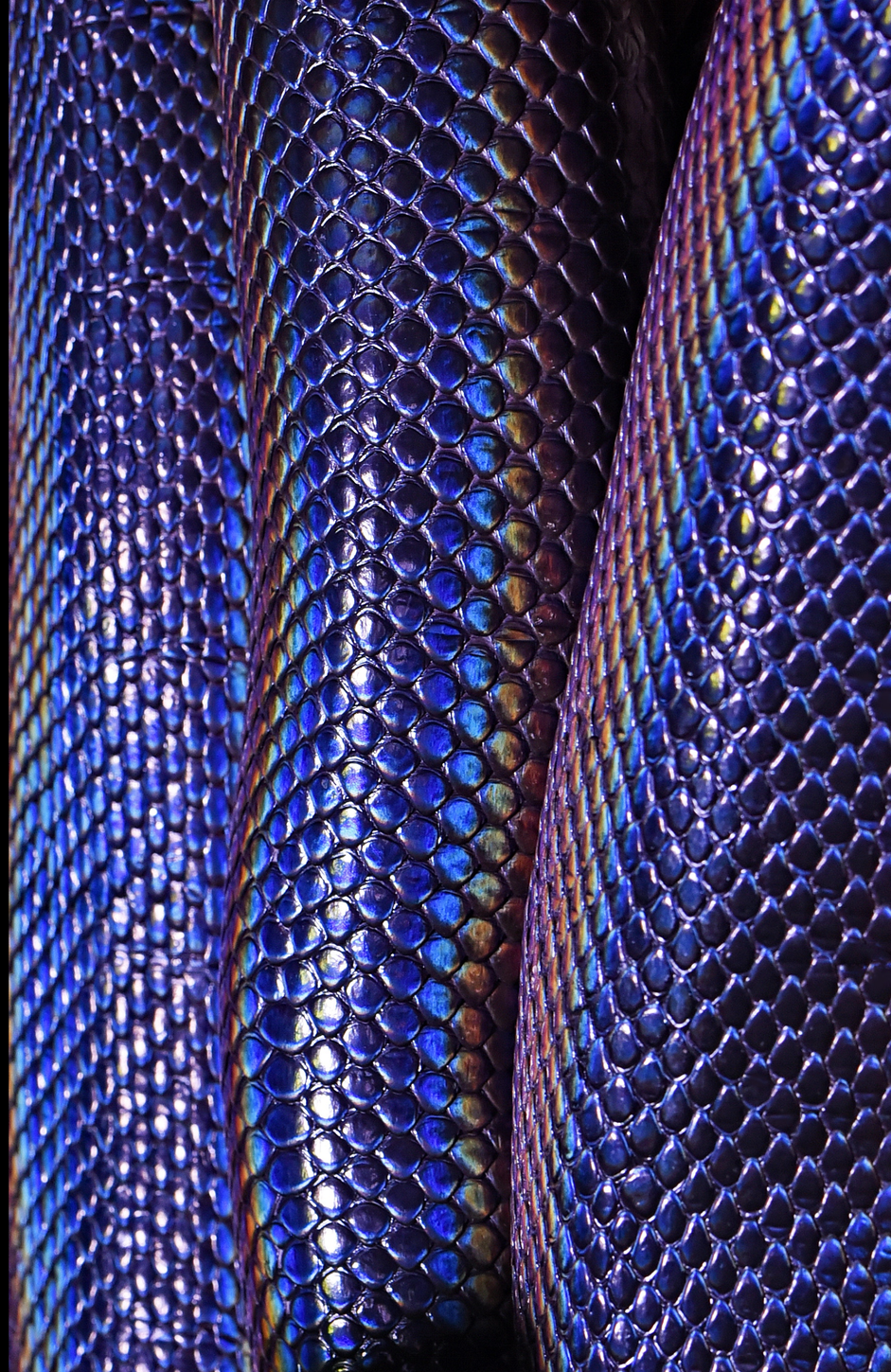
Introdução a Python para análise de dados

Parte 1: a linguagem

Thiago Cardoso
thiago.figueredo@cesar.org.br



Conteúdo

Índice de exercícios

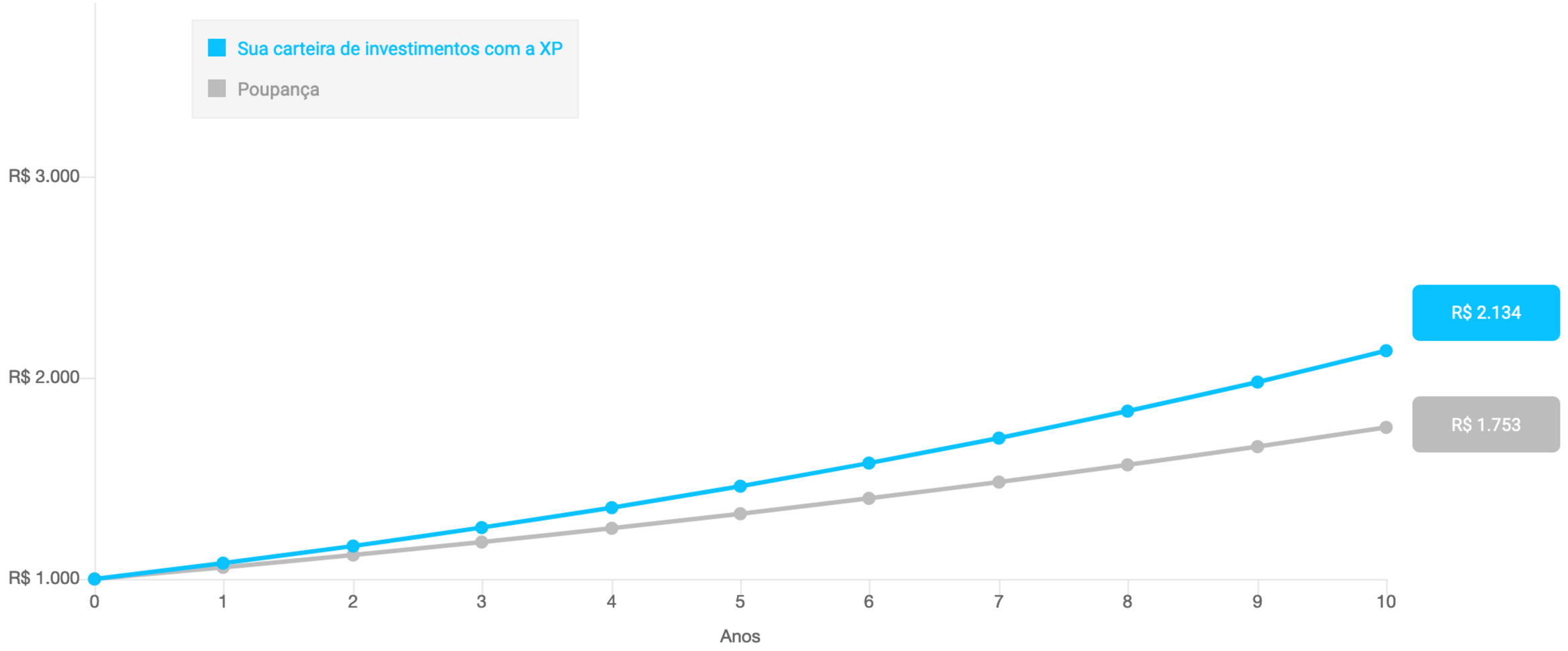
“But how does Spotify actually use that concept in practice to calculate millions of users’ suggested tracks based on millions of other users’ preferences?

With matrix math, done with Python libraries!”

How Does Spotify Know You So Well
https://medium.com/s/story/spotifys-discover-weekly-how-machine-learning-finds-your-new-music-19a41ab76efe

“Python is a widely used high-level, general-purpose, interpreted, dynamic programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C++ or Java. The language provides constructs intended to enable clear programs on both a small and large scale.”

Problema



Para cálculo da rentabilidade da carteira XP levou-se em consideração um retorno médio de 105% do CDI, considerando CDI de 7,5% ao ano. Para o cálculo de rentabilidade da poupança levou-se em consideração o retorno no aniversário do dia 13/11/2017 (0,469%). AS INFORMAÇÕES PRESENTES NESTE MATERIAL SÃO BASEADAS EM SIMULAÇÕES E OS RESULTADOS REAIS PODERÃO SER SIGNIFICAMENTE DIFERENTES.

Cálculo simples

Instalação

ou

Oficial
Propósito geral

Foco em data science/ML
Pacotes pré-instalados
Gerenciamento de pacotes

Interpretador

```
$ python
>>> 10 + 2
12
>>>
```

(CTRL + D para sair)

Operadores aritméticos

Operador	Significado	Exemplo
+	Soma de dois elementos	5 + 2
-	Subtração de dois elementos	5 - 2
*	Multiplicação de dois elementos	5 * 2
/	Divisão de dois elementos	5 / 2
%	Módulo	5 % 2
* *	Exponenciação	5 ** 2
/ /	Divisão com arredondamento	5 // 2

Exercício 1

Considerando um rendimento mensal de **0,469%** da poupança mostrada no problema, qual será o valor acumulado em **10 anos** para um investimento inicial de **R$1000**?

Parametrizando o cálculo

Variáveis

Armazenamento temporário de informação

Identificador iniciado por:

- a-z
- A-Z
- _

Seguido por, adicionalmente, 0-9

```
>>> inicial = 1000.0
>>> inicial * 1.1
1100.0
>>>
```

Exercício 2

Qual seria o valor acumulado em **7, 8 e 9 anos**? E se eu investir **R$2.000**, **R$10.000** e **R$100.000**?

Lembrete: rendimento de 0,469% ao mês

Sugestão: use variáveis para os valores que mudam

Funções

Recebe argumentos e retorna um valor e/o causa um efeito.

Sintaxe:

```
def nome(arg1, arg2, …):
    operação1
    operação2
    …
    return resultado
```

```
>>> def incr(value):
...     return value + 1
...
>>> incr(11)
12
>>>
```

Recuo (indentação)

Diferenciação de blocos de código é feita por recuos.

```
def incr(value):
__return value + 1
```

```
>>> def incr(value):
... return value + 1
...
  File "<stdin>", line 2
    return value + 1
           ^
IndentationError: expected
an indented block
>>>
```

Documentação

Sintaxe:

```
def nome(arg1, arg2, …):
    '''
    Documentação da função
    '''
```

Pedindo ajuda:

```
help(nome)
```

```
>>> def incr(value):
...     '''
...     Increments a value
...     '''
...     return value + 1
...
>>> help(incr)
```

Comentários

Sintaxe:

 # Comentário

```
>>> def incr(value):
...     '''
...     Increments a value
...     '''
...     # Soma 1 a value
...     return value + 1
...
>>>
```

Exercício 3

Qual seria o valor acumulado em **7, 8 e 9 anos**? E se eu investir **R$2.000**, **R$10.000** e **R$100.000**? Qual a diferença entre poupança e o investimento atrelado ao CDI?

Lembrete:

- Poupança: rendimento de 0,469% ao mês
- CDI: retorno de 105% do CDI, que rende 7,5% ao ano

Sugestão: use funções para cada investimento

Interagindo com o usuário

Scripts

Arquivos de extensão .py onde se coloca qualquer código Python.

Executados com o interpretador:

```
>>> python <nome>.py
```

incr.py

```
def incr(value):
  '''
  Increments a value
  ,'''
  return value + 1


incr(10)
```

Scripts

O que acontece ao executar o script ao lado?

incr.py

```
def incr(value):
  '''
  Increments a value
  ,''
  return value + 1


incr(10)
```

Strings e print()

Imprime uma **string**

Sintaxe:

'texto'

“texto"

```
>>> print(‘Hi’)
Hi
>>>
```

Strings e print()

Imprime uma **string**

Sintaxe:

'texto'

"texto"

incr.py

```
def incr(value):
    '''
    Increments a value
    '''
    return value + 1


print(incr(10))
```

Concatenação e formatação de strings

Concatenação é feita com operador +

Apenas strings são aceitas

```
>>> ‘Resultado é ‘ + 1000

Traceback (most recent call
last):

  File "<stdin>", line 1, in
<module>

TypeError: can only concatenate
str (not "int") to str

>>> ‘Resultado é ‘ + str(1000)

‘Resultado é 1000
```

Concatenação e formatação de strings

Método `format()`

Template:

- String
- Contém blocos {} onde se deve inserir valores
- {n}: posicional
- {nome}: por nome
- {n/nome:.x}: precisão de x casas
- {n/nome:f}: exibir como número real
- {n/nome:%}: exibir como porcentagem

Cheat sheet

```
>>> t = ‘Resultado é {0}'
>>> t.format(1000)
Resultado é 1000
>>> t = ‘Resultado é {valor}'
>>> t.format(valor=1000)
Resultado é 1000
>>> t = ‘Rendimento de {0:.2%}’
>>> t.format(0.1)
Rendimento de 10.00%
```

Exercício 4

Altere o script para imprimir uma mensagem ao usuário dizendo o valor acumulado em 10 anos para a poupança?

Lembrete:

- Poupança: rendimento de 0,469% ao mês

Sugestão: use formatação de strings para gerar a mensagem

input()

Faz um pedido de entrada ao usuário

```
>>> input(‘Qual é o seu nome? ’)

Qual é o seu nome? Thiago

Thiago

>>>
```

Exercício 5

Altere o script para que o usuário informe:

- nome
- montante inicial
- tempo de investimento

Então imprima uma mensagem contendo o nome e o resultado para os dois investimentos.

Lembrete:

- Poupança: rendimento de 0,469% ao mês
- CDI: retorno de 105% do CDI, que rende 7,5% ao ano

Tipos de dados básicos

Tipo	Significado	Exemplo
`int`	Número inteiro	5
`float`	Número real	5.47
`bool`	Valor booleano	True ou False
`str`	String	'Hello'
`list`	Listas	[1, 2]
`dict`	Dicionários	{ ‘Zé’: 10.0 }
`tuple`	Tuplas	(1,2)

Inspeção de tipo

type()

```
>>> nome = input(‘Qual é o seu nome? ’)
>>> type(nome)
<class ‘str'>
>>> type(True)
<class ‘bool’>
>>> type(1.0)
<class 'float'>
```

Conversão de tipo

Uso de função com o nome do tipo:

```
int()

float()

str()

bool()

…
```

```
>>> str(1000.01)
‘1000.01'
>>> float(‘1000.01’)
1000.01
>>> bool('True')
True
>>>
```

Conversão de tipo

Falha na conversão gera erro

```
>>> float('Thiago')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: could not convert string to float: 'Thiago'
>>>
```

Tratamento de erro

```
try:
   operação
except:
except exceção:
except exceção as apelido:
   tratamento
```

```
try:
   float('Thiago')
except:
   print('Erro na conversão')
```

```
try:
   float('Thiago')
except ValueError as e:
   print(e)
```

Condicional

```
if condição:
  …
elif condição:
  …
elif condição:
  …
else:
  …
```

```
>>> if type(x) is str:
...   print(‘A string’)
... elif type(x) is int:
...   print(‘A boolean’)
... else:
...   print("Don’t know”)
...
>>>
```

Operadores relacionais

Operador	Significado	Exemplo
==	Testa igualdade	x == y
!=	Testa desigualdade	x != y
>	Testa se é maior que	x > y
>=	Testa se é maior que ou igual a	x >= y
<	Testa se é menor que	x < y
<=	Testa se é menor que ou igual a	x <= y

Operadores de identidade/pertencimento

Operador	Significado	Exemplo
`is`	Testa identidade	x is None
`is not`	Negação do anterior	x is not None
`in`	Testa pertencimento	x in y
`not in`	Negação do anterior	x not in y

Operadores lógicos

Operador	Significado	Exemplo
`and`	Ambas condições são verdadeiras	x > 4 and y < 2
`or`	Alguma das condições é verdadeira	x > 4 or y < 2
`not`	A condição não é verdadeira	not (x > 4 and y < 2)

Estruturas de repetição (1/2)

```
while condição:
   …
else:
   # executa ao sair do loop
   …
```

```
>>> while x < 2:
...     print(x)
...     x = x + 1
...
>>>
```

Exercício 6

Altere o script para validar que o usuário está digitando valores válidos:

- nome deve ser não numérico
- montante inicial deve ser número
- tempo de investimento deve ser número

Mantenha o script perguntando enquanto o usuário não digita o valor correto

Dica: há métodos de string para verificar se o conteúdo é numérico

Gerando séries

Listas

Sintaxe:

`[e1, e2, e3, …, en]`

Operações:

`append()/pop()`

`insert()/remove()`

`lista[indice]`

`lista[start:stop:step]`

```
>>> l = [10, 11, 12, 13]
>>> l.append(14)
>>> l
[10, 11, 12, 13, 14]
>>> l.pop()
14
>>> l[1]
11
>>> l[-2]
12
>>> l[1:3]
[11,12]
```

Estruturas de repetição (2/2)

```
for variável in iterador:
   …
else:
   # executa ao sair do loop
  …
```

Gerando iteradores:

```
range(stop)
range(start, stop, step)
```

```
>>> for x in range(3):
...   print(x)
...
0
1
2
>>> for x in range(2, -1, -1)
...   print(x)
...
2
1
0
```

Exercício 7

Altere o script para calcular o valor acumulado a cada ano até o ano informado pelo usuário.

Lists comprehension

Outra forma de declarar uma lista

Sintaxe:

`[`*`operacao`* `for` *`var`* `in` *`iterador`*`]`

```
>>> [x for x in range(3)]
[0, 1, 2]
>>> [incr(x) for x in range(3)]
[1, 2, 3]
>>> [x + y for x in range(3) for y in range(2)]
[0, 1, 1, 2, 2, 3]
```

Exercício 8

Reescreva o script anterior usando *list comprehension*.
Ficaria mais simples?

Gerando resultados

Arquivos

`open(`*`filename`*`, `*`mode`*`)`

`file.close()`

r	Leitura (padrão)
w	Escrita (sobreescreve arquivo)
a	Escrita (a partir do final)
r	Leitura e escrita

Adicionar `b` para ler como binário. Ex: .: `‘rb'`

```
>>> f = open(‘arquivo.txt’, ‘w’)

>>> f.close()

>>> f = open(‘arquivo.bin’,
‘rb')
```

Arquivos

`file.read(`*`size`*`)`

`file.readline()`

`file.readlines()`

`file.write(`*`string`*`)`

```
>>> f.readlines()

['Lorem ipsum\n', 'Lorem
ipsum\n', …]

>>>
```

Exercício 9

Altere o script para ao invés de imprimir o resultado na tela, salvar em um arquivo no formato CSV (comma separated values).
Cada linha deverá conter o nome do investimento como primeiro elemento, seguido dos resultados para cada ano.

CSV:

- Tabela
- Cada do arquivo representa uma linha da tabela
- Colunas são delimitadas por “,”.
- Exemplo de CSV com duas linhas e quatro colunas:

1,2,3,4
1,2,4,5

Dicas:

- Ver método `join()` de string
- Representação de nova linha em string é “\n”. Ex.:

```
'Primeira linha\nSegunda linha'
```

Pacotes

Instalação

`pip install *nome*`

`conda install *nome*`

Uso

`import *nome*`

`import *nome* as *apelido*`

```
>>> import matplotlib.pyplot as
plt

>>> plt.plot([1,2,3])
```

matplotlib

Básico

```
plt.plot(pontos)
plt.show()
```

```
>>> import matplotlib.pyplot as
plt

>>> plt.plot([x for x in
range(10)])

>>> plt.show()
```

matplotlib

Título e texto dos eixos

```
plt.title(titulo)
plt.xlabel(texto_x)
plt.ylabel(texto_y)
```

```
>>> import matplotlib.pyplot as plt
>>> plt.title('Linear')
>>> plt.xlabel('Eixo x')
>>> plt.ylabel('Eixo y')
>>> plt.plot([x for x in range(10)])
>>> plt.show()
```

matplotlib

Legendas

```
plt.plot(pontos,
label='legenda')

plt.legend()
```

```
>>> import matplotlib.pyplot as
plt

>>> plt.title('Linear')

>>> plt.xlabel('Eixo x')

>>> plt.ylabel('Eixo y')

>>> plt.plot([x for x in
range(10)], label='Pontos')

>>> plt.legend()

>>> plt.show()
```

Exercício 10

Crie um **novo** script que lê o arquivo CSV e plota um gráfico com todas as linhas (cada linha do CSV, será uma linha no gráfico). Use o nome do investimento como legenda.

Dicas:

- plt.plot() pode ser chamado várias vezes
- Ver método `split()` de string
- Relembre conversão de tipos

Dicionários

Sintaxe:

```
{
  chave: valor,
  chave: valor
}
```

Operações:

```
items()/keys()/values()
get()/[chave]
pop(chave)
chave in dicionario
```

```
>>> d = {}
>>> d[‘Ze’] = 9.4
>>> d[‘Maria’] = 9.8
>>> d.get(‘Maria’)
9.8
>>> d[‘Maria’]
9.8
>>> ‘Ze' in d
True
```

Exercício 11

Altere o novo script para pedir o nome do investimento que o usuário desejar plotar e plote apenas este.

Sugestão:

- Use dicionários para guardar as linhas do CSV

Referências e material complementar

string — Common string operations

7. Input and Output

Python Format Strings Cheat Sheet