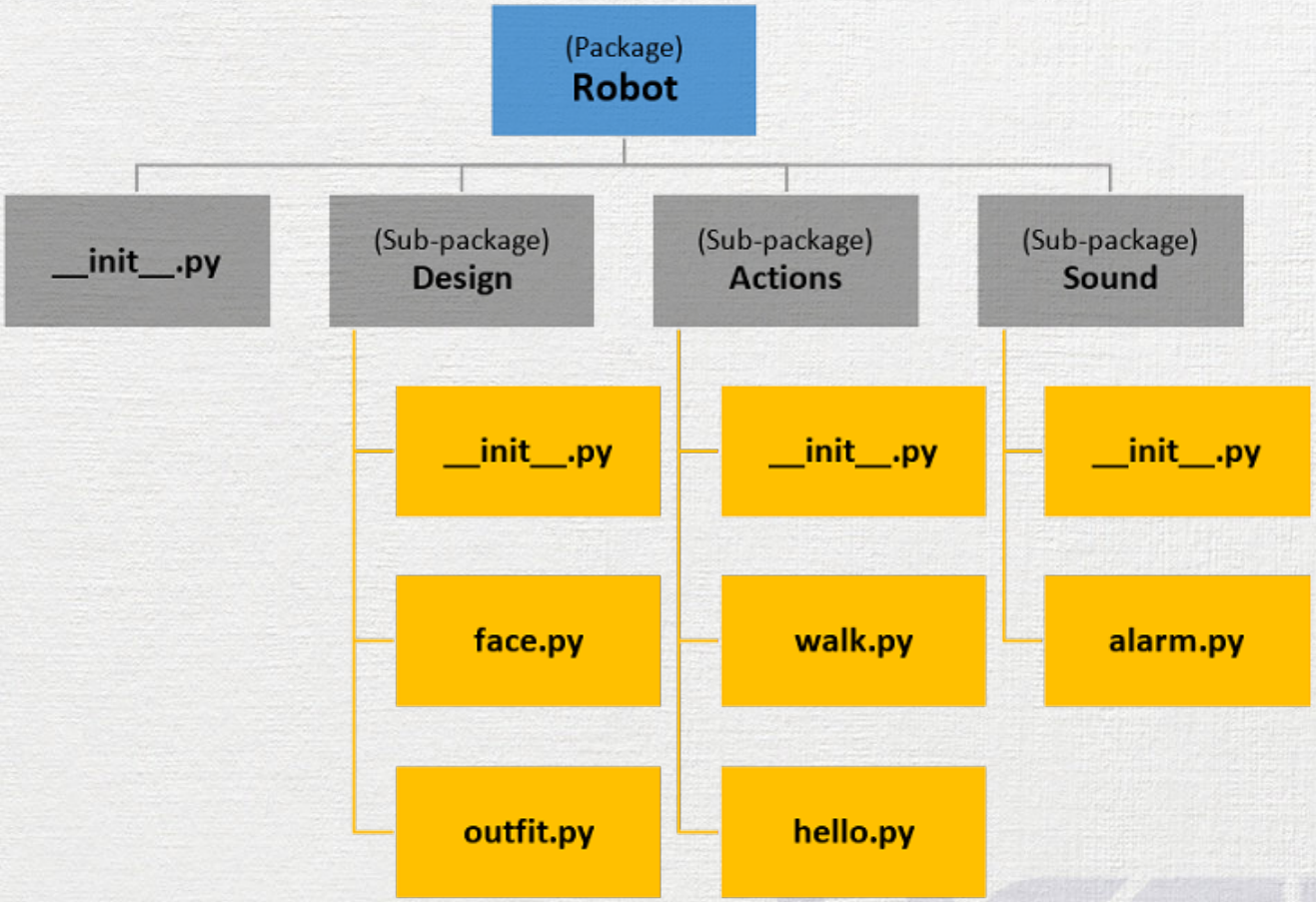
Modules and Packages

Tran Giang Son, tran-giang.son@usth.edu.vn

ICT Department, USTH

Intro


(Package)
Robot
__init__.py
(Sub-package)
Design
(Sub-package)
Actions
(Sub-package)
Sound
__init__.py
face.py
outfit.py
__init__.py
walk.py
hello.py
__init__.py
alarm.py

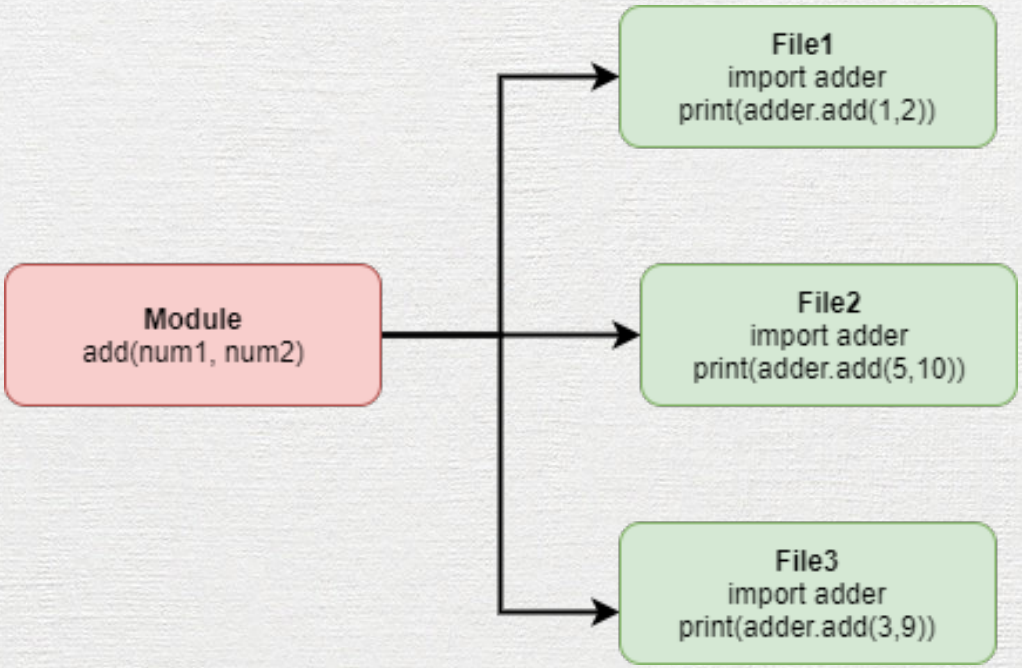
Modules

What

- Module: reusable piece of code
  - Can be from external sources
  - Regular `.py` file with `def`s and `class`es

What

- Similar to `.ko`, `.so`, `.dll`, ...



Why

- Modularity
- Reusability
- Shareibility
- Maintainability

How: Making a module

- Create a separated `.py` file
  - `define` functions, classes as usual
  - `define` `__init__()`, as needed

How: Using a module

- `import` a module to the global namespace
  - No path, no extension
- Use functions, classes, constants provided by module
  - `<module>.<method/class/const>`

```
>>> import math
>>> print(math.pi)
3.141592653589793
```

How: Using a module

- Shortcut to the global namespace
  - `from <module> import <func/class/const>`
  - Use `<func/class/const>` directly

```
>>> from math import pi
>>> print(pi)
3.141592653589793
>>> from math import *
>>> print(e)
2.718281828459045
```

How: Using a module

- Aliasing
  - `from <module> import <func/class/const> as <alias>`
  - Use `<alias>`

```
>>> import numpy as np
>>> np.pi
3.141592653589793
```

Packages

What

- A bunch of related modules
- [optional] A bunch of sub-packages
- [optional] A bunch of sub-sub-packages

Why

- Higher level of modularity
- Less `import` modules from the same packages
- Module name A.B designates a submodule named B in a package named A.

How

- Install from Python Package Index (PyPI)
  - `pip install <packageName>`
- Write your own package
  - Add your modules
  - Add a dedicated file `__init__.py` for initialization

How

- `import` a module in a specific package

```
import Package1.PackageModule1
```

- `import` a whole package

```
import Package1
```

- Should also import modules in package `__init__.py` for automatic module imports

```
import Module1
import Module2
import Module3
```

How

- Package can be nested
  - Sub-package inside a package
    - Sub-sub-package inside a sub-package
- Just make an `__init__.py`, even empty one

Practice!

Practical work 3: some maths and decorations

- Copy your practical work 2 to 3.student.mark.oop.math.py
- Use `math` module to round-down student scores to 1-digit decimal upon input, `floor()`
- Use `numpy` module and its `array` to
  - Add function to calculate average GPA for a given student
    - Weighted sum of credits and marks
  - Sort student list by GPA descending
- Decorate your UI with `curses` module
- Push your work to corresponding forked Github repository

Practical work 4: modularization

- Split your program 3.student.mark.oop.math.py to modules and packages in a new `pw4` directory
  - `input.py`: module for input
  - `output.py`: module for `curses` output
  - `domains`: package for classes
  - `main.py`: main script for coordination
- Push your work to corresponding forked Github repository