Estruturas de Dados Nativas do Python

Professor: Alex Pereira

Como aumentar o risco dos seus desafios

- Aulas e disciplinas
- Motivação
  - Individual;
  - Social.

Procrastinação (Tim Urban): https://youtu.be/9DbYXR4k18k?t=71

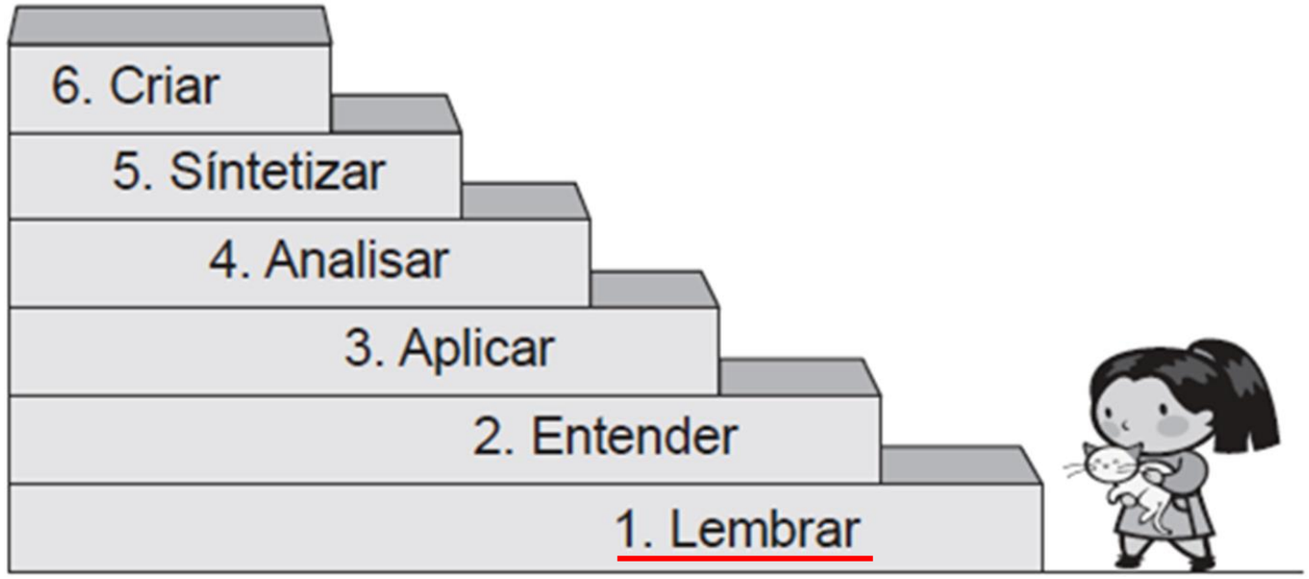
Professor: Alex Pereira

Revisão

- Sentença composta e Indentação
  - pra que serve a indentação?
- Retorno de uma função
  - Explícito
  - Implícito
    - ✓ Qual o retorno?
- Argumento de uma função
  - Pra que serve?
- Como testar sua função no jupyter Google Colab?
- Validar somente depois de
  - se convencer que chegou numa solução

Conceitos de Python abordados na aula

- def
- If
- else
- while
- for
- in
- break
- return
- import
- from



Vantagens do Python

- Extenso suporte de bibliotecas
  - NumPy, Pandas, Flask e etc.
- Open source e desenvolvido em comunidade
- Fácil de aprender
- Estruturas de dados amigáveis
- Linguagem tipada dinamicamente
  - não é necessário definir previamente o tipo de dados com base no valor atribuído

Vantagens do Python

- Linguagem orientada a objetos
- Possui um console interativo
- Portável em diversos sistemas operacionais
- O código Python é significativamente menor que o código C ++ / Java equivalente.
  - Isso implica que há menos para digitar, depurar e manter.

Desvantagens do Python

- Velocidade / Desempenho computacional
  - Se precisar de alto desempenho, dê preferência para C ou C++.
    - ✓ Velocidade de execução pode não ser tão importante quanto velocidade de desenvolvimento*.
- Consumo de memória:
  - Python consome mais memória do que outras linguagens como C ou C++.
- Desenvolvimento Mobile, aplicativos embarcados e IoT:
  - Java e outras linguagens oferecem mais suporte para desenvolvimento de aplicativos de celular e IoT.

Atividade Extra para Alunos Experientes

- Resolva alguma issue de alguma biblioteca opensource
  - Usando IA Generativa

Ordenando uma lista

- O método sorted (aceita qualquer objeto iterável)
  - retorna uma cópia lista com seus elementos ordenados

```
sorted([4,1,5,6,9])
```

```
[1, 4, 5, 6, 9]
```

- list.sort() (built-in method, modifies in-place)
  - ordena a propria lista em que o método foi executado, e retorna **None**

```
al = [2, 4, 0, 3, 7, 10, 4, 5]
al.sort()
print(al)
```

```
[0, 2, 3, 4, 4, 5, 7, 10]
```

Erro comum!!!

- O que será impresso com pelo código abaixo?

```
al = [2, 4, 0, 3, 7, 10, 4, 5]
outra_lista = al.sort()
print(outra_lista[0])
```

- O que se pode imprimir para evidenciar o problema acima?
  - Habilidade importante!
- Consulte na documentação o que o método retorna
  - Se retorna alguma coisa
    - ✓ https://docs.python.org/3/search.html

Erros de Sintaxe (Syntax Errors)

- Sintaxe
  - É um conjunto de regras que definem o que são ou não sentenças válidas numa determinada linguagem
- Erro de sintaxe
  - Ocorre quando executamos uma sentença inválida
- Exemplo:

```
if 3 > 1
     print("True")
else
     print("False")
```

Erros de Sintaxe (Syntax Errors)

- Exemplos comuns:
  - Abrir mas não fechar um desses caracteres: `[ ( { " '`
  - Identação incorreta
  - Erro de digitação de um keyword (palavra reservada)
  - Colocar um keyword no lugar errado
    - ✓ Exemplo: `for in x [1,2,3]:`

Erros em Tempo de Execução (Runtime Errors)

- Ocorre quando a sintaxe de um programa está correta
  - porém os dados do programa levaram o interpretador do python a encontrar uma inconsistência.
- Interrompe a execução do seu programa (crash)
- Exemplos:
  - Divisão por zero
  - Executar uma operação com tipos incompatíveis
    - ✓ Ex.: 10/"5"
  - Acessar elementos de uma lista ou dicionário que não existem
- Analogia:
  - Flap your arms and fly to Australia
    - ✓ Abra seus braços e voe para a Australia

Erros de Lógica (Logical Errors)

- Fazem o programa produzir um resultado incorreto
  - mas não causam um crash (interrupção da execução).
- Exemplo:

```
a = "10"
if a == 10:
    print("True")
else:
    print("False")
```

```
False
```

dict (Dicionário)

- Uma coleção de key-value (chave-valor) de tamanho flexível
  - O key e o value podem ser (quase) qualquer objeto python

```
empty_dict = {} # Cria um dicionário vazio
d1 = {'tipo' : 'carro', 'ano':2020, 'ocup' : [1, 2]}
d1['cor'] = 1 # Cria um novo par chave-valor
d1['fabric'] = 3 # Cria um novo par chave-valor
print(d1)
print(d1['ocup'])
```

```
{'ocup': [1, 2], 'ano': 2020, 'fabric': 3, 'tipo': 'carro', 'cor': 1}
[1, 2]
```

dict (Dicionário)

- Outros métodos relacionados a dicionários

```
d1 = {'tipo' : 'carro', 'ano':2020, 'ocup' : [1, 2]}
print('tipo' in d1) # Testa: 'tipo' está em d1.keys()?
del d1['ano'] # Remove o par identificado por 'ano'
ret = d1.pop('ocup') # Remove do dict e retorna o valor
print(list(d1.keys()))
d1.update({'motor' : '18c', 'dono' : 12}) #atualiza
print(d1)
```

```
True
['tipo']
{'motor': '18c', 'dono': 12, 'tipo': 'carro'}
```

dict (Dicionário)

- O método items retorna um iterator para uma sequência
  - de tuplas (chave, valor)

```
d1 = {'tipo' : 'carro', 'ano':2020, 'ocup' : [1, 2]}
for k, v in d1.items():
     print(k, v)
```

```
tipo carro
ano 2020
ocup [1, 2]
```

Criando um dicionário com listas

```
s = [['yellow',1],['blue', 2],['yellow', 3],['blue', 4],['red', 1]]
d = dict()
for k, v in s:
    if k in d.keys(): # Testar se já existe esta chave no dicionário
        d[k].append(v) # Se existir, adiciona um elemento
    else:
        d[k] = [v] # Se não existir, cria uma lista e o adiciona
print(d)
```

```
{'yellow': [1, 3], 'blue': [2, 4], 'red': [1]}
```

Criando um dicionário com listas com defaultdict

- No defaultdict
  - Cada elemento do dicionário é inicializado com um valor padrão

```
from collections import defaultdict
s = [['yellow',1],['blue', 2],['yellow', 3],['blue', 4],['red', 1]]
d = defaultdict(list)
for k, v in s:
    d[k].append(v)
print(d)
```

```
defaultdict(<class 'list'>, {'yellow': [1, 3], 'blue': [2, 4], 'red': [1]})
```

Tuplas (tuple)

- Uma sequência de elementos
  - Ordenada e imutável

```python
# Uma tupla é iterável
tup = (True, "banana", 2, None)
for x in tup:
    print(x)
```

```python
# Uma tupla é imutável
tup[0] = "a"
```

  - Exemplos de uso
    - ✓ dict.items()
    - ✓ Retorno de uma função: return a, b

Conjunto (set)

- Set é uma coleção não ordenada de objetos distintos
  - útil para
    - ✓ testar pertinência a grupos/conjuntos,
    - ✓ fazer operações como intersecção e união e
    - ✓ remover elementos repetidos de uma lista.
- Como criar um set

```
1  c1 = {1, 2, 3, 4}
2  c2 = {3, 4, 5, 6}
```

Operações em Conjuntos

- Pertinência

```
1  c1 = {1, 2, 3, 4}
2  c2 = {3, 4, 5, 6}
```

```
1  print(1 in c1)
2  print(9 in c2)
```

```
True
False
```

Operações em Conjuntos

- União
  - usa-se o operador | (pipe), ou
    - ✓ o método `set.union()`

```
1  c1 = {1, 2, 3, 4}
2  c2 = {3, 4, 5, 6}
```

```
1  print(c1 | c2)
```

```
{1, 2, 3, 4, 5, 6}
```

Operações em Conjuntos

- Intersecção
  - usa-se o operador &, ou
    - ✓ o método `set.intersection()`

```
1  c1 = {1, 2, 3, 4}
2  c2 = {3, 4, 5, 6}
```

```
1  print(c1 & c2)
```

```
{3, 4}
```

Função zip

- Junta os elementos de duas sequências,
  - na forma de pares (tuplas)

```
1  letras = ["a", "b", "c"]
2  numeros = [1, 2, 3]
3
4  for l, n in zip(letras, numeros):
5      print(f'{l}, {n}')
```

```
a, 1
b, 2
c, 3
```

Instruções inline (na mesma linha)

```
a = 2

# Sintaxe: valor_se_verdadeiro if condicao else valor_se_falso

b = 0 if a > 10 else 1

print(b)

# [ OPERACAO_VAL for VAL in SEQUENCIA ]

resultado1 = [x for x in range(10)] # list comprehension

resultado2 = [x * x for x in range(10)]

print(resultado1)

print(resultado2)
```

```
1
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

Requisições HTTP

- HTTP
  - protocolo de transmisão de Hipertextos
- Requisição
  - invocação de uma ação num servidor web
    - ✓ Espera-se uma resposta, que pode ser
      - uma confirmação, um conjunto de dados ou um Código de erro
        - Exemplo 1: !curl http://www.google.com (execute numa célula de um Notebook)
        - Exemplo 2: !curl https://api.exchangerate-api.com/v4/latest/USD

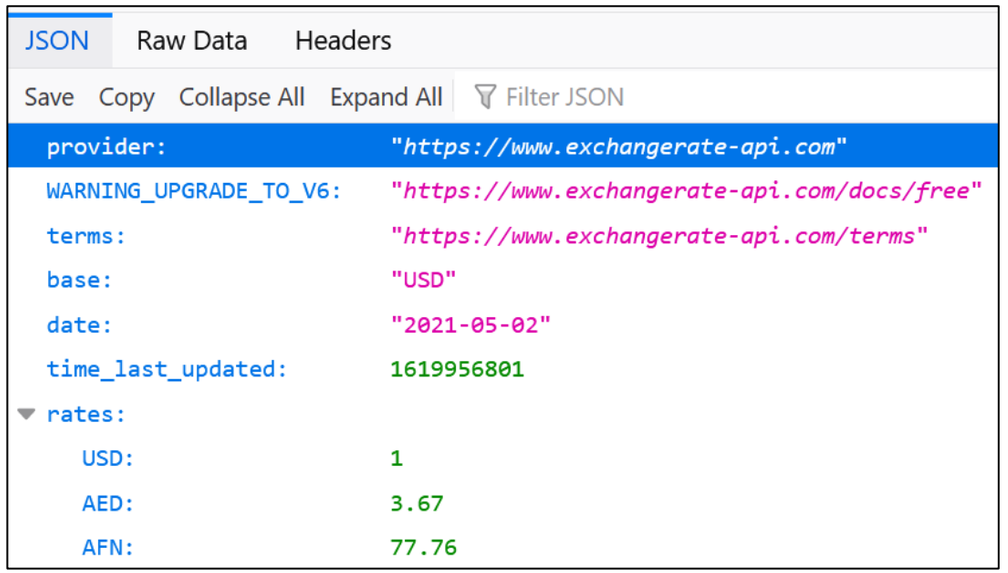
Cliente envia uma requisição

Servidor envia uma resposta

Requisições HTTP

- !curl https://api.exchangerate-api.com/v4/latest/USD
  - Retorna um arquivo JSON


```
{"provider":"https://www.exchangerate-
api.com","WARNING_UPGRADE_TO_V6":"https://www.exchangerate-ap
/docs/free","terms":"https://www.exchangerate-api.com
/terms","base":"USD","date":"2021-05-02","time_last_updated":
{"USD":1,"AED":3.67,"AFN":77.76,"ALL":101.34,"AMD":520.69,"AN
```

Requisições HTTP em Python

```python
import requests
r = requests.get("https://api.exchangerate-api.com/v4/latest/USD")
resp_json = r.json()
print(resp_json)
```

```
{'provider': 'https://www.exchangerate-api.com', 'WARNING_UPGRADE_TO_V6':
', 'terms': 'https://www.exchangerate-api.com/terms', 'base': 'USD', 'date
56801, 'rates': {'USD': 1, 'AED': 3.67, 'AFN': 77.76, 'ALL': 101.34, 'AMD'
': 93.44, 'AUD': 1.29, 'AWG': 1.79, 'AZN': 1.7, 'BAM': 1.62, 'BBD': 2, 'BI
': 1949.91, 'BMD': 1, 'BND': 1.33, 'BOB': 6.87, 'BRL': 5.36, 'BSD': 1, 'BT
': 2, 'CAD': 1.23, 'CDF': 1982.38, 'CHF': 0.912, 'CLP': 704.36, 'CNY': 6.4
```

Requisições HTTP em Python

```python
import requests
r = requests.get("https://api.exchangerate-api.com/v4/latest/USD")
resp_json = r.json()
print(resp_json)
```

```python
print(resp_json['provider'])
print(resp_json['date'])
print(resp_json['rates'])
print(resp_json['rates']['BRL'])
```

- Inspeção visual de arquivos JSON
  - https://jsonformatter.org/json-viewer
- APIs do Governo Federal
  - https://www.gov.br/conecta/catalogo

Tratamento de Exceção em Python

- Divisão por zero gera uma exceção
  - Como fazer para o seu programa continuar a execução,
    - ✓ e falhar graciosamente ?

```
2/2
```

```
1.0
```

```
a = 0 / 0
print("Passou aqui")
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-23-efda217e1f08> in <module>
----> 1 a = 0 / 0
      2 print("Passou aqui")

ZeroDivisionError: division by zero
```

Tratamento de Exceção em Python

- Use a sintaxe de tratamento de exceção com
  - try e except, conforme o exemplo a seguir
- Boas práticas
  - Coloque poucas linhas de código dentro do escopo do try/except
  - Capture exceções específicas

```
try:
    a = 0/0
    print("resultado: {0}".format(i))
except ZeroDivisionError as e:
    print(e)
    print("deu erro.")
    pass
```

```
division by zero
deu erro.
```

Prática no Colab Notebook

- Escolham por onde começar: Teoria, Warmup ou Exercícios;
  - As soluções dos warmups já estão publicadas;
  - As soluções dos exercícios extra serão disponibilizadas ao final do dia;
- É esperado que não terminem todos os exercícios durante a aula;
  - Façam o restante ao longo da semana.
- **Ao final da lista você será capaz executar tarefas relevantes**