



D01 - Treinamento Python-Django

Base Python 1

Resumo: Hoje embarcaremos em uma jornada para descobrir os fundamentos da sintaxe e da semântica do Python.

Conteúdo

I	Preâmbulo	2
II	Instruções	3
III	Regras específicas de hoje	4
IV	Exercício 00: minhas primeiras variáveis	5
V	Exercício 01: Números	6
VI	Exercício 02: Meu primeiro dicionário	7
VII	Exercício 03: Pesquisa de chaves	9
VIII	Exercício 04: Busca por valor	10
IX	Exercício 05: Busca por chave ou valor	11
X	Exercício 06: Ordenação do dicionário	12
XI	Exercício 07: Tabela periódica dos elementos	13

Capítulo I

Preâmbulo

O Zen de Python, de Tim Peters

Bonito é melhor que feio.
Explícito é melhor que implícito.
Simples é melhor que complexo.
Complexo é melhor do que complicado.
Plana é melhor que aninhada.
Esparso é melhor do que denso.
A legibilidade conta.
Casos especiais não são especiais o suficiente para quebrar as regras.
Embora a praticidade supere a pureza.
Erros nunca devem passar silenciosamente.
A menos que explicitamente silenciado.
Diante da ambiguidade, recuse a tentação de adivinhar.
Deve haver uma – e de preferência apenas uma – maneira óbvia de fazer isso.
Embora esse caminho possa não ser óbvio a princípio, a menos que você seja holandês.
Agora é melhor do que nunca.
Embora nunca seja frequentemente melhor do que *agora*.
Se a implementação for difícil de explicar, é uma má ideia.
Se a implementação for fácil de explicar, pode ser uma boa ideia.
Namespaces são uma ótima ideia – vamos fazer mais deles!



importar isso

Capítulo II

Instruções

A menos que haja uma contradição explícita, as instruções a seguir serão válidas para todos os dias deste Python Django Piscine.

- Somente esta página servirá como referência; não confie em boatos.
- Atenção! Este documento pode ser alterado até uma hora antes do envio.
- Esses exercícios são cuidadosamente organizados em ordem de dificuldade - do mais fácil ao mais difícil. Não levaremos em consideração um exercício mais difícil concluído com sucesso se um mais fácil não for perfeitamente funcional.
- Certifique-se de ter as permissões apropriadas em seus arquivos e diretórios.
- Você deve seguir os procedimentos de submissão para cada exercício.
- Seus exercícios serão verificados e avaliados por seus colegas de classe.
- Além disso, seus exercícios serão verificados e classificados por um programa chamado Moulinette. A Moulinette é muito metódica e rigorosa na avaliação do seu trabalho. É totalmente automatizado e não há como negociar com ele. Portanto, se você quiser evitar surpresas ruins, seja o mais minucioso possível.
- Exercícios em Shell devem ser executáveis com `/bin/sh`.
- Você não pode deixar nenhum arquivo adicional em seu diretório além daqueles especificados no tema.
- Tem uma questão? Pergunte ao seu colega à direita. Caso contrário, tente seu par à esquerda.
- Seu guia de referência se chama Google / man / the Internet /
- Lembre-se de discutir no fórum piscine do seu Intra e no Slack!
- Examine os exemplos minuciosamente. Eles poderiam muito bem pedir detalhes que não são explicitamente mencionados no assunto...

Capítulo III

Regras específicas de hoje


- Nenhum código no escopo global. Queremos funções!
- Cada arquivo entregue deve terminar com uma chamada de função em uma condição idêntica a:

```
se __name__ == '__main__':  
    your_function (qualquer que seja, parâmetro, é obrigatório)
```

- Você pode definir um gerenciamento de erro nesta condição.
- Nenhuma importação será autorizada, exceto aquelas explicitamente mencionadas em 'Funções autorizadas' na descrição de cada exercício.
- Você não terá que gerenciar as exceções geradas pela função open.
- Você terá que usar o interpretador python3.

Capítulo IV

Exercício 00: minhas primeiras variáveis

	Exercício 00
Exercício 00: minhas primeiras variáveis	
Diretório de entrega: ex00/	
Arquivos a serem entregues:	
var.py Funções permitidas: n/a	

Crie um script chamado var.py no qual você definirá uma função my_var. Nesta função, você declarará 9 variáveis de tipos diferentes e as imprimirá na saída padrão. Você reproduzirá esta saída exatamente:


```
$> python3 var.py 42
tem um tipo <class 'int'> 42 tem um
tipo <class 'str'> quarante-deux tem
um tipo <class 'str'> 42.0 tem um tipo <class 'float'>
True tem um tipo <class 'bool'> [42] tem um tipo
<class 'list'> {42: 42} tem um tipo <class 'dict'>
(42,) tem um tipo <class 'tuple'> set() tem um tipo
<class 'set'> $>
```

Claro, escrever explicitamente seus tipos de variáveis nas impressões de seu código é estritamente **proibido**. Não se esqueça de chamar sua função no final do seu script, conforme exigido pelas instruções:

```
se __name__ == '__main__':
    meu_onda()
```

Capítulo V

Exercício 01: Números

	Exercício 01
Exercício 01: Números	
Diretório de entrega: ex01/	
Arquivos a serem entregues:	
números.py Funções permitidas: n/a	


Para este exercício, você pode definir quantas funções quiser e nomeá-las como quiser.

O tarball d01.tar.gz no apêndice deste assunto contém uma subpasta ex01/ que contém um arquivo numbers.txt contendo os números de 1 a 100 separados por vírgula.

Projete um script Python chamado números.py cuja função é abrir um arquivo números.txt, ler os números que ele contém e exibi-los na saída padrão, um por linha, sem nenhuma vírgula.

Capítulo VI

Exercício 02: Meu primeiro dicionário

	Exercício 02
Exercício 02: Meu primeiro dicionário	
Diretório de entrega: ex02/	
Arquivos a serem entregues:	
var_to_dict.py Funções permitidas: NA	

Mais uma vez, você é livre para definir quantas funções quiser e nomeá-las como quiser Curti. Não repetiremos esta instrução, exceto se ela tiver que ser explicitamente contrariada.

Crie um script chamado var_to_dict.py no qual você copiará a seguinte lista de d casais como está em uma de suas funções:

```
d = [
    ('Hendrix', '1942'),
    ('Allman', '1946'),
    ('Rei', '1925'),
    ('Clapton', '1945'),
    ('Johnson', '1911'),
    ('Berry', '1926'),
    ('Vaughan', '1954'),
    ('Cooder', '1947'),
    ('Página', '1944'),
    ('Richards', '1943'),
    ('Hammett', '1962'),
    ('Cobain', '1967'),
    ('Garcia', '1942'),
    ('Beck', '1944'),
    ('Santana', '1947'),
    ('Ramone', '1948'),
    ('Branco', '1975'),
    ('Rustling', '1970'),
    ('Thompson', '1949'),
    ('Burton', '1939') ,
]
```


Seu script deve transformar essa variável em um dicionário. O ano será a chave, o nome do músico o valor. Ele deve então exibir este dicionário na saída padrão seguindo um formato claro:


```
1970: Frusciante  
1954: Vaughan  
1948: Ramone  
1944: Page Beck  
1911: Johnson  
...
```



A ordem final não deverá ser a mesma do exemplo. Este é um comportamento normal.
Você sabe por quê?

Capítulo VII

Exercício 03: Pesquisa de chaves

	Exercício 03
Exercício 03: Pesquisa de chaves	
Diretório de entrega: ex03/ Arquivos	
a serem entregues: capital_city.py Funções	
permitidas: import sys	

Aqui estão os dicionários que você deve copiar inalterados em uma das funções do seu script:

```
estados = {
    "Oregon" : "OU",
    "Alabama" : "AL",
    "Nova Jersey" : "NJ",
    "Colorado" : "CO"
}


capitais_cidades = {
    "OU" : "Salém",
    "AL" : "Montgomery",
    "NJ" : "Trenton",
    "O QUÊ" : "Denver"
}
```

Escreva um programa que receba um estado como argumento (ex: Oregon) e exiba sua capital (ex: Salem) na saída padrão. Se o argumento não der nenhum resultado, seu script deverá exibir: Estado desconhecido. Se não houver nenhum argumento - ou muitos - seu script não deve fazer nada e sair.

```
$> python3 capital_city.py Oregon Salem $>
python3 capital_city.py Ile-De-France Estado
desconhecido $> python3 capital_city.py $> python3
capital_city.py Oregon Alabama $> python3 capital_city.py
Oregon Alabama Ile-De-France $>
```

Capítulo 8

Exercício 04: Pesquisa por valor

	Exercício 04
Exercício 04: Pesquisa por valor	
Diretório de entrega: ex04/	
Arquivos a serem entregues:	
state.py Funções permitidas: import sys	


Você obtém os mesmos dicionários do exercício anterior. Você tem que copiá-los inalterado novamente em uma das funções do seu script.

Crie um programa que considere a capital como argumento e exiba o estado correspondente dessa vez. O restante dos comportamentos do seu programa deve permanecer o mesmo do exercício anterior.

```
$> python3 state.py Salem
Oregon $> python3 state.py
Paris Capital desconhecida $>
python3 state.py $>
```

Capítulo IX

Exercício 05: Pesquisa por chave ou valor

	Exercício 05
Exercício 05: Pesquisa por chave ou valor	
Diretório de entrega: ex05/	
Arquivos a serem entregues:	
all_in.py Funções permitidas: import sys	


Começando com os mesmos dicionários, você deve copiá-los inalterados novamente em uma de suas funções de script e escrever um programa que se comporte da seguinte maneira:

- O programa deve receber como argumento uma string contendo quantas expressões forem buscadas, separadas por vírgula.
- Para cada expressão nesta string, o programa deve detectar se é uma maiúscula, um estado ou nenhum deles.
- O programa não deve diferenciar maiúsculas de minúsculas. Não deve ocupar vários espaços em consideração também.
- Se não houver parâmetro ou muitos parâmetros, o programa não exibe nada.
- Quando há duas vírgulas sucessivas na string, o programa não exibe nada.
- O programa deve exibir os resultados separados por um retorno de carro e usar estritamente o seguinte formato:

```
$> python3 all_in.py "New jersey, Tren ton, NewJersey, Trenton, toto, Trenton é a capital de New Jersey Tren ton não é uma capital nem um estado NewJersey não é uma capital nem um estado Trenton é a capital de New Jersey toto não é uma capital nem um estado Salem é a capital do Oregon $"
```

Capítulo X

Exercício 06: Ordenação do dicionário

	Exercício 06
Exercício 06: Ordenação do dicionário	
Diretório de entrega: ex06/	
Arquivos a serem entregues:	
my_sort.py Funções permitidas: NA	


Integre este dicionário em qualquer uma das suas funções como:

```
d = {
    'Hendrix': '1942',
    'Allman': '1946',
    'Rei' : '1925',
    'Clapton': '1945',
    'Johnson': '1911',
    'Baga' : '1926',
    'Vaughan': '1954',
    'Cooder': '1947',
    'Página': '1944',
    'Richards': '1943',
    'Hammett': '1962',
    'Cobain': '1967', '1942', '1944',
    'Garcia': '1947',
    'Beck' : '1948',
    'Santana': '1975',
    'Ramone':
    'Branco' :
    'Rustling': '1970',
    'Thompson': '1949',
    'Burton': '1939',
}
```

Escreva um programa que exiba os nomes dos músicos classificados por ano em ordem crescente e, em seguida, em ordem alfabética para anos semelhantes. Um por linha, sem mostrar o ano.

Capítulo XI

Exercício 07: Tabela periódica dos elementos

	Exercício 07
Exercício 07: Tabela periódica dos elementos	
Diretório de entrega: ex07/	
Arquivos a serem entregues: periódico_table.py	
Funções permitidas: import sys	

O tarball d01.tar.gz no apêndice deste assunto contém a subpasta ex07/ na qual você encontrará o arquivo periodic_table.txt, que descreve uma tabela periódica dos elementos em um formato feito para programadores.

Crie um programa que usa o arquivo para escrever uma página HTML representando o periódico tabela dos elementos em um formato adequado.

- Cada elemento deve estar em uma 'caixa' de uma tabela HTML.
- O nome de um elemento deve estar em uma tag de título de nível 4.
- Os atributos de um elemento devem aparecer como uma lista. As listas devem conter pelo menos os números atômicos, o símbolo e a massa atômica.
- Você deve, pelo menos, respeitar o layout da Tabela de Mendeleiev conforme aparece no Google. Deve haver caixas vazias onde deveria haver, bem como retorno de carro onde deveria.

Seu programa deve criar o arquivo de resultado periodic_table.html. Obviamente, esse arquivo HTML deve ser legível em qualquer navegador e deve ser válido pelo W3C.

Você é livre para projetar seu programa como quiser. Não hesite em fragmentar seu código em funcionalidades específicas que você pode reutilizar. Você pode personalizar as tags com um estilo CSS "inline" para tornar sua entrega mais bonita (pense nas cores das bordas da tabela). Você também pode

gere o arquivo `periodic_table.css` se preferir.

Aqui está um trecho de um exemplo de saída que lhe dará uma ideia:

```
[...]
<tabela>
  <tr>
    <td style="border: 1px preto sólido; padding:10px">
      <h4>Hidrogênio</h4>
      <ul>
        <li>No 42</li>
        <li>H</li>
        <li>1,00794</li> <li>1
          elétron</li>
        <li>
      </li>
    </td>
  </tr>
[...]
```