uglúk","Saroumane")
Traceback (most recent call last):

  File "<ipython-input-36-44938988904b>", line 1, in <module>
    c = Orque("Uglúk","Saroumane")

  File "/home/moi/UCO/L1/ANGERS2018/Python/Orques.py", line 29, in __init__
    Etre.__init__(self,nom,poids,"Ash nazg durbatulúk")

  File "/home/moi/UCO/L1/ANGERS2018/Python/Orques.py", line 13, in __init__
    assert type(poids) == int, "le poids est un entier"

AssertionError: le poids est un entier
```

Sommaire

Descriptif

Modélisation

Vision statique : diagramme de Classes

Python

Juste assez de POO pour traduire notre

UML en actes

Diagramme de cas d'utilisation

Diagramme d'activité

Diagramme de séquence

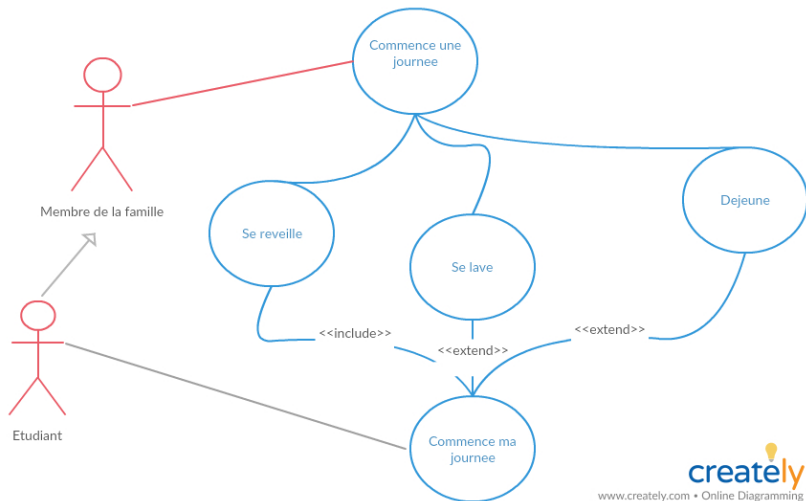
GIT

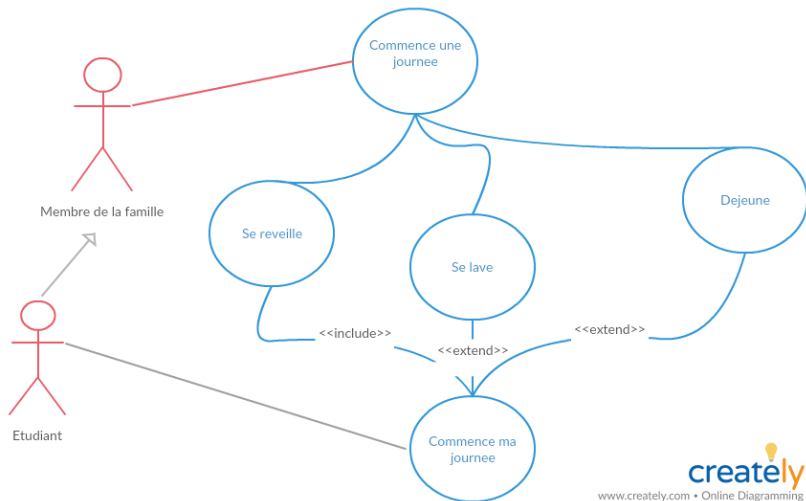
Dictionnaires

Découverte

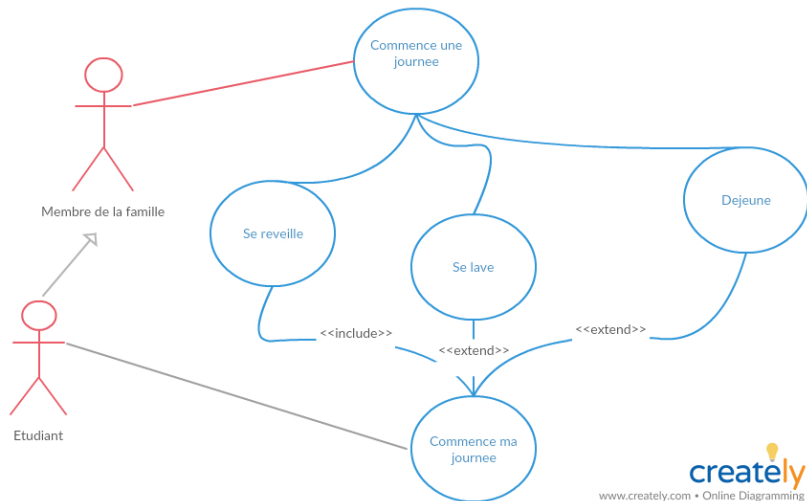
Travail à faire

Lecture / Écriture de fichiers

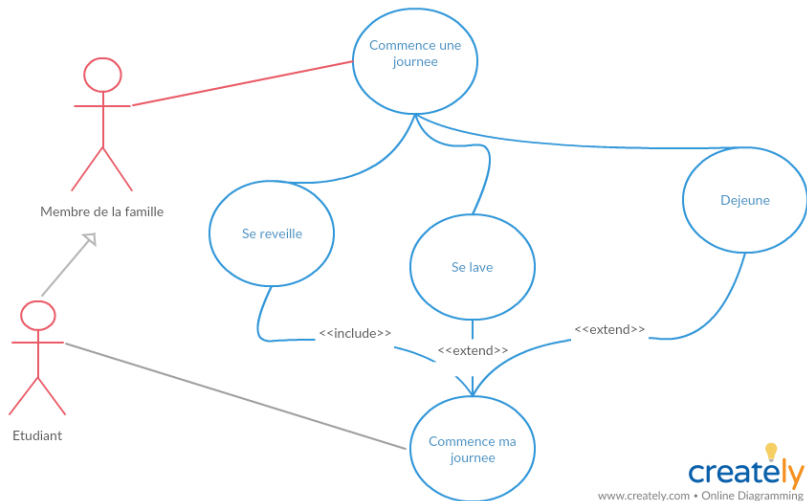




<<include>> : je fais toujours



<<include>> : je fais toujours <<extend>> : je choisis de faire



<<include>> : je fais toujours <<extend>> : je choisis de faire
comment savoir si je choisis de faire une action ?

Sommaire

Descriptif

Modélisation

Vision statique : diagramme de Classes

Python

Juste assez de POO pour traduire notre

UML en actes

Diagramme de cas d'utilisation

Diagramme d'activité

Diagramme de séquence

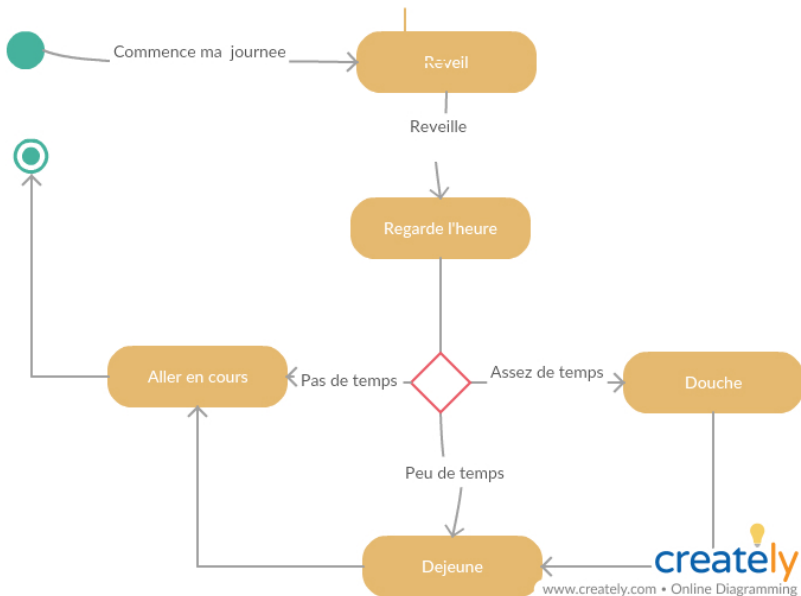
GIT

Dictionnaires

Découverte

Travail à faire

Lecture / Écriture de fichiers



Sommaire

Descriptif

Modélisation

Vision statique : diagramme de Classes

Python

Juste assez de POO pour traduire notre

UML en actes

Diagramme de cas d'utilisation

Diagramme d'activité

Diagramme de séquence

GIT

Dictionnaires

Découverte

Travail à faire

Lecture / Écriture de fichiers

Diagramme de séquence

- Représente une interaction entre des instances qui jouent des rôles.

Diagramme de séquence

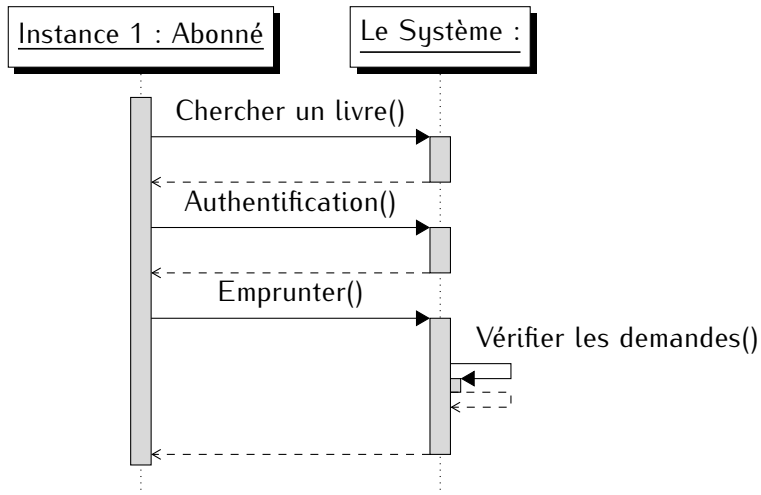
- ▶ Représente une interaction entre des instances qui jouent des rôles.
- ▶ Dynamique car permet de visualiser une durée de vie.

Diagramme de séquence

- ▶ Représente une interaction entre des instances qui jouent des rôles.
- ▶ Dynamique car permet de visualiser une durée de vie.
- ▶ L'interaction se fait par envoi de « messages » : création, destruction...

Diagramme de séquence

- ▶ Représente une interaction entre des instances qui jouent des rôles.
- ▶ Dynamique car permet de visualiser une durée de vie.
- ▶ L'interaction se fait par envoi de « messages » : création, destruction...
- ▶ Sémantique un peu floue...



Sommaire

Descriptif

Modélisation

Vision statique : diagramme de Classes

Python

Juste assez de POO pour traduire notre

UML en actes

Diagramme de cas d'utilisation

Diagramme d'activité

Diagramme de séquence

GIT

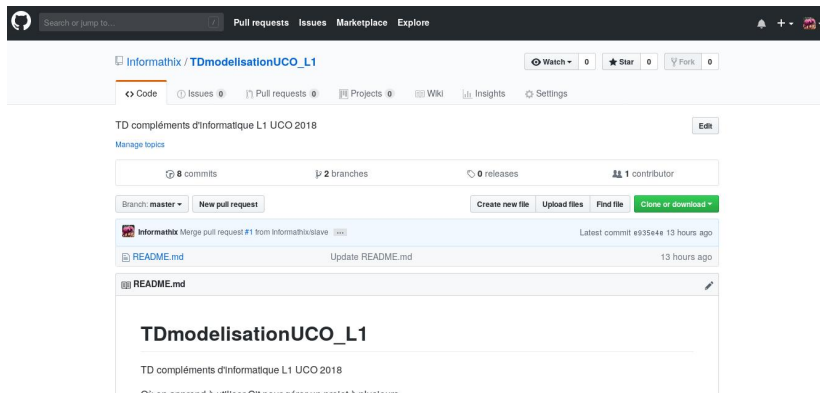
Dictionnaires

Découverte

Travail à faire

Lecture / Écriture de fichiers

Vous créez en TD un dépôt GIT pour travailler à deux sur un même projet.



The screenshot shows the GitHub interface for the repository 'Informathix / TDmodelisationUCO_L1'. The repository has 8 commits, 2 branches, 0 releases, and 1 contributor. The 'master' branch is selected. A pull request is visible, titled 'Merge pull request #1 from Informathix/slave', with the latest commit 'e935e4a' from 13 hours ago. The repository contains a 'README.md' file, which is displayed below. The README content includes the title 'TDmodelisationUCO_L1' and the subtitle 'TD compléments d'Informatique L1 UCO 2018'. Below the subtitle, there is a line of text that is partially obscured but appears to be 'On ne apprend à utiliser Git pour créer un projet à plusieurs'.

<https://github.com/>

Learn Git and GitHub without any code!

Using the Hello World guide, you'll create a repository, start a branch, write comments, and open a pull request.

[Read the guide](#)

[Start a project](#)

Sommaire

Descriptif

Modélisation

Vision statique : diagramme de Classes

Python

Juste assez de POO pour traduire notre

UML en actes

Diagramme de cas d'utilisation

Diagramme d'activité

Diagramme de séquence

GIT

Dictionnaires

Découverte

Travail à faire

Lecture / Écriture de fichiers

Sommaire

Descriptif

Modélisation

Vision statique : diagramme de Classes

Python

Juste assez de POO pour traduire notre UML en actes

Diagramme de cas d'utilisation

Diagramme d'activité

Diagramme de séquence

GIT

Dictionnaires

Découverte

Travail à faire

Lecture / Écriture de fichiers

Ensemble contenant des éléments auxquels on a accès à l'aide de clés choisies par le créateur...

Ensemble contenant des éléments auxquels on a accès à l'aide de clés choisies par le créateur...

```
In [45]: Tel = {}  
  
In [46]: Tel['Roger'] = '06 12 11 13 20'  
  
In [47]: Tel  
Out[47]: {'Roger': '06 12 11 13 20'}  
  
In [48]: Tel['Roger']  
Out[48]: '06 12 11 13 20'
```

```
In [49]: Tel['Josette'] = '07 00 00 01 00'

In [50]: Tel['Bill'] = '06 05 04 03 02'

In [51]: Tel
Out[51]:
{'Bill': '06 05 04 03 02',
 'Josette': '07 00 00 01 00',
 'Roger': '06 12 11 13 20'}

In [52]: les06 = {ami for ami in Tel if Tel[ami][:2] == '06'}

In [53]: les06
Out[53]: {'Bill', 'Roger'}
```



```
In [54]: del Tel['Josette']
```

```
In [55]: Tel
```

```
Out[55]: {'Bill': '06 05 04 03 02', 'Roger': '06 12 11 13 20'}
```

```
In [54]: del Tel['Josette']
```

```
In [55]: Tel
```

```
Out[55]: {'Bill': '06 05 04 03 02', 'Roger': '06 12 11 13 20'}
```

```
In [56]: Tel.keys()
```

```
Out[56]: dict_keys(['Bill', 'Roger'])
```

```
In [57]: Tel.values()
```

```
Out[57]: dict_values(['06 05 04 03 02', '06 12 11 13 20'])
```

Sommaire

Descriptif
Modélisation
Vision statique : diagramme de Classes
Python
Juste assez de POO pour traduire notre UML en actes
Diagramme de cas d'utilisation

Diagramme d'activité
Diagramme de séquence
GIT
Dictionnaires
Découverte
Travail à faire
Lecture / Écriture de fichiers

Revenons à notre problème d'état-civil.

Revenons à notre problème d'état-civil.
On veut créer une table qui tient à jour les personnes inscrites.

Revenons à notre problème d'état-civil.
On veut créer une table qui tient à jour les personnes inscrites.
De quoi a-t-on besoin ?

```
class Table(object):

    def __init__(self) -> None:
        self.idmax = 0 # identifiant maxi courant
        self.taille = 0 # taille du dictionnaire
        self.table = {} # le dictionnaire
        self.inscrits = self.table.values() # les inscrits

    def __repr__(self) -> str:
        t = self.table
        return str({k: str(t[k]) for k in t.keys()})

    def __getitem__(self, k) -> Inscrit:
        """
        Pour pouvoir utiliser les crochets
        Si T est un objet Table
        T[10] renvoie l'inscrit portant l'identifiant 10
        """
        return self.table[k]

    def ajoute(self, p:Inscrit) -> None:

    def enleve(self, p:Inscrit) -> None:
```

Sommaire

Descriptif

Modélisation

Vision statique : diagramme de Classes

Python

Juste assez de POO pour traduire notre

UML en actes

Diagramme de cas d'utilisation

Diagramme d'activité

Diagramme de séquence

GIT

Dictionnaires

Découverte

Travail à faire

Lecture / Écriture de fichiers

On dispose de tableaux de données au format CSV



File Edit View Insert Format Sheet Data Tools Window Help					
C14					
	A	B	C	D	E
1	Nom	Sexe	Planete	NoCabine	
2	Zorglub	M	Trantor	1	
3	Blorx	M	Euterpe	2	
4	Urxiz	M	Aurora	3	
5	Zbleurdite	F	Trantor	4	
6	Dameurane	M	Trantor	4	
7	Mulzo	M	Helicon	6	
8	Zzzzzz	F	Aurora	7	
9	Arggh	M	Nexon	8	
10	Jorandum	F	Euterpe	9	
11					



CSV : Comma Separated Values

Nom, Sexe, Planete, NoCabine

Zorglub, M, Trantor, 1

Blorx, M, Euterpe, 2

Urxiz, M, Aurora, 3

Zbleurdite, F, Trantor, 4

Darneurane, M, Trantor, 4

Mulzo, M, Helicon, 6

Zzzzzzz, F, Aurora, 7

Arghh, M, Nexon, 8

Joranum, F, Euterpe, 9

```
BaseAliens = {  
    Alien('Zorglub', 'M', 'Trantor', '1'),  
    Alien('Blorx', 'M', 'Euterpe', '2'),  
    Alien('Urxiz', 'M', 'Aurora', '3'),  
    Alien('Zbleurdite', 'F', 'Trantor', '4'),  
    Alien('Darneurane', 'M', 'Trantor', '4'),  
    Alien('Mulzo', 'M', 'Helicon', '6'),  
    Alien('Zzzzzz', 'F', 'Aurora', '7'),  
    Alien('Arghh', 'M', 'Nexon', '8'),  
    Alien('Jorandum', 'F', 'Euterpe', '9')  
}
```

```
class Alien:
    def __init__(self, Nom, Sexe, Planete, NoCabine):
        self.Nom = Nom
        self.Sexe = Sexe
        self.Planete = Planete
        self.NoCabine = NoCabine
```

```
/home/moi/UCO/L1/ANGERS2018/Python/MIB_Files:
total used in directory 32 available 60318924
drwxr-xr-x 2 moi moi 4096 Nov 20 20:53 .
drwxr-xr-x 5 moi moi 4096 Nov 20 23:45 ..
-rw-r--r-- 1 moi moi 149 Nov 20 20:45 BaseAgents.csv
-rw-r--r-- 1 moi moi 199 Nov 20 20:46 BaseAliens.csv
-rw-r--r-- 1 moi moi 53 Nov 20 20:46 BaseCabines.csv
-rw-r--r-- 1 moi moi 98 Nov 20 10:53 BaseGardiens.csv
-rw-r--r-- 1 moi moi 163 Nov 20 20:46 BaseMiams.csv
-rw-r--r-- 1 moi moi 31 Nov 20 20:46 BaseResponsables.csv
```

```
/home/moi/UCO/L1/ANGERS2018/Python/MIB_Files:
total used in directory 32 available 60318924
drwxr-xr-x 2 moi moi 4096 Nov 20 20:53 .
drwxr-xr-x 5 moi moi 4096 Nov 20 23:45 ..
-rw-r--r-- 1 moi moi  149 Nov 20 20:45 BaseAgents.csv
-rw-r--r-- 1 moi moi  199 Nov 20 20:46 BaseAliens.csv
-rw-r--r-- 1 moi moi   53 Nov 20 20:46 BaseCabines.csv
-rw-r--r-- 1 moi moi   98 Nov 20 10:53 BaseGardiens.csv
-rw-r--r-- 1 moi moi  163 Nov 20 20:46 BaseMiams.csv
-rw-r--r-- 1 moi moi   31 Nov 20 20:46 BaseResponsables.csv
```

```
mes_csv_file = {Path(f).stem:open(f,"r") for f in glob.glob("./MIB_Files/*.csv")}
```



```
import glob
# The glob module finds all the pathnames matching a specified pattern according to the rules
#      used by the Unix shell
from pathlib import Path
# The final path component, without its suffix
```

```
In [58]: mes_csv_file
Out[59]:
{'BaseAgents': <_io.TextIOWrapper name='./MIB_Files/BaseAgents.csv' mode='r' encoding='UTF-8'>,
 'BaseAliens': <_io.TextIOWrapper name='./MIB_Files/BaseAliens.csv' mode='r' encoding='UTF-8'>,
 'BaseCabines': <_io.TextIOWrapper name='./MIB_Files/BaseCabines.csv' mode='r'
    encoding='UTF-8'>,
 'BaseGardiens': <_io.TextIOWrapper name='./MIB_Files/BaseGardiens.csv' mode='r'
    encoding='UTF-8'>,
 'BaseMiams': <_io.TextIOWrapper name='./MIB_Files/BaseMiams.csv' mode='r' encoding='UTF-8'>,
 'BaseResponsables': <_io.TextIOWrapper name='./MIB_Files/BaseResponsables.csv' mode='r'
    encoding='UTF-8'>}
```

```
mon_chemin = input('Quel est le chemin relatif du répertoire contenant les fichiers csv ?\n')
#"/MIB_Files/"

mon_alias = input('Alias du fichier py créé (sera ./MaBase_alias.py) ?\n')

mon_fic = "MaBase_%s.py" % mon_alias

mes_csv_file = {Path(f).stem:open(f,"r") for f in glob.glob(mon_chemin + "*.csv")}

mes_csv = {Path(f).stem:open(f,"r").readlines() for f in glob.glob(mon_chemin + "*.csv")}

mon_py = open(mon_fic,"w+")
```

```
In [60]: mes_csv
Out[60]:
{'BaseAgents': ['Nom,Ville\n',
                'Branno,Terminus\n',
                'Darell,Terminus\n',
                'Demerzel,Uco\n',
                'Seldon,Terminus\n',
                'Dornick,Kalgan\n',
                'Hardin,Terminus\n',
                'Trevize,Hesperos\n',
                'Pelorat,Kalgan\n',
                'Riose,Terminus\n'],
 'BaseAliens': ['Nom,Sexe,Planete,NoCabine\n',
                'Zorglub,M,Trantor,1\n',
                'Blorx,M,Euterpe,2\n',
                'Urxiz,M,Aurora,3\n',
                'Zbleurdite,F,Trantor,4\n',
                'Darneurane,M,Trantor,4\n',
                'Mulzo,M,Helicon,6\n',
                'Zzzzzz,F,Aurora,7\n',
                'Arghh,M,Nexon,8\n',
                'Joranum,F,Euterpe,9\n'],
 'BaseCabines': ['NoCabine,NoAllee\n',
                 '1,1\n',
                 '2,1\n',
                 '3,1\n',
                 '4,1\n',
                 '5,1\n',
                 '6,2\n',
                 '7,2\n',
                 '8,2\n',
                 '9,2\n']},
```

```
mon_py = open(mon_fic, "w+")
```

```
In [63]: b = 'BaseGardiens'
```

```
In [64]: mes_csv[b]
```

```
Out[64]:
```

```
['Nom,NoCabine\n',  
 'Branno,1\n',  
 'Darell,2\n',  
 'Demerzel,3\n',  
 'Seldon,4\n',  
 'Dornick,5\n',  
 'Hardin,6\n',  
 'Trevize,7\n',  
 'Pelorat,8\n',  
 'Riose,9\n']
```

```
In [65]: lignes = mes_csv[b]
```

```
In [66]: lignes[2]
```

```
Out[66]: 'Darell,2\n'
```

```
In [65]: lignes = mes_csv[b]
```

```
In [66]: lignes[2]
```

```
Out[66]: 'Darell,2\n'
```

```
In [67]: lignes[2].split()
```

```
Out[67]: ['Darell,2']
```



```
In [65]: lignes = mes_csv[b]
```

```
In [66]: lignes[2]
```

```
Out[66]: 'Darell,2\n'
```

```
In [67]: lignes[2].split()
```

```
Out[67]: ['Darell,2']
```

```
In [68]: lignes[2].split()[0]
```

```
Out[68]: 'Darell,2'
```

```
In [65]: lignes = mes_csv[b]
```

```
In [66]: lignes[2]
```

```
Out[66]: 'Darell,2\n'
```

```
In [67]: lignes[2].split()
```

```
Out[67]: ['Darell,2']
```

```
In [68]: lignes[2].split()[0]
```

```
Out[68]: 'Darell,2'
```

```
In [69]: lignes[2].split()[0].split(',')
```

```
Out[69]: ['Darell', '2']
```



```
def creer_classes():
    for b in mes_csv:
        mon_py.write("class " + b[4:-1] + ":\n\tdef __init__(self")
        lignes = mes_csv[b]
        attributs = lignes[0].split()[0].split(',')
        for a in attributs:
            mon_py.write(", " + a)
        mon_py.write("):\n\t\t")
        for a in attributs:
            mon_py.write("self.%s = %s\n\t\t" % (a,a))
        mon_py.write("\n\n")
```

```
class Alien:
    def __init__(self, Nom, Sexe, Planete, NoCabine):
        self.Nom = Nom
        self.Sexe = Sexe
        self.Planete = Planete
        self.NoCabine = NoCabine

class Cabine:
    def __init__(self, NoCabine, NoAllee):
        self.NoCabine = NoCabine
        self.NoAllee = NoAllee
```

```
def creer_bases():
    for b in mes_csv:
        nom = b[4:-1]
        mon_py.write(b + " = { ")
        lignes = mes_csv[b]
        for index, ligne in enumerate(lignes[1:]):
            ligne = ligne.split()[0].split(',')
            debut = '' if index == 0 else ', '
            mon_py.write(debut + nom + "(")
            for att in ligne[:-1]:
                mon_py.write("'" + att + "', ")
            mon_py.write("'" + ligne[-1] + "'")
        mon_py.write(" }\n\n")
```

```
BaseCabines = { Cabine('1', '1'), Cabine('2', '1'), Cabine('3', '1'), Cabine('4', '1'),  
    ↳ Cabine('5', '1'), Cabine('6', '2'), Cabine('7', '2'), Cabine('8', '2'),  
    ↳ Cabine('9', '2') }  
  
BaseMiams = { Miam('Zorglub', 'Bortsch'), Miam('Blorx', 'Bortsch'), Miam('Urxiz', 'Zoumise'),  
    ↳ Miam('Zbleurdite', 'Bortsch'), Miam('Darneurane', 'Schwanstucke'),  
    ↳ Miam('Mulzo', 'Kashpir'), Miam('Zzzzzz', 'Kashpir'), Miam('Arghh', 'Zoumise'),  
    ↳ Miam('Joranum', 'Bortsch') }
```

```
def ferme():  
    for b in mes_csv_file:  
        mes_csv_file[b].close()  
    mon_py.close()
```

```
In [70]: villes = { agent.Ville for agent in BaseAgents }
```

```
In [71]: villes
```

```
Out[71]: {'Hesperos', 'Kalgan', 'Terminus', 'Uco'}
```


À vous de jouer...

```
# -1 l'ensemble des gardiens ;  
gardiens = { gardien.nom for gardien in baseGardien }
```

```
# -2 l'ensemble des villes où habitent les agents ;  
villes = { agent.ville for agent in baseAgent }
```

```
# -3 l'ensemble des triplets (no de cabine,alien,gardien) pour chaque cabine ;  
triples = { (alien.no_cabine, alien.nom, gardien.nom)  
            for alien in baseAlien  
            for gardien in baseGardien if gardien.no_cabine == alien.no_cabine}
```

```
# -4 l'ensemble des couples (alien,allée) pour chaque alien ;  
# -5 l'ensemble de tous les aliens de l'allée 2 ;  
# -6 l'ensemble de toutes les planètes dont sont originaires les aliens habitant une cellule de  
    ↪    numéro pair ;  
# -7 l'ensemble des aliens dont les gardiens sont originaires de Terminus ;  
# -8 l'ensemble des gardiens des aliens féminins qui mangent du bortsch ;  
# -9 l'ensemble des cabines dont les gardiens sont originaires de Terminus ou dont les aliens  
    ↪    sont des filles
```

```
# Y a-t-il
# -10 des aliments qui commencent par la même lettre que le nom du gardien qui surveille
#      l'alien qui les mange ?
test10 = True in { miam.aliment[0] == gardien.nom[0]
                  for miam in baseMiam
                  for alien in baseAlien
                  for gardien in baseGardien
                  if gardien.no_cabine == alien.no_cabine and alien.nom ==
                    miam.nom_alien }
```

```
# -11 des aliens originaires d'Euterpe ?  
# Est-ce que  
# -12 tous les aliens ont un 'x' dans leur nom ?
```

```
# -13 tous les aliens qui ont un 'x' dans leur nom ont un gardien qui vient de Terminus ?
test13 = False not in { agent.ville == 'Terminus' and 'x' in alien.nom
                        for alien in baseAlien
                        for gardien in baseGardien
                        for agent in baseAgent
                        if gardien.no_cabine == alien.no_cabine and gardien.nom == agent.nom }
# -14 il existe un alien masculin originaire de Trantor qui mange du Bortsch ou dont le gardien
    ↪ vient de Terminus
```