Aula 22 - Módulos e pacotes

18/07/2022 - 14:53

Esta é a minha transição para o mac...
Mas não definitiva. Só por enquanto...
Quer saber como se coloca "ç" no mac? Use o comando option(alt) + c e verá a mágica acontecer!

Bom, agora vamos falar sobre módulos em um ambiente novo.

O conceito de modularização surgiu por volta da década de 60, no início dela, sendo levemente mais exato. A computação havia quebrado barreiras e empresas passaram a entrar nesta área. Naturalmente, os programas desenvolvidos passara a ficar maiores, o que trouxe a clássica necessidade de otimizar os processos de criação.

Dividir para conquistar (ferramentas de título!)

Não dá para incluir tudo em um único programa só. Ele ficaria extenso demais, pesado demais, e difícil demais para corrigir ou melhorar. A divisão facilita a legibilidade do programa.

Mini código de exemplo:

```
def fatorial(number):
	f = 1
	for c in range(1, number+1):
		f *= c
	return f

def duplo(number):
	return number * 2


# Programa Principal
n = int(input('Digite um número e eu lhe direi seu fatorial: '))
fat = fatorial(n)
print(f'O fatorial de {n} é {fat}')
print(f'O duplo de {n} é {duplo(n)}')
```

Este era o funcionamento base dos nossos programas.
Para trabalharmos com modularizações, criaremos um novo projeto "módulos".

Bom, com mais de 2 "defs" fica claro que o programa se torna um livro de

muitas páginas. Então, criaremos um arquivo “úteis.py”, que vai abarcar todas as definições de funções.
Neste caso, não funcionaria rodar o resto do programa sem as defs. Para isso, teríamos que usar um “import uteis”, ou seja, criaríamos nossas próprios módulos.

```
import uteis
n = int(input('Digite um número e eu lhe direi seu fatorial: '))
fat = fatorial(n)
print(f'O fatorial de {n} é {fat}')
print(f'O duplo de {n} é {duplo(n)}')
```

O python já recomenda o arquivo uteis pois, para ele, qualquer arquivo “.py” pode ser um módulo contanto que tenha funções internas.

Também posso usar a estratégia “from uteis import fatorial, duplo”, mas lembre-se que, se houver uma outra importação que já contenha uma função chamada duplo, o python considera a função do módulo que foi importado por último.

Vantagens

- Organização do código
- Facilidade na manutenção
- Ocultação do código detalhado (não a necessidade de saber o que foi feito para atingir o resultado da função)
- Reutilização em outros projetos

Quando ficar claro que a importação de funções é muito grande e extensa, é possível abarcar todos os módulos em pacotes. Algumas linguagens costumam chamar isto de bibliotecas, mas o python define como pacotes.

E como crio pacotes? Utilizando pastas!
Da mesma forma que arquivos .py são potenciais módulos, pastas são potenciais pacotes. Basta criar uma nova pasta no projeto python.
Posso colocar pacotes dentro de um pacote, separando-os por assuntos.

Por razões extra-lógicas, cada pasta de assunto precisa conter um arquivo chamado __init__.py, e isso pode ser feito na própria pasta do pacote principal.
No pycharm, para criar um pacote, eu crio um python package.

Curiosidade do exercício 110: O uso de \t realiza uma tabulação dos elementos.
Ex:

```
print(f'Preço analisado: \t{moeda(price)}')
```