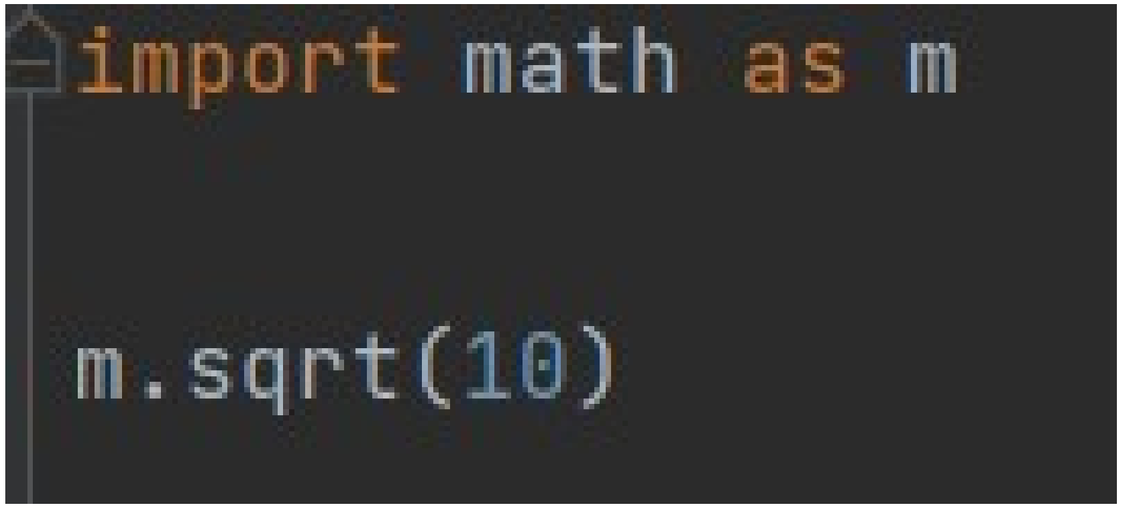
# Modules Python

Dany Siriphol – CODA - 2025

# Module

Un module est une librairie python contenant des variables, fonctions ou classes qui peut être reutilisée dans plusieurs projets

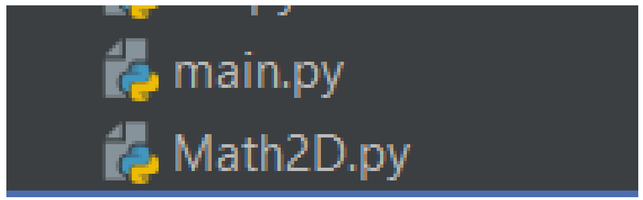
# Import de modules : renommage



# Créer ses propres modules

Les modules permettent de séparer un code en plusieurs fichiers.
Les fichiers dans le dossier courant sont considérés comme des modules que l’on peut importer

```
from Math2D import *
```

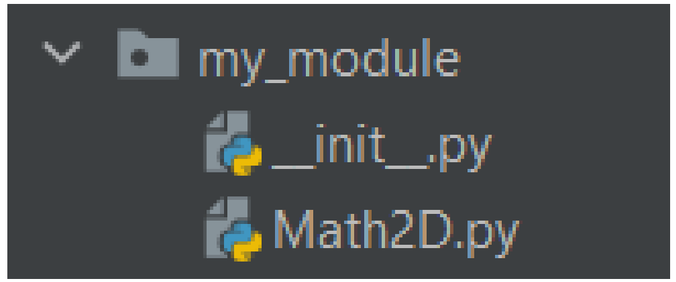



```python
import math

class Vector2D:

    def __init__(self, x, y):
        self._x: float = x
        self.y: float = y

    def get_distance(self):
        return math.sqrt((self._x) ** 2 + (self.y) ** 2)

    def normalize(self):
        distance = self.get_distance()
        self._x /= distance
        self.y /= distance
```

# Créer un package

Il est possible de regrouper plusieurs modules dans un dossier :

- un dossier qui sert de package doit contenir le fichier __init__.py
- chaque fichier du package est un module qui peut être importé individuellement




```
from my_module.Math2D import *
```

# package : import automatique

Il est possible d’importer tout les modules d’un package automatiquement
Pour cela il faut configurer le __init__.py avec les imports à effectuer. Ce fichier est un script python appelé automatiquement lors du premier import du package

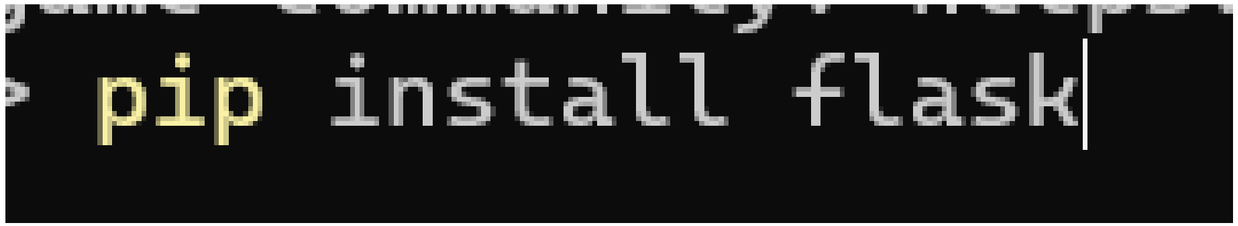
```python
from .player import *
from .enemy import *
from .item import *

__all__ = ["Player", "Enemy", "Item"]
```

__init__.py

```python
from game import *
```

# Installation de module avec pip

pip est le gestionnaire de package de python. Similaire à Composer ou NPM, il permet d’installer des libraries globalement sur la machine




```
from flask import Flask, render_template
```

# Développement web avec Python avec Flask

Dany Siriphol – CODA - 2025

# Flask

pip est le gestionnaire de package de python. Similaire à Composer ou NPM, il permet d’installer des libraries globalement sur la machine

Flask

# Démarrer un serveur flask

# Routes et URL

Dany Siriphol – CODA - 2025

# Routes

Une route est un mapping entre une URL et une fonction Python

Une route doit retourner la string qui sera affichée par le navigageur

```python
@app.route("/")
def home():
    return "<h1>Hello World!</h1>"
```

# Route avec paramètres

```python
@app.route("/hello/<name>")
def hello(name):
    return f"<h1>Hello {name}!</h1>"
```

exemple de route : site.com/hello/dany

# Templates

Dany Siriphol – CODA - 2025

# Templates

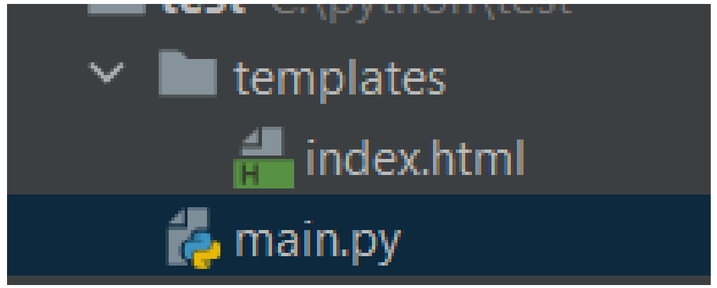
Il est peu recommendé de gérer l’affichage directement dans le code. Pour séparer la logique de l’affichage, Flask utilise un système de template

```
<html>
    <head>

    </head>
    <body>
        <h1> Hello World</h1>
    </body>
</html>
```

# Afficher un template

Les templates sont des fichiers HTML se trouvant dans le dossier **templates** du projet

La fonction **render_template** permet d’afficher un template HTML




```python
from flask import Flask, render_template

app = Flask(__name__)

@app.route("/")
def home():
    return render_template("index.html")
```

# Langage de templating

La plupart des frameworks utilisent un système de template qui permet séparer l’algorithme d’affichage et le code backend.

Flask utilise Jinja pour ses templates. C’est un langage très simplifié qui permet de d’exprimer de la logique simple dans du HTML

Jinja

# Passer des variables aux template

Pour passer des données du code backend aux templates, il faut donner en arguments de render_template la liste des variables à utiliser dans les templates

```python
@app.route("/")
def home():
    name = "Dany"
    return render_template("index.html", name=name)
```

```
<h1> Hello {{ name }}</h1>
```

Afficher une variable avec Jinja

# Créer des liens

La fonction **url_for** permet de créer des liens qui menent vers d'autres route
url_for prend en paramètre le nom de la fonction du code backend

```
<nav>
  <a href="{{ url_for('home') }}">Accueil</a> |
  <a href="{{ url_for('about') }}">À propos</a> |
  <a href="{{ url_for('contact') }}">Contact</a>
</nav>
```

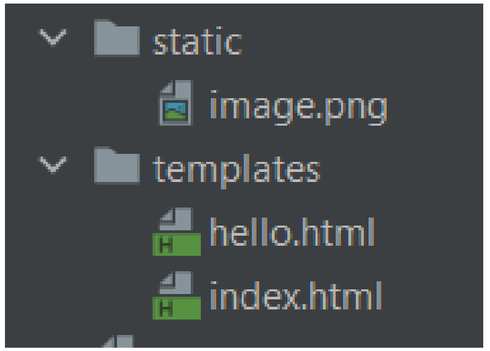
```
@app.route("/")
def home():
```

# Afficher des assets

Pour afficher des assets, la fonction url_for permet d'obtenir l'url d'un fichier se trouvant dans le dossier « **static** » du projet

```
{{ url_for('static', filename='style.css') }}
```

```
<body>
    <h1> Hello World</h1>

    <img src="{{ url_for('static', filename='image.png')}}">
</body>
```



# Conditions

```python
@app.route("/")
def home():
    return render_template("index.html", age=18)
```

```
{% if age >= 18 %}
    <p> vous etes majeur</p>
{% else %}
    <p> vous etes mineur</p>
{% endif %}
```

# Template : Dictionnaire

```python
@app.route("/")
def home():
    car = {"brand": "Volvo", "price": 50000}
    return render_template("index.html", car=car)
```

```html
<p> Marque : {{ car.brand }} </p>
<p> Prix : {{ car.price }}</p>
```

Marque : Volvo

Prix : 50000

# Template : Collections

Pour parcourir des collections, il est courant d’utiliser la boucle **for … in**

```
products = ['Apple', 'Banana', 'Orange']
return render_template("index.html", products=products)
```

```
<ul>
    {% for product in products %}
        <li> {{ product }} </li>
    {% endfor %}
</ul>
```

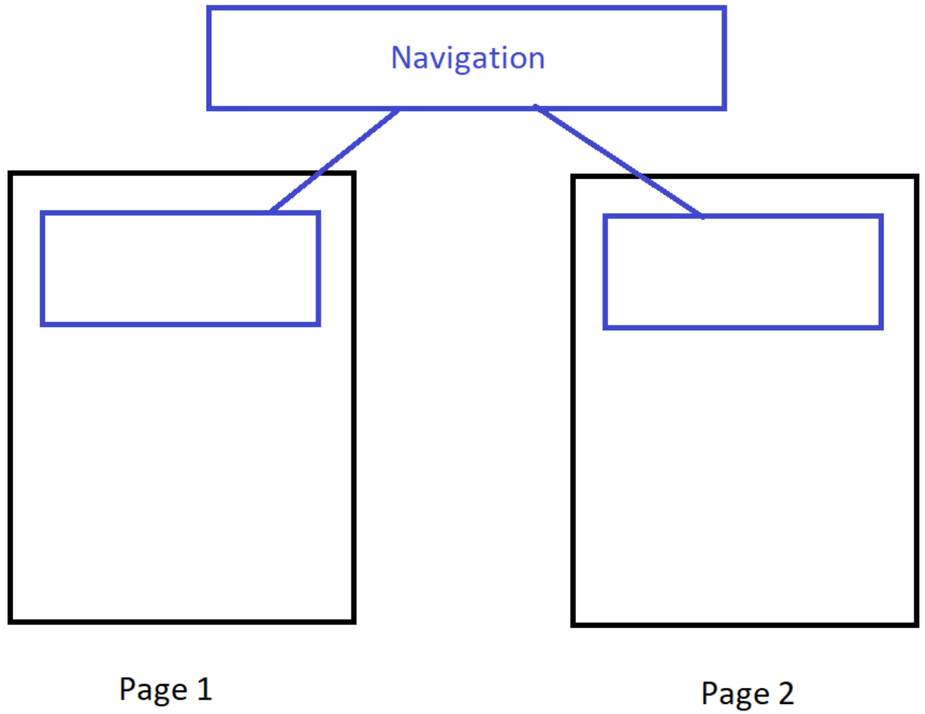
Marque : Volvo

Prix : 50000

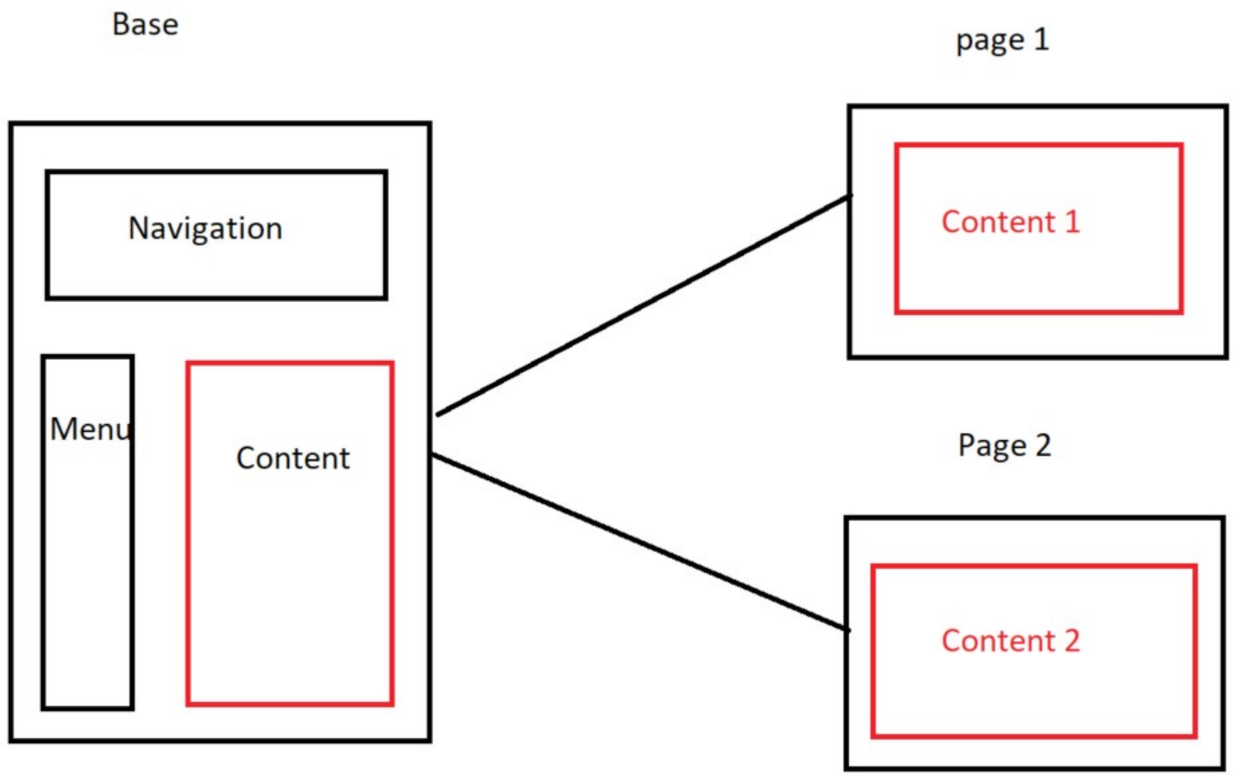
# Héritage de template

Dany Siriphol – CODA - 2025

# Héritage de template

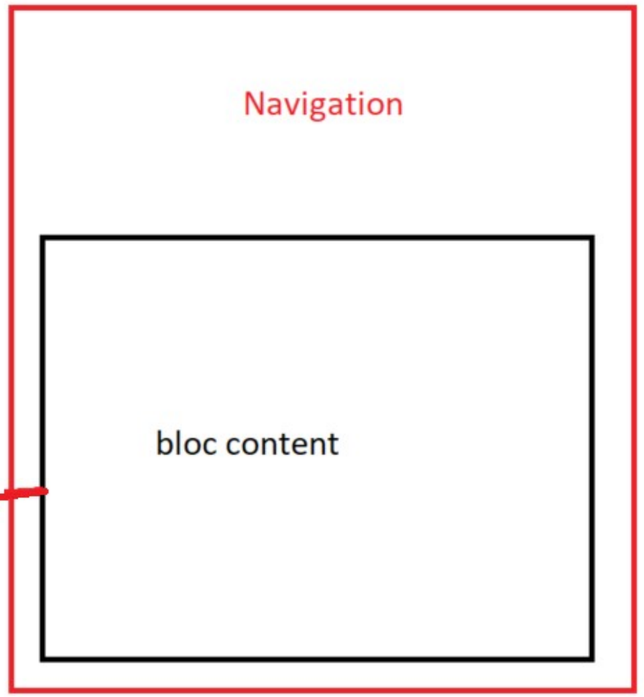


# Héritage de template

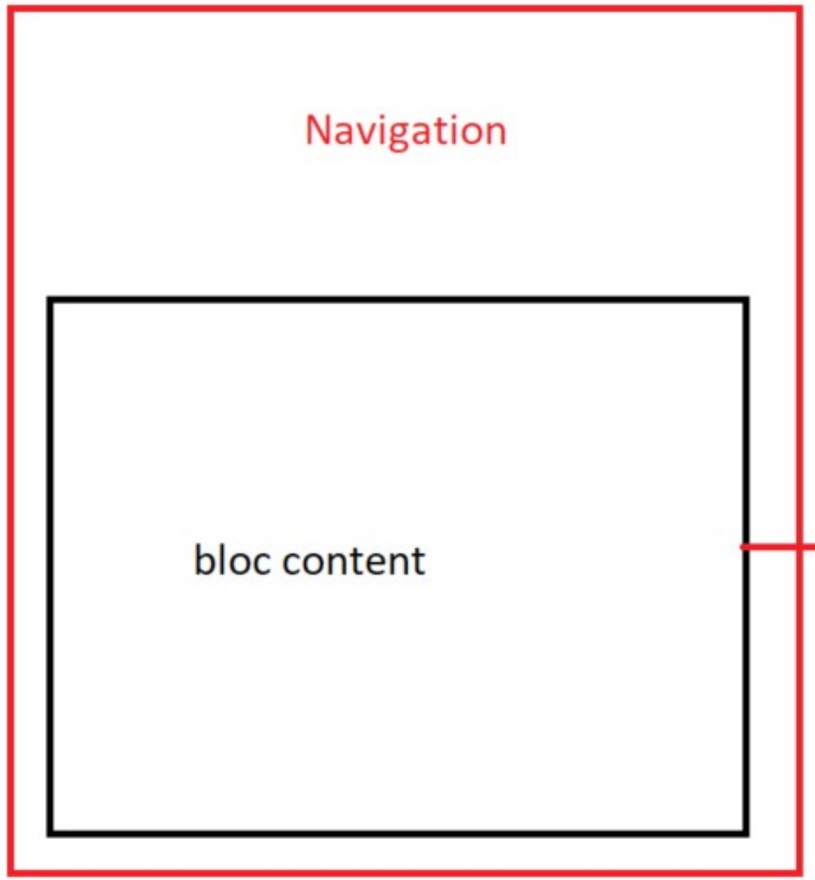


# Héritage de template

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <title>{% block title %}Mon Application{% endblock %}</title>

    {% block stylesheets %}
        <link rel="stylesheet" href="{{ url_for('static', filename='css/base.css') }}">
    {% endblock %}
</head>
<body>

    <nav>
        <a href="{{ url_for('home') }}">Lien 1</a>
        <a href="{{ url_for('about') }}">Lien 2</a>
    </nav>

    <div id="content">
        {% block content %}{% endblock %}
    </div>

</body>
</html>
```



# Héritage de template




```
{% extends "base.html" %}

{% block title %}
    Titre page
{% endblock %}

{% block content %}
    <h1>Contenu</h1>

    <p> contenu de la page 1 </p>
{% endblock %}
```

# Exercices

Dany Siriphol – CODA - 2025