UNIVERSIDADE UNIGRANRIO

GUILHERME BRANDÃO NABARRO

2556027

RYAN PORTO DO NASCIMENTO

2555498

APLICAÇÃO PRÁTICA DE PYTHON NA CIÊNCIA DOS DADOS

CURSO: ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

DISCIPLINA: CIÊNCIA DOS DADOS PARA TOMADA DE DECISÕES

DOCENTE: SERGIO RICARDO

NITERÓI – RJ, 2025

No nosso trabalho foi utilizado o prompt do Anaconda, instalado no Windows.

Exercício 1 (Etapa 1): Examinando o Anaconda e familiarizando-se com o Python

A etapa 1 é dividida em 5 partes:

1. Abra um Terminal, se estiver usando o macOS ou o Linux, ou uma janela de prompt de comando no Windows. Digite conda list na linha de comando.
2. Digite python no Terminal para abrir um interpretador Python de linha de comando.
3. Crie um loop for no prompt de comando para exibir os valores de 0 a 4 usando o código a seguir:

   for counter in range(5):

   ... print(counter)
   ...

4. Crie um dicionário de frutas (apples [maçãs], oranges [laranjas] e bananas [bananas]) usando o código a seguir:

   example_dict = {'apples':5, 'oranges':8, 'bananas':13}

5. Converta o dicionário em uma lista usando a função list(), como mostrado neste fragmento de código:

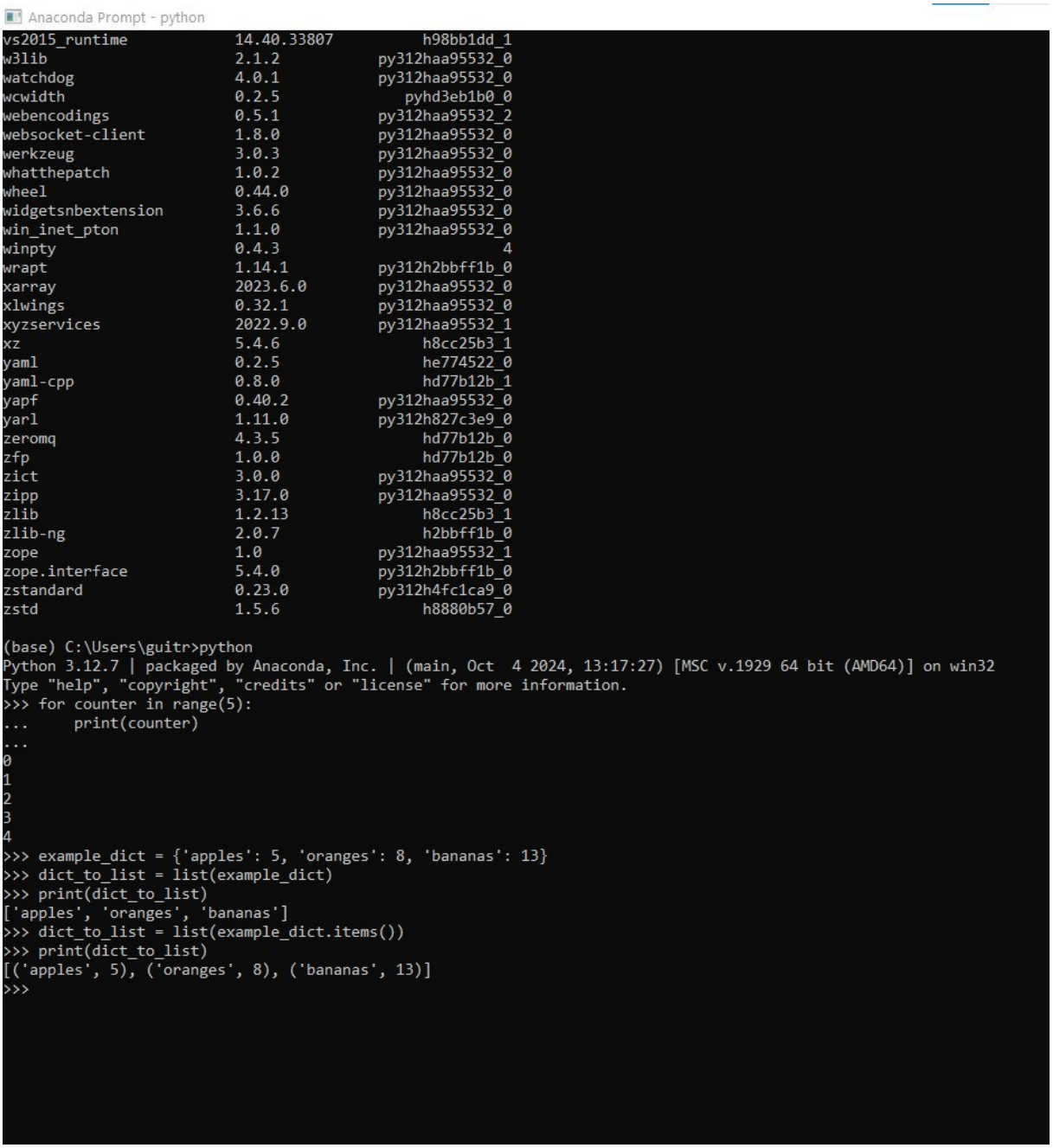
   dict_to_list = list(example_dict)

   dict_to_list

   E assim ficou o código após realizar as 5 etapas:

Anaconda Prompt - python

```
(base) C:\Users\guitr>conda list
# packages in environment at D:\Anaconda:
#
# Name                    Version                   Build  Channel
_anaconda_depends         2024.10            py312_mkl_0
aext-assistant            4.1.0           py312haa95532_jl4_0
aext-assistant-server     4.1.0           py312haa95532_0
aext-core                 4.1.0           py312haa95532_jl4_0
aext-core-server          4.1.0           py312haa95532_0
aext-panels               4.1.0           py312haa95532_0
aext-panels-server        4.1.0           py312haa95532_0
aext-project-filebrowser-server 4.1.0           py312haa95532_0
aext-share-notebook       4.1.0           py312haa95532_0
aext-share-notebook-server 4.1.0           py312haa95532_0
aext-shared               4.1.0           py312haa95532_0
aiobotocore               2.12.3          py312haa95532_0
aiohappyeyeballs          2.4.0           py312haa95532_0
aiohttp                   3.10.5          py312h827c3e9_0
aioitertools              0.7.1              pyhd3eb1b0_0
aiosignal                 1.2.0              pyhd3eb1b0_0
alabaster                 0.7.16          py312haa95532_0
alembic                   1.13.3          py312haa95532_0
altair                    5.0.1           py312haa95532_0
anaconda-anon-usage       0.4.4           py312hfc23b7f_100
anaconda-catalogs         0.2.0           py312haa95532_1
anaconda-cli-base         0.5.2           py312haa95532_0
anaconda-client           1.13.0          py312haa95532_1
anaconda-cloud-auth       0.7.2           py312haa95532_0
anaconda-navigator        2.6.3           py312haa95532_0
anaconda-project          0.11.1          py312haa95532_0
anaconda-toolbox          4.1.0           py312haa95532_0
anaconda_powershell_prompt 1.1.0               haa95532_0
anaconda_prompt           1.1.0               haa95532_0
annotated-types           0.6.0           py312haa95532_0
anyio                     4.2.0           py312haa95532_0
aom                       3.6.0               hd77b12b_0
appdirs                   1.4.4              pyhd3eb1b0_0
archspec                  0.2.3              pyhd3eb1b0_0
argon2-cffi               21.3.0             pyhd3eb1b0_0
argon2-cffi-bindings      21.2.0          py312h2bbff1b_0
arrow                     1.2.3           py312haa95532_1
arrow-cpp                 16.1.0              h7cd61ee_0
astroid                   2.14.2          py312haa95532_0
astropy                   6.1.3           py312h827c3e9_0
astropy-iers-data         0.2024.9.2.0.33.23 py312haa95532_0
asttokens                 2.0.5              pyhd3eb1b0_0
async-lru                 2.0.4           py312haa95532_0
atomicwrites              1.4.0                       py_0
attrs                     23.1.0          py312haa95532_0
automat                   20.2.0                      py_0
autopep8                  2.0.4              pyhd3eb1b0_0
aws-c-auth                0.6.19              h2bbff1b_0
aws-c-cal                 0.5.20              h2bbff1b_0
aws-c-common              0.8.5               h2bbff1b_0
aws-c-compression         0.2.16              h2bbff1b_0
aws-c-event-stream        0.2.15              hd77b12b_0
aws-c-http                0.6.25              h2bbff1b_0
aws-c-io                  0.13.10             h2bbff1b_0
```


Anaconda Prompt - python
vs2015_runtime 14.40.33807 h98bb1dd_1
w3lib 2.1.2 py312haa95532_0
watchdog 4.0.1 py312haa95532_0
wcwidth 0.2.5 pyhd3eb1b0_0
webencodings 0.5.1 py312haa95532_2
websocket-client 1.8.0 py312haa95532_0
werkzeug 3.0.3 py312haa95532_0
whatthepatch 1.0.2 py312haa95532_0
wheel 0.44.0 py312haa95532_0
widgetsnbextension 3.6.6 py312haa95532_0
win_inet_pton 1.1.0 py312haa95532_0
winpty 0.4.3 4
wrapt 1.14.1 py312h2bbff1b_0
xarray 2023.6.0 py312haa95532_0
xlwings 0.32.1 py312haa95532_0
xyzservices 2022.9.0 py312haa95532_1
xz 5.4.6 h8cc25b3_1
yaml 0.2.5 he774522_0
yaml-cpp 0.8.0 hd77b12b_1
yapf 0.40.2 py312haa95532_0
yarl 1.11.0 py312h827c3e9_0
zeromq 4.3.5 hd77b12b_0
zfp 1.0.0 hd77b12b_0
zict 3.0.0 py312haa95532_0
zipp 3.17.0 py312haa95532_0
zlib 1.2.13 h8cc25b3_1
zlib-ng 2.0.7 h2bbff1b_0
zope 1.0 py312haa95532_1
zope.interface 5.4.0 py312h2bbff1b_0
zstandard 0.23.0 py312h4fc1ca9_0
zstd 1.5.6 h8880b57_0
(base) C:\Users\guitr>python
Python 3.12.7 | packaged by Anaconda, Inc. | (main, Oct 4 2024, 13:17:27) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> for counter in range(5):
... print(counter)
...
0
1
2
3
4
>>> example_dict = {'apples': 5, 'oranges': 8, 'bananas': 13}
>>> dict_to_list = list(example_dict)
>>> print(dict_to_list)
['apples', 'oranges', 'bananas']
>>> dict_to_list = list(example_dict.items())
>>> print(dict_to_list)
[('apples', 5), ('oranges', 8), ('bananas', 13)]
>>>

Exercício 2 (Etapa 2): Carregando os dados do estudo de caso com o Jupyter e o pandas.

O exercício 2 foi enviado por um link Html através do Github, com todo o passo a passo existente no trabalho. Tais como:

1. Abra um Terminal (macOS ou Linux) ou uma janela de prompt de comando (Windows) e digite jupyter notebook. Você verá a interface do Jupyter em seu navegador web. Se ele não abrir automaticamente, copie e cole a URL do terminal para o navegador. Nessa interface, você poderá navegar em seus diretórios começando por aquele em que estava quando iniciou o servidor do notebook.

2. Navegue para um local conveniente para armazenar os materiais deste livro e crie um novo notebook Python 3 a partir do menu New, como mostrado na Figura abaixo:

3. Faça com que sua primeira célula seja do tipo markdown digitando *m* no modo de comando (pressione Esc para entrar no modo de comando) e digite o símbolo de

número, #, no começo da primeira linha, seguido por um espaço, para o cabeçalho. Crie um título para seu notebook (Jupyter notebook_nome aluno). Nas próximas linhas, insira uma descrição (O que está achando da atividade até aqui). Na Figura a seguir, temos um screenshot de um exemplo, incluindo outros tipos de markdown como negrito, itálico e a maneira de escrever texto no estilo código em uma célula markdown:

```
# First Jupyter notebook
Welcome to your first jupyter notebook! The first thing to know about Jupyter notebooks is
that there are two kinds of cells. This is a markdown cell.

There are a lot of different ways to mark up the text in markdown cells, including __bold__
and *italics*.

The next one will be a `code` cell.
```

4. Pressione *Shift* + *Enter* para renderizar a célula markdown. Isso também deve criar uma nova célula, que será uma célula de código. Você pode alterá-la para uma célula markdown, já que agora sabe como fazê-lo, pressionando *m*, e transformá-la novamente em uma célula de código pressionando *y*. Sabemos que é uma célula de código porque estamos vendo *In [ ]:* próximo a ela.

5. Digite *import pandas as pd* na nova célula, como mostrado no screenshot a seguir:

Quando você executar essa célula, a biblioteca pandas será carregada em seu ambiente de computação. É comum a importação com “as” para que seja criado um alias para a biblioteca. Agora usaremos o pandas para carregar o arquivo de dados. Ele está no formato do Microsoft Excel, logo, podemos usar *pd.read_excel*.

6. Importe o dataset, que está no formato do Excel, como um DataFrame usando o método pd.read_excel(), como mostrado neste fragmento:

df = pd.read_excel('../Data/default_of_credit_card_clients_courseware_version_1_21_19.xls')

Observe que você precisa apontar o leitor do Excel para o local onde o arquivo está localizado. Se ele estiver no mesmo diretório de seu notebook, é possível inserir apenas o nome do arquivo. O método *pd.read_excel* carregará o arquivo do Excel em um *DataFrame*, que chamamos de *df*. Agora o poder do pandas está disponível para nós.

O problema da empresa

Nosso cliente é uma empresa de cartão de crédito. Eles nos trouxeram um *dataset* que inclui dados demográficos e dados financeiros recentes (últimos seis meses) de uma amostra de 30.000 titulares de contas. Esses dados estão no nível de conta de crédito; em outras palavras, há uma linha para cada conta (você deve sempre esclarecer qual é a definição de linha, em um *dataset*). As linhas são rotuladas de acordo com se no mês seguinte ao período de dados histórico de seis meses um proprietário de conta ficou inadimplente, ou seja, não fez o pagamento mínimo.

Etapas da exploração de dados

Agora que conhecemos o problema da empresa e temos uma ideia do que os dados devem conter, podemos comparar essas impressões com o que estamos vendo realmente nos dados. Nossa função ao explorar dados é percorrê-los tanto diretamente quanto usando resumos numéricos e gráficos e considerar criticamente se fazem sentido e correspondem ao que nos foi dito sobre eles.

Nessa etapa do trabalho de exploração de dados, vamos responder os seguintes questionamentos: (Use comandos no Python para responder).

1. Saber quantas colunas os dados contêm. (Podem ser de características, resposta ou metadados).

2. Quantas linhas (amostras).

3. Que tipos de características existem. Quais são categóricas e quais são numéricas? (Os valores das características categóricas pertencem a classes discretas como “sim”, “não” ou “talvez”. Normalmente as características numéricas pertencem a uma escala numérica contínua, como as quantias em dólares).

4. Qual é a aparência dos dados segundo essas características. (Para saber isso, você pode examinar o intervalo de valores em características numéricas ou a frequência de classes diferentes em características categóricas, por exemplo).

5. Existem dados faltando?