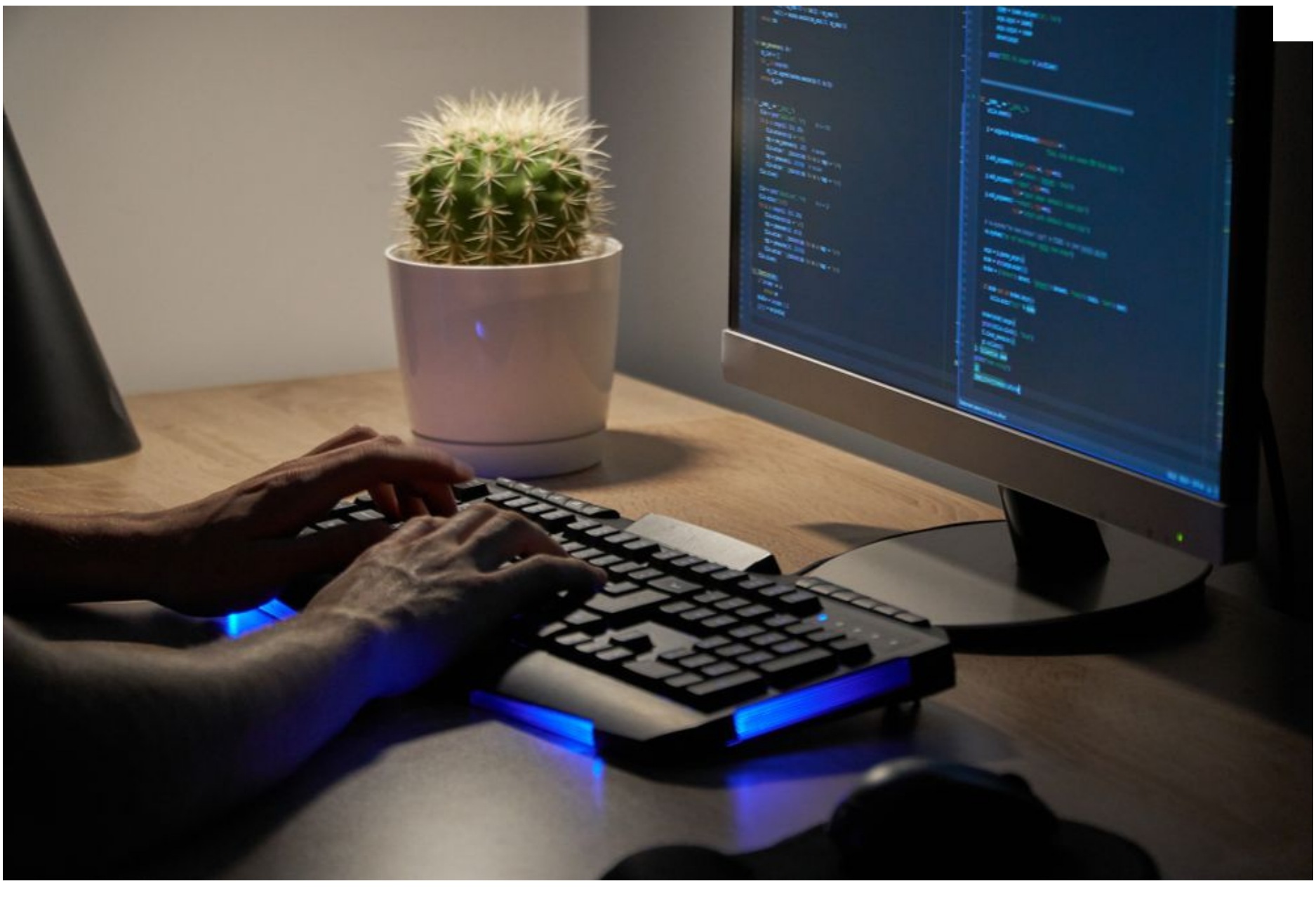
FOCO NO MERCADO DE TRABALHO

BIBLIOTECAS E MÓDULOS EM PYTHON

Vanessa Cadan Scheffer

BIBLIOTECA *REQUESTS*

A biblioteca requests habilita funcionalidades do procotolo HTTP, como o *get* e o *post*. Dentre seus métodos, o *get()* é o responsável por capturar informação da internet.



Fonte: Shutterstock.

Deseja ouvir este material?

Áudio disponível no material digital.

DESAFIO

No Brasil, existe um órgão responsável por gerar as estatísticas da atividade econômica no país. Para tal tarefa, as atividades são classificadas em grupos; por exemplo, as atividades do grupo 262 referem-se à fabricação de equipamentos de

informática e periféricos. "A CNAE, Classificação Nacional de Atividades Econômicas, é a classificação oficialmente adotada pelo Sistema Estatístico Nacional na produção de estatísticas por tipo de atividade econômica, e pela Administração Pública, na identificação da atividade econômica em cadastros e registros de pessoa jurídica." (API CNAE, 2017, [s.p.])

Como desenvolvedor em uma empresa de consultoria de software, você foi alocado em um projeto com base no qual o cliente deseja automatizar a extraçãc dos dados do CNAE e gerar um relatório. Os dados estão disponíveis no endereço https://servicodados.ibge.gov.br/api/v2/cnae/classes. Você deve extraí-los e gerar as seguintes informações:

- Quantas atividades distintas estão registradas?
- Quantos grupos de atividades existem?
- Quantas atividades estão cadastradas em cada grupo?
- Qual grupo ou quais grupos possuem o maior número de atividades vinculadas?

RESOLUÇÃO

Para automatizar o processo de extração dos dados do CNAE e gerar o relatório, vamos ter de usar bibliotecas. Para fazer a extração dos dados do CNAE, podemos usar a biblioteca *requests*. Para responder às perguntas, vamos precisar manipular listas e dicionários. Então vamos começar pela extração.

In [19]:

```
import requests


dados =
requests.get('https://servicodados.ibge.gov.br/api/v2/cnae/classes').json() #
resulta em uma lista de diconários
dados[0] # exibindo o primeiro registro de dados (primeiro dicionário da lista)
```

Out[19]:

```
{'id': '01113',
 'descricao': 'CULTIVO DE CEREAIS',
 'grupo': {'id': '011',
  'descricao': 'PRODUÇÃO DE LAVOURAS TEMPORÁRIAS',
  'divisao': {'id': '01',
   'descricao': 'AGRICULTURA, PECUÁRIA E SERVIÇOS RELACIONADOS',
   'secao': {'id': 'A',
    'descricao': 'AGRICULTURA, PECUÁRIA, PRODUÇÃO FLORESTAL, PESCA E
AQÜICULTURA'}}},
 'observacoes': ['Esta classe compreende - o cultivo de alpiste, arroz, aveia,
centeio, cevada, milho, milheto, painço, sorgo, trigo, trigo preto, triticale e
outros cereais não especificados anteriormente',
  'Esta classe compreende ainda - o beneficiamento de cereais em estabelecimento
agrícola, quando atividade complementar ao cultivo\r\n- a produção de sementes
de cereais, quando atividade complementar ao cultivo',
  'Esta classe NÃO compreende - a produção de sementes certificadas dos cereais
desta classe, inclusive modificadas geneticamente (01.41-5)\r\n- os serviços de
preparação de terreno, cultivo e colheita realizados sob contrato (01.61-0)\r\n-
o beneficiamento de cereais em estabelecimento agrícola realizado sob contrato
(01.63-6)\r\n- o processamento ou beneficiamento de cereais em estabelecimento
não-agrícola (grupo 10.4) e (grupo 10.6)\r\n- a produção de biocombustível
(19.32-2)']}
```

Agora que temos os dados guardados em uma lista de dicionários, podemos usar a função *built-in len()* para saber quantos elementos essa lista tem. Esse resultado será a quantidade de dicionários que representa a quantidade distintas de

atividades.

In [20]:

```
# Quantidade distintas de atividades, basta saber o tamanho da lista.


qtde_atividades_distintas = len(dados)
```

Ver anotações 0

Para saber quantos grupos de atividades existem e já começar a preparar os dad para os próximos passos, vamos criar uma lista que percorre cada registro e extrai a informação do grupo. Dado um registro, essa informação está na chave interna 'descricao' da chave externa 'grupo'. Logo, para acessar, temos que usar a sintaxe: *dicionario['chave_externa']['chave_interna']*. Na entrada 21, criamos uma lista vazia na linha 3 e, dentro da estrutura de repetição, vamos extraindo a informação e guardando-a na lista.

In [21]:

```
# Criar uma lista dos grupos de atividades, extraindo a descrição de cada
registro


grupos = []
for registro in dados:
    grupos.append(registro['grupo']['descricao'])


grupos[:10]
```

Out[21]:

```
['PRODUÇÃO DE LAVOURAS TEMPORÁRIAS',
 'PRODUÇÃO DE LAVOURAS TEMPORÁRIAS',
 'PRODUÇÃO DE LAVOURAS TEMPORÁRIAS',
 'PRODUÇÃO DE LAVOURAS TEMPORÁRIAS',
 'PRODUÇÃO DE LAVOURAS TEMPORÁRIAS',
 'PRODUÇÃO DE LAVOURAS TEMPORÁRIAS',
 'PRODUÇÃO DE LAVOURAS TEMPORÁRIAS',
 'PRODUÇÃO DE LAVOURAS TEMPORÁRIAS',
 'HORTICULTURA E FLORICULTURA',
 'HORTICULTURA E FLORICULTURA',
 'EXTRAÇÃO DE MINERAIS METÁLICOS NÃO-FERROSOS']
```

Ver anotações 0

Agora que temos uma lista com todos os grupos, para saber quantos grupos distintos existem, basta eliminar as duplicações e fazer a contagem. Na entrada 22, usamos o construtor *set()* para criar um conjunto de dados, sem repetições e sem alterar a lista com todos, uma vez que ainda vamos utilizá-la. O resultado do *set()* fazemos a contagem com a função *len()*, obtendo, então, a quantidade de grupos distintos.

In [22]:

```
# A partir da lista, podemos extrair a quantidade de grupos de atividades


qtde_grupos_distintos = len(set(grupos)) # o construtor set cria uma estrutura
de dados removendo as duplicações.
```

Agora vamos contar quantas atividades estão cadastradas em cada grupo. O código na entrada 23 faz esse trabalho. Usamos uma list comprehension para criar uma lista de tuplas. Cada tupla vai conter o grupo e a contagem de quantas vezes esse grupo aparece na lista de grupos: `(grupo, grupos.count(grupo)`. Isso será feito para cada grupo distinto: `for grupo in set(grupos)`.

In [23]:

```
# Resultado é uma lista de tuplas. Cria uma nova lista com o grupo e a
quantidade de atividades pertencentes a ele


grupos_count = [(grupo, grupos.count(grupo)) for grupo in set(grupos)]
grupos_count[:5]
```

Out[23]:

```
[('TECELAGEM, EXCETO MALHA', 3),
 ('COMÉRCIO ATACADISTA DE PRODUTOS DE CONSUMO NÃO-ALIMENTAR', 8),
 ('ATIVIDADES DE ORGANIZAÇÕES ASSOCIATIVAS PATRONAIS, EMPRESARIAIS E
PROFISSIONAIS',
  2),
 ('SEGURIDADE SOCIAL OBRIGATÓRIA', 1),
 ('FABRICAÇÃO DE ELETRODOMÉSTICOS', 2)]
```

Para sabermos qual grupo ou quais grupos possuem o maior número de atividades vinculadas, vamos transformar a lista de tuplas em um dicionário.

In [24]:

```
# Por conveniência, transformamos a lista em um dicionário


grupos_count = dict(grupos_count)
```

Agora podemos criar uma nova lista que contém todos os grupos que possuem a contagem com o mesmo valor da quantidade máxima que encontramos. Ao usar dicionário, conseguimos acessar a chave e o valor, o que facilita o trabalho.

In [25]:

```
# A partir do dicionário vamos descobrir qual (ou quais) grupos possuem mais
atividades


valor_maximo = max(grupos_count.values())
grupos_mais_atividades = [chave for (chave, valor) in grupos_count.items() if
valor == valor_maximo]
print(len(grupos_mais_atividades))
grupos_mais_atividades
```

```
1
```

Ver anotações 0

Out[25]:

```
['REPRESENTANTES COMERCIAIS E AGENTES DO COMÉRCIO, EXCETO DE VEÍCULOS
AUTOMOTORES E MOTOCICLETAS']
```

Para formalizar a entrega do componente de extração, que tal criar uma classe e um método com todo o código? Assim, quando for preciso extrair, basta instanciar a classe e invocar o método.

In [26]:

```
import requests


from datetime import datetime



class ETL:
    def __init__(self):
        self.url = None

    def extract_cnae_data(self, url):
        self.url = url
        data_extracao = datetime.today().strftime("%Y/%m/%d - %H:%M:%S")
        # Faz extração
        dados = requests.get(self.url).json()

        # Extrai os grupos dos registros
        grupos = []
        for registro in dados:
            grupos.append(registro['grupo']['descricao'])

        # Cria uma lista de tuplas (grupo, quantidade_atividades)
        grupos_count = [(grupo, grupos.count(grupo)) for grupo in set(grupos)]
        grupos_count = dict(grupos_count)  # transforma a lista em dicionário


        valor_maximo = max(grupos_count.values())  # Captura o valor máximo de
atividades
        # Gera uma lista com os grupos que possuem a quantidade máxima de
atividades
        grupos_mais_atividades = [chave for (chave, valor) in
grupos_count.items() if valor == valor_maximo]


        # informações
        qtde_atividades_distintas = len(dados)
        qtde_grupos_distintos = len(set(grupos))
```

Ver anotações 0

```
        print(f"Dados extraídos em: {data_extracao}")
        print(f"Quantidade de atividades distintas =
{qtde_atividades_distintas}")
        print(f"Quantidade de grupos distintos = {qtde_grupos_distintos}")
        print(f"Grupos com o maior número de atividades =
{grupos_mais_atividades}, atividades = {valor_maximo}")

        return None
```


In [27]:

```
# Usando a classe ETL


ETL().extract_cnae_data('https://servicodados.ibge.gov.br/api/v2/cnae/classes')
```

```
Dados extraídos em: 2020/06/04 - 19:09:39
Quantidade de atividades distintas = 673
Quantidade de grupos distintos = 285
Grupos com o maior número de atividades = ['REPRESENTANTES COMERCIAIS E AGENTES
DO COMÉRCIO, EXCETO DE VEÍCULOS AUTOMOTORES E MOTOCICLETAS'], atividades = 9
```

DESAFIO DA INTERNET

Ganhar habilidade em programação exige estudo e treino (muito treino). Acesse a biblioteca virtual no endereço http://biblioteca-virtual.com/ e busque pelo livro a seguir referenciado. Na página 198, no Capítulo 6 da referida obra, você encontrará o problema prático 6.9 (*Implemente a função adivinhe()*). Que tal tentar resolver esse desafio?

LJUBOMIR, P. **Introdução à computação usando Python**: um foco no desenvolvimento de aplicações. Rio de Janeiro: LTC, 2016.