



Prof. Luciano Nunes
2019/1
lnunes at gmail

Módulos e Pacotes

- Módulos e pacotes são conceitos básicos em Python
- Um script Python é considerado um módulo (arquivos com extensão .py)
- Uma pasta dentro da estrutura do projeto é considerada como pacote, porém para que o interpretador entenda que se trata efetivamente de um pacote e não de uma pasta qualquer, é necessário a presença de um arquivo denominado `__init__.py`
- O arquivo `__init__.py` é responsável por informar ao interpretador, quais são os módulos presentes naquele pacote, evitando assim o *scan* do pacote
- Cada pacote pode conter vários módulos (arquivos .py), que são denominados submódulos
- A criação de pacotes e a distribuição dos programas em submódulos permite organizar de forma bastante flexível seus projetos
- Uma vez que as funções estão organizadas em módulos e pacotes, torna-se muito mais fácil reaproveitá-las

Módulos e Pacotes

- Quando se pretende utilizar funções que estão distribuídas em módulos e pacotes, é necessário informar ao interpretador dessa intenção, utilizando o comando *import*
- Importar um módulo ou pacote pode ser feito de várias forma diferentes:

```
main.py x
1 import math
2 print(math.pow(2, 3))
3
```

```
main.py x
1 from math import pow
2 print(pow(2, 3))
3
```

```
main.py x
1 from math import *
2 print(pow(2, 3))
3 print(sqrt(4))
4
```

Módulos e Pacotes

- Note que com o comando *import* é possível importar um subpacote, submodulo, classe, função ou variável
- O comando import primeiro testa se o item está definido no pacote, senão assume que é um módulo e tenta carregá-lo. Se falhar em encontrar o módulo uma exceção *ModuleNotFoundError* é lançada
- Na utilização do terminal, o uso do statement “*import **” é muito utilizado para facilitar a digitação e tornar a programação mais rápida, no entanto, fora deste ambiente é uma prática desaconselhável uma vez que dificulta o entendimento do código
- O arquivo `__init__.py` além de informar o interpretador que trata-se de um pacote, pode conter código de inicialização do pacote
- A variável `__all__` é utilizada para informar ao interpretador quais são os submódulos presentes no pacote em forma de lista
 - `__all__ = ['submodulo1', 'submodulo2', ..., 'submodulon']`
- A função embutida `dir()` é usada para se descobrir quais nomes são definidos por um módulo, ela devolve a tabela de símbolos em forma de uma lista ordenada de strings
- Para saber mais: <https://docs.python.org/3/reference/import.html#packages>

Módulos datetime, time e calendar

- Python possui alguns módulos com diversos tipos e funções específicos para manipulação de data e hora, desde formas simples até cálculos mais complexos
 - Módulo datetime
 - Fornece funções para manipulação de data e tempo
 - Embora funções aritméticas sejam suportadas o foco de sua implementação é uma eficiente forma de extração de atributos e formatação de saída
 - Tipos disponíveis: date, time, datetime, timedelta, tzinfo, timezone
 - Módulo time
 - Funções específicas para manipulação de tempo
 - Embora sempre disponível algumas funções deste módulo não funcionam da mesma forma em todas as plataformas, por isso é importante consultar a documentação da função para saber se existe alguma restrição à sua utilização
 - `time.asctime([t])` / `time.clock()` / `time.localtime([secs])`
 - Módulo calendar
 - Fornece funções utilitárias relacionadas à calendários, baseadas no calendário Gregoriano
 - `Calendar.iterweekdays()` / `Calendar.monthdatescalendar(year, month)`

Entrada e Saída – I/O

- O objeto *file* é responsável por operações de leitura e escrita (input/output) e também são conhecidos como *file-like* ou *streams*
- Dependendo da forma como é criado, o objeto *file* pode mediar escrita em tempo real em disco, outros tipos de armazenamentos ou dispositivos de comunicação como *buffers* em memória, *sockets*, *pipes*, etc
- *Files* são divididos em 3 categorias definidas no módulo *io*
 - raw binary files
 - buffered binary files
 - text files
- Python provê uma função embutida para denominada *open()* para criar ou acessar objetos do tipo *file*
 - `open(file, mode='r', buffering=-1, encoding=None, errors=None, newline=None, closefd=True, opener=None)`
 - Retorna o objeto *file* correspondente ou lança uma exceção do tipo *OSError* caso algum erro ocorra no acesso ao recurso solicitado

Entrada e Saída – I/O

- Pode se abrir um arquivo somente para leitura ou para leitura e escrita
 - `f = open('arquivo', 'r')` (default)
 - `f = open('arquivo', 'w')`
- Pode-se especificar o encoding do arquivo no terceiro parâmetro como por exemplo UTF-8
- Ao se abrir um arquivo que não existe no modo leitura, um erro é lançado (*FileNotFoundError*), caso seja especificado escrita, o arquivo será criado
- Um arquivo aberto, sempre deve ser fechado através do método *close()*
- Métodos para leitura de dados de um arquivo
 - `read()`
 - `readline()`
 - `readlines()`
- Métodos para escrita de dados em um arquivos
 - `write()`
 - `writelines()`
- Mais em <https://docs.python.org/3/library/io.html#module-io>

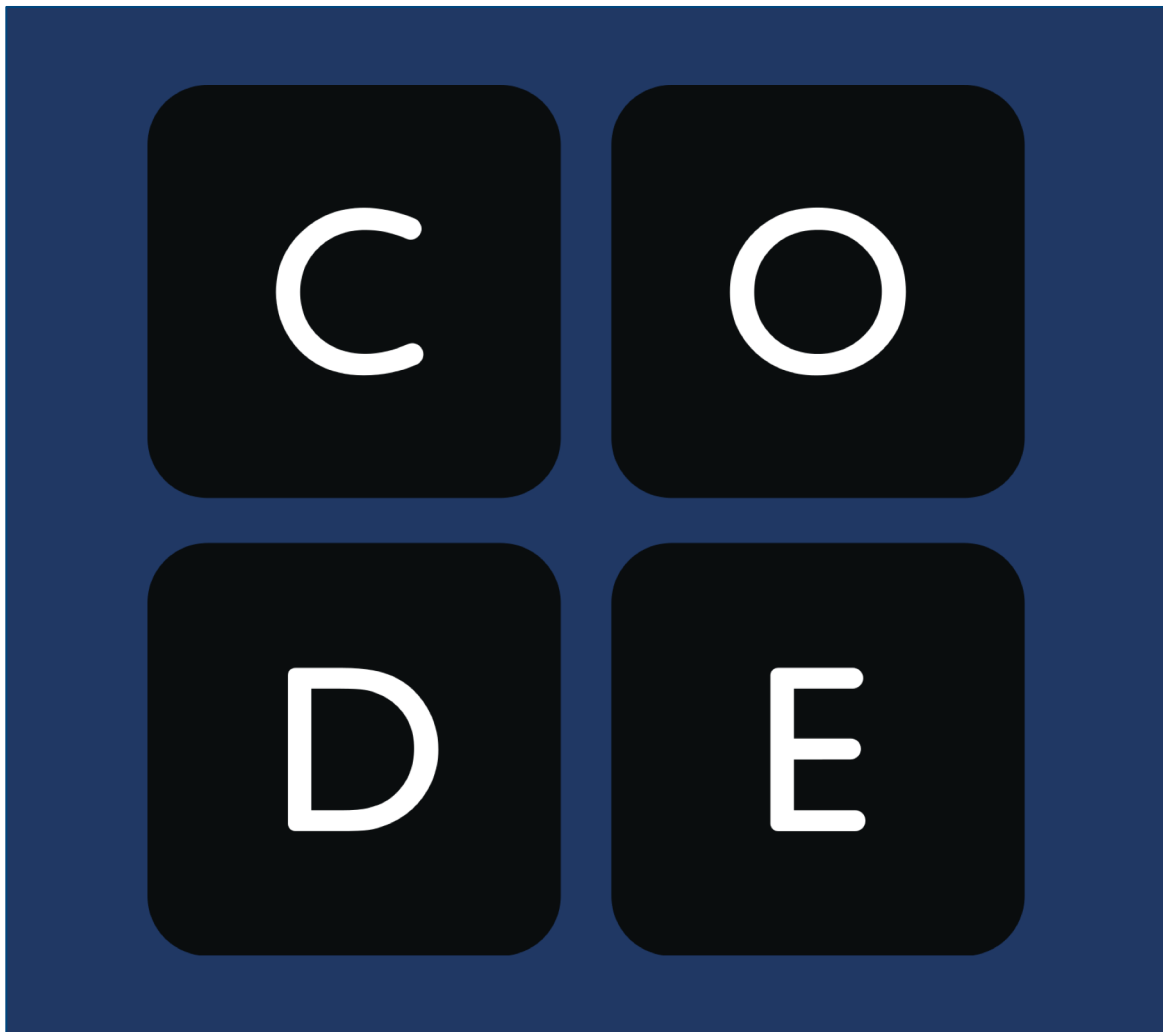
Erros e Exceções

- Há dois tipos de erros em Python
 - Syntax Errors
 - Exceptions
- Erros de sintaxe ou erros de parse são mais comuns quando se está aprendendo a linguagem
 - O interpretador irá informar a linha e uma seta, indicando onde o erro foi encontrado
- Exceções são eventos que ocorrem quando a sintaxe está correta, porém no momento da execução uma situação anormal ocorre
 - Divisão por zero
 - Conversão inválida de tipos
 - Arquivo não encontrado

Capturando e Tratando Exceções

- Python disponibiliza uma forma simples de capturar exceções através dos *statements* TRY e EXCEPT
- É permitido a criação de exceções definidas pelo usuário, herdando-se a classe Exception
- O *statement* FINALLY permite que operações denominadas *clean-up* sejam executadas independente da ocorrência de uma exceção
- O *statement* RAISE é utilizado para forçar o lançamento de uma exceção, muito utilizado principalmente para o tratamento adequado de exceções de negócio
- Todas as exceções embutidas no Python, estendem *BaseException*
- Mais em <https://docs.python.org/3/library/exceptions.html#builtin-exceptions>

Challenges Time



Exercicio 1

- Considere um arquivo texto que contem uma lista de arquivos. Cada linha desse arquivo, possui as seguintes informações:
 - Nome do criador do arquivo, 6 posições, alinhado à esquerda
 - Tamanho do arquivo no formato *human-readable size* (ex. 2K – 2 kilobytes, 3M – 3 megabytes, 5G – 5 gigabytes, ou o número total de bytes caso menor que 1K), com 7 posições, alinhado à direita sendo o caracter mais à direita um espaço em branco
 - Data de criação com 11 posições, no formato dd MMM YYYY (ex: 05 Jun 2005), acompanhado de um espaço em branco à direita
 - Nome do arquivo, até o final da linha
- Pede-se
 - Crie uma função que receba como parâmetro o nome do arquivo
 - A função deverá abrir o arquivo em modo somente leitura e retornar a **quantidade** de nomes dos criadores de arquivo únicos considerando que:
 - O tamanho máximo do arquivo em bytes, seja menor que $2 * (2^{20})$ (2 vezes 2 elevado à vigésima potencia)
 - A data de criação do arquivo seja anterior à 28 de Fevereiro de 2000.

Exercicio 2

- Considere um arquivo CSV (comma separated values) informações demográficas sobre super heróis
- As informações do arquivo são
 - id - Numero da linha
 - name - Nome
 - Gender - Gênero
 - Eye color - Cor dos olhos
 - Race - Raça
 - Hair color - Cor do cabelo
 - Height - Altura em centímetros
 - Publisher - Estúdio
 - Skin color - Cor da pele
 - Alignment - Vilão ou herói
 - Weight - Peso em libras

Exercicio 2 - continuação

- Pede-se
 - Crie uma função que efetue a leitura do arquivo e crie
 - Um arquivo de saída (.csv) com todos os heróis do gênero masculino (id, name)
 - Um arquivo de saída (.csv) com todos os heróis do gênero feminino (id, name)
 - Um arquivo de saída (.csv) com todos os heróis sem gênero definido (id, name)
 - Um arquivo de saída (.txt) com o nome de todos os heróis dos estúdios Marvel Comics e DC Comics (um nome por linha)
 - Um arquivo de saída (.txt) com o nome de todos os heróis que tenham cor de pele azul e que sejam vilões (uma única linha com os nomes separados por #)

Exercicio 3

- Considere um arquivo CSV (comma separated values) com os seguintes dados de todas as copas do mundo até o ano de 2014
- As informações do arquivo são
 - Year - Ano da copa
 - Country - Pais sede
 - Winner - Campeão
 - Runners-Up - Vice-Campeão
 - Third - Terceiro lugar
 - Fourth - Quarto Lugar
 - GoalsScored - Total de Gols
 - QualifiedTeams - Quantidade de Times
 - MatchesPlayed - Quantidade de Jogos
 - Attendance - Publico Total

Exercicio 3 - continuação

- Pede-se
 - Crie uma função que efetue a leitura do arquivo e crie um arquivo de saída com as seguintes informações (uma informação por linha):
 - Soma de público das copas com anos final 0 (1930, 1950, etc)
 - Quantidade total de gols entre as copas de 1954 e 1990, inclusive
 - Média de público
 - Média de gols por partida
 - Quantidade de vezes em que o país sede foi campeão
 - Quantidade de vezes em que o time do Brasil ficou entre uma das 4 primeiras posições
 - Ano das edições em que o time da França finalizou em terceiro lugar
 - Quantidade de vitórias por país, classificada em ordem crescente do número de títulos



Facens

AQUI TEM ENGENHARIA