**O seu primeiro pacote: passo 1**

Imagine que num futuro não tão distante, você e os seus associados escrevem um grande número de funções Python.

A sua equipa decide agrupar as funções em módulos separados, e este é o resultado final da ordenação:

#! /usr/bin/env python3

""" module: alpha """

def funA():

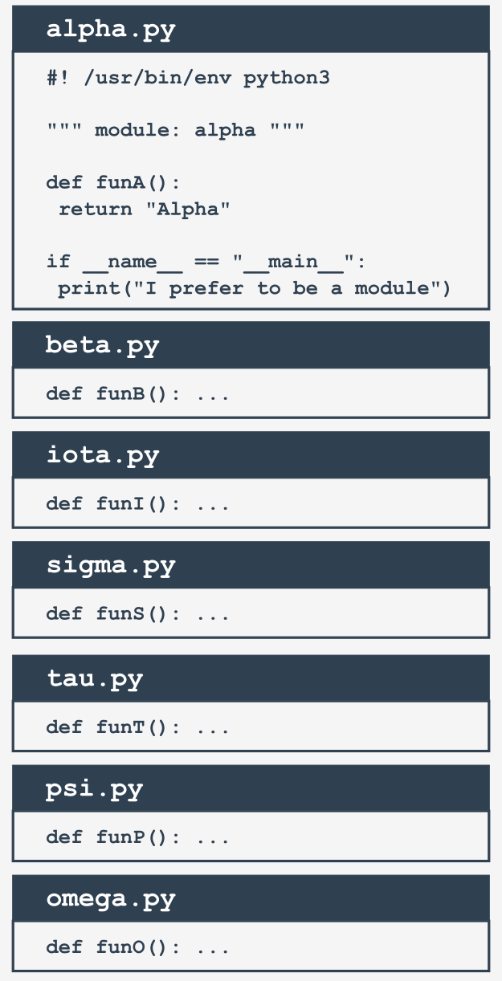
return "Alpha"

if \_\_name\_\_ == "\_\_main\_\_":

print("I prefer to be a module.")

alpha.py

Nota: apresentámos todo o conteúdo apenas para o módulo alpha.py - assumir que todos os módulos são semelhantes (eles contêm uma função chamada funX, onde *X* é a primeira letra do nome do módulo).



**O seu primeiro pacote: passo 2**

De repente, alguém repara que estes módulos formam a sua própria hierarquia, pelo que colocá-los todos numa estrutura plana não será uma boa ideia.

Após alguma discussão, a equipa chega à conclusão de que os módulos têm de ser agrupados. Todos os participantes concordam que a seguinte estrutura em árvore reflete perfeitamente as relações mútuas entre os módulos:



Vamos rever isto de baixo para cima:

* o grupo ugly contém dois módulos: psi e omega;
* o grupo best contém dois módulos: sigma e tau;
* o grupo good contém dois módulos (alfa e beta) e um subgrupo (best)
* o grupo extra contém dois subgrupos (good e bad) e um módulo (iota)

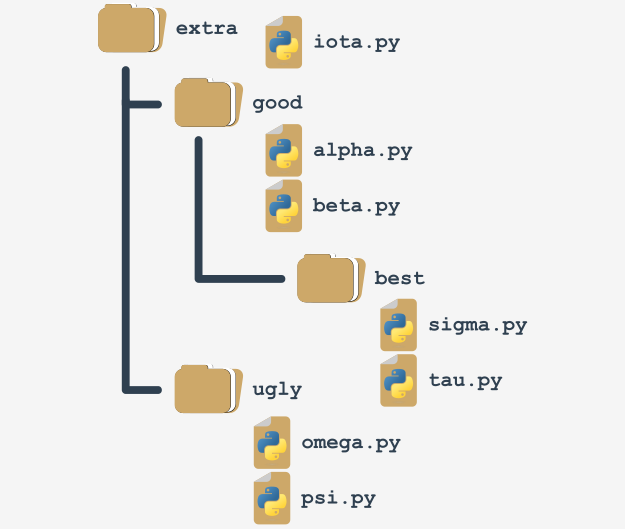
Tem mau aspecto? De modo algum - analise cuidadosamente a estrutura. Assemelha-se a algo, não é?

Parece ser uma **estrutura de diretoria**.

Vamos construir uma árvore refletindo as dependências projetadas entre os módulos.

**O seu primeiro pacote: passo 3**

Esta é a aparência atual da árvore:



Tal estrutura é quase um pacote (no sentido Python do termo). Falta-lhe o detalhe de ser ao mesmo tempo funcional e operacional. Vamos completá-lo num momento.

Se assumir que extra é o nome de um **pacote recém-criado** (pense nele como a **raiz do pacote**), ele imporá uma regra de nomenclatura que lhe permite nomear claramente cada entidade da árvore.

Por exemplo:

* a localização de uma função chamada funT() a partir do pacote tau pode ser descrito como:

extra.good.best.tau.funT()

* uma função marcada como:

extra.ugly.psi.funP()

vem do módulo psi a ser armazenado no ugly subpacote do pacote extra .

**O seu primeiro pacote: passo 4**

Há duas perguntas a responder:

* **como** transforma tal árvore (na realidade, uma subárvore) num **pacote** Python real (por outras palavras, como convencer o Python de que tal árvore não é apenas um monte de junk files, mas um conjunto de módulos)?
* **onde** coloca a subárvore para a tornar acessível ao Python?

A primeira pergunta tem uma resposta surpreendente: **os pacotes, como os módulos, podem precisar de inicialização**.

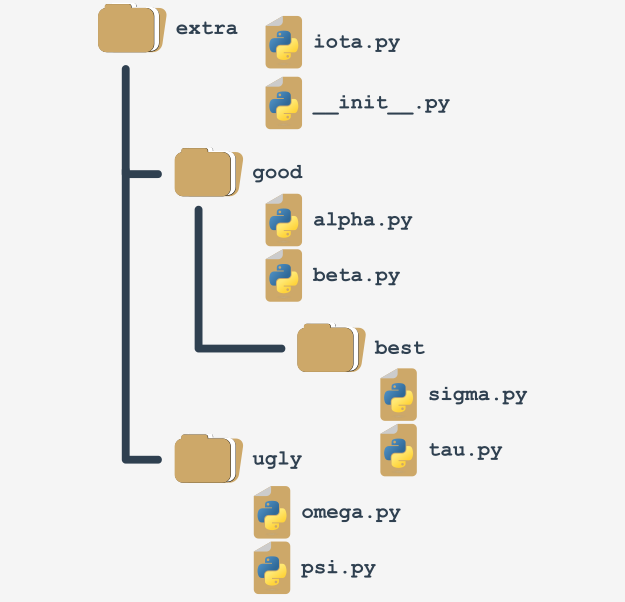
A inicialização de um módulo é feita por um *código unbound (não vinculado) (não faz parte de nenhuma função) localizado dentro do ficheiro do módulo. Como um pacote não é um ficheiro, esta técnica é inútil para inicializar pacotes.*

*Em vez disso, é preciso usar um truque diferente - o Python espera que haja um ficheiro com um nome muito único dentro da pasta do pacote: \_\_init\_\_.py.*

*O conteúdo do ficheiro é executado quando qualquer um dos módulos do pacote é* ***importado****. Se não quiser inicializações especiais, pode deixar o ficheiro vazio, mas não o deve omitir.*

**O seu primeiro pacote: passo 5**

Lembre-se: **a presença do ficheiro \_\_init.py\_\_ finalmente compõe o pacote**.



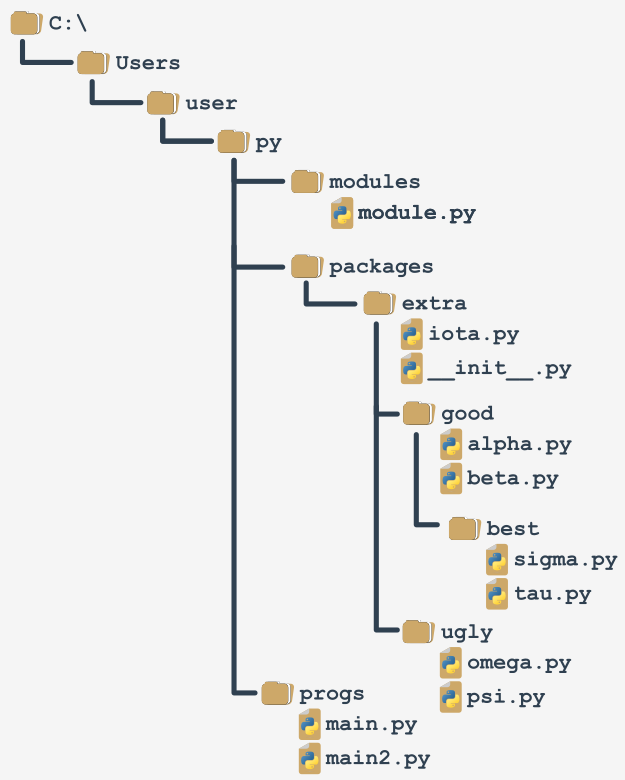
Nota: não é apenas a *root* file que pode conter o ficheiro \_\_init.py\_\_ - também pode colocá-lo dentro de qualquer uma das suas subpastas (subpacotes). Pode ser útil se alguns dos subpacotes requerem tratamento individual e tipos especiais de inicialização.

Agora está na altura de responder à segunda pergunta - a resposta é simples: **em qualquer lugar**. Só precisa de garantir que o Python esteja ciente da localização do pacote. Já sabe como fazê-lo.

Está pronto para fazer uso do seu primeiro pacote.

**O seu primeiro pacote: passo 6**

Vamos supor que o ambiente de trabalho tem o seguinte aspeto:



Preparámos um ficheiro zip contendo todos os ficheiros do ramo de pacotes. Pode descarregá-lo e usá-lo nas suas próprias experiências, mas **lembre-se de descompactá-lo na pasta apresentada no esquema**, caso contrário ele não estará acessível ao código a partir do main file.

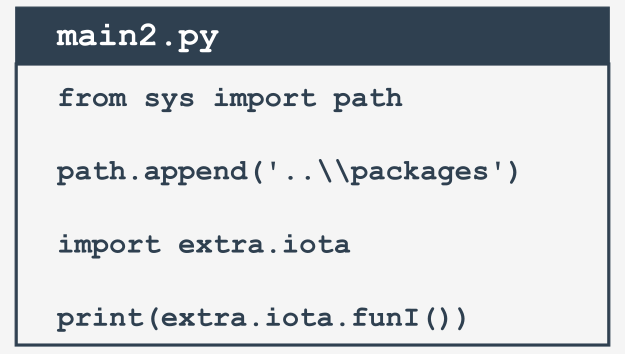
**DOWNLOAD** [Modules and Packages ZIP file](https://edube.org/uploads/media/default/0001/01/39a03067f85a9b39c0c94ca50b5d445b7df18a04.zip)

Continuará as suas experiências utilizando o main2.py ficheiro.

**O seu primeiro pacote: passo 7**

Vamos aceder à função funI() do módulo iota a partir do topo do pacote extra . Isto obriga-nos a utilizar nomes de pacotes qualificados (associe isto com a nomeação de pastas e subpastas - as convenções são muito semelhantes).

Esta é a forma de o fazer:



from sys import path

path.append('..\\packages')

import extra.iota

print(extra.iota.funI())

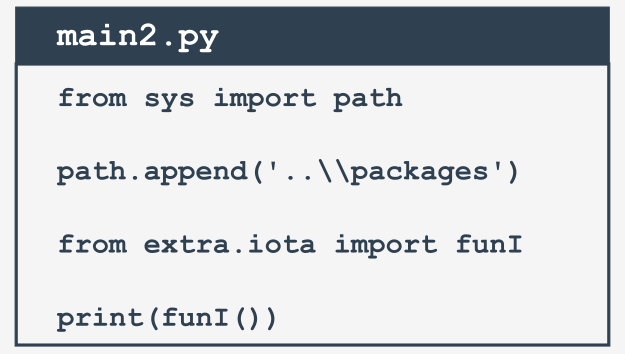
main2.py

Nota:

* modificámos a variável path para a tornar acessível ao Python;
* o ramo import não aponta diretamente para o módulo, mas especifica o caminho totalmente qualificado a partir do topo do pacote;

substituir import extra.iota por import iota causará um erro.

A seguinte variante também é válida:



from sys import path

path.append('..\\packages')

from extra.iota import funI

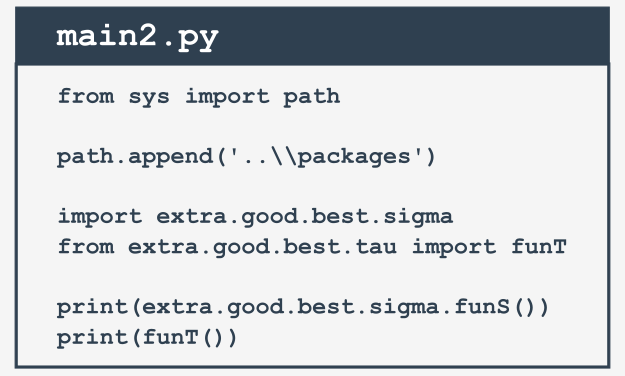
print(funI())

main2.py

Observe o nome qualificado do módulo iota .

**O seu primeiro pacote: passo 8**

Vamos agora chegar até ao fundo da árvore - é assim que se pode ter acesso aos módulos sigma e tau :



from sys import path

path.append('..\\packages')

import extra.good.best.sigma

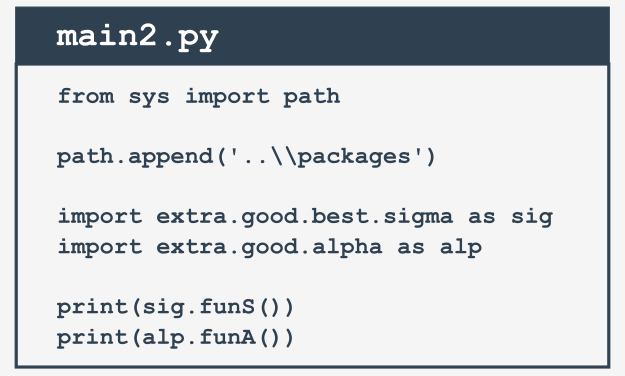
from extra.good.best.tau import funT

print(extra.good.best.sigma.funS())

print(funT())

main2.py

Pode tornar a sua vida mais fácil através da utilização de aliasing:



from sys import path

path.append('..\\packages')

import extra.good.best.sigma as sig

import extra.good.alpha as alp

print(sig.funS())

print(alp.funA())

main2.py

**O seu primeiro pacote: passo 9**

Vamos assumir que zipámos toda a subdiretoria, a partir da pasta extra (incluindo a pasta), e vamos obter um ficheiro com o nome extrapack.zip. A seguir, colocamos o ficheiro dentro da pasta packages .

Agora podemos utilizar o ficheiro zip com o papel de pacotes:

from sys import path

path.append('..\\packages\\extrapack.zip')

import extra.good.best.sigma as sig

import extra.good.alpha as alp

from extra.iota import funI

from extra.good.beta import funB

print(sig.funS())

print(alp.funA())

print(funI())

print(funB())

main2.py

Se quiser realizar as suas próprias experiências com o pacote que criámos, pode descarregá-lo em baixo. Encorajamo-lo a fazê-lo.

**DOWNLOAD** [Extrapack ZIP file](https://edube.org/uploads/media/default/0001/01/d9df38daa0410952c4cbf85b47892954c45b9215.zip)

Agora pode criar módulos e combiná-los em pacotes. É tempo de iniciar uma discussão completamente diferente - sobre erros, falhas e crashes.