Lógica de Programação Orientada a Objetos

Bem-vindos!

Objetivos

- Um pouco sobre Python
- Variáveis
- Estruturas Condicionais

Características Python

- Tipagem dinâmica
- Sintaxe intuitiva
- Interpretado
- Tipos de dados de alto nível
- Compromisso entre shell script e C++/Java

INTERPRETADO?

- O que significa para uma linguagem ser "interpretada?"
- Dica - "interpretado" e "compilado" se refere a implementações, não linguagens
- Essencialmente, código-fonte pode ser executado diretamente

PORQUE APRENDER PYTHON?

- Fácil de aprender a praticar
- Fortemente usado no mercado: Google,
- Facebook(Instagram), Microsoft, Dropbox,
- Globo.com, etc.
- Utilizando em várias áreas - web, data science,
- devops, automação, IA e muito mais

COMO FOI E COMO O PYTHON É FEITO?

- Comunidade de voluntários, aka core
- developers (você também pode ser um)
- Processo transparente através do Python
- Enhancement Proposals (PEPs)

Comparativo Java, C++ e Python

JAVA

```java
public class Hello{
  public static void main(String[] args){
    System.out.println("Hello, world!");
  }
}
```

C++

```cpp
#include <iostream>

int main(){
  std::cout << "Hello World!" << std::endl;
  return 0;
}
```

PYTHON

```python
print('Hello World!')
```

LET'S CODE

PORQUE PYTHON 3?

- ✔ Financiamento fornecido pela Google
- ✔ Uma língua nova nasceu (só um pouco)

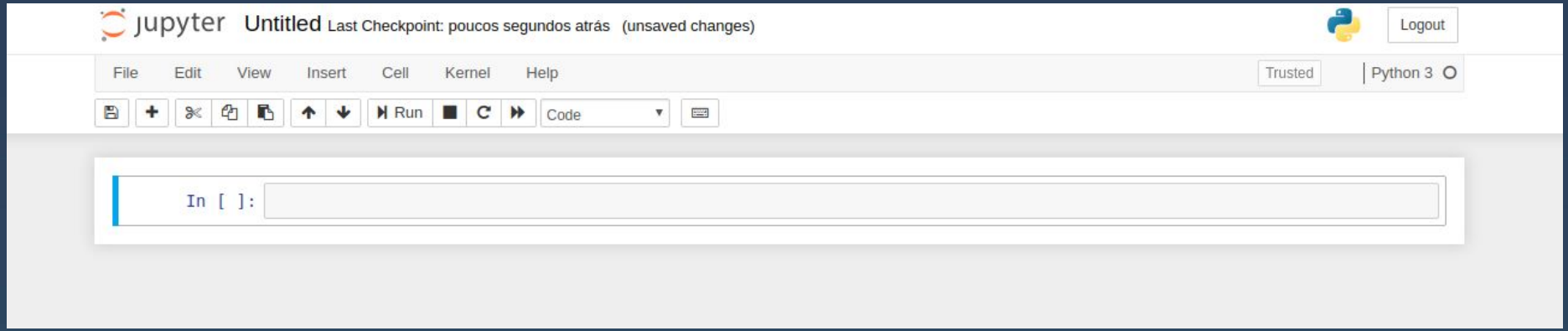
PYTHON 3 INCOMPATÍVEL COM VERSÕES ANTERIORES!

- print e exec tornam-se funções
- Todo o texto (str) é Unicode e texto encoded é um dado binário (bytes)
- Outras pequenas alterações no standard library

ENTÃO, POR QUE USAR PYTHON 3?

- Encoding adequado
- Programação assíncrona (async/await)
- Inserção do virtualenv na Standard Library
- Diretório __pycache__
- Argumento com somente palavras-chave (PEP 3102)

FERRAMENTAS PARA CODIFICAÇÃO



PORQUE EU DEVO USAR ISSO??

- ✔ Fácil!
- ✔ Ajuda a testar o comportamento do código Python

PYTHON COMO UMA CALCULADORA

- Tente executar o seguinte código no seu python jupyter:

```
>>> 10 + 10
20
>>> 50 * 2
100
>>> 10 + 20 * 3
70
>>> (10 + 20) * 3
90
```

- O que ** faz?
- O que % faz?

VARIÁVEIS E COMPARAÇÕES

Variáveis em python podem armazenar valores.

Isso é feito usando o operador de atribuição:

```
>>> a = 100
>>> a
100
>>> b = 200
200
>>> c = a + b
300
```

REATRIBUINDO

Você pode modificar o valor de qualquer variável somente reatribuindo o valor:

```
>>> a = 10
>>> a
10
>>> a = 20
>>> a
20
>>> a = 5.5
>>> a
5.5
```

ATRIBUIÇÃO AUMENTADA

```
>>> a = 50
>>> a += 10
>>> a
60
```

- A instrução acima é o mesmo que isso: a = a + 10
- Tente: -=, *=, /=, //=, %= e **=.

CONVENÇÃO PARA NOMES

De acordo com as convenções Python, nomes de variáveis deve ser minúsculas com palavras separadas por underlines:

```
# Bom
age = 12
first_name = "Joe"
x = 100

# Ruim!
J = 5.5
UserName = "itzik"
numOfRetries = 5
```

ERROS DE NOMES

- O que acontece se você tentar acessar o valor de uma variável que não existe?

```
>>> x = 100
>>> x
100
>>> y
NameError: name 'y' is not defined
```

- No script Python, cada variável deve ser atribuída antes dela ser acessada:

```
print(x)  # This line raises NameError
x = 1
```

REMOVENDO NOMES

Variáveis ("nomes") podem ser removidas com del

```
>>> x = 100
>>> x
100
>>> del x
>>> x
NameError: name 'x' is not defined
```

COMPARAÇÕES

- Cada tipo de dado tem sem comportamento específico quando encontra cada operador de comparação diferente
- Vamos ver mais no próximo slide, mas vamos verificar alguns exemplos agora:

```
>>> 1 > 2
True
>>> 5 < 7 <= 10:
True
>>> 'abd' > 'abc'
True
```

Principais Operadores

Operação	Significado
<	estritamente menor que
<=	menor ou igual que
>	estritamente maior que
>=	maior ou igual que
==	igual
!=	diferente
is	identidade do objeto
is not	não identidade do objeto

BOOLEANOS

Os tipos booleanos literais no Python são True e False. Vamos rodar eles através da declaração if:

```
>>> if True:
        print('Sure is')
Sure is

>>> if False:
        print('You shouldn't see me')
```

BOOLEANOS E OUTROS TIPOS DE DADOS

- Os seguintes valores são definidos como False:

  None

  0

  [ ](ou qualquer outra sequência vazia, string inclusive)
- Tudo diferente dos anteriores age como um True

OPERAÇÕES BOOLEANAS

Operação	Resultado
x or y	SE x é False, então y, senão x
x and y	SE x é False, então x, senão y
not x	SE x é False, então True, senão False

BLOCOS

- Todos os conteúdos das estruturas de controle (if, while/for, funções) devem ser um bloco.
- Blocos são determinados por endentação
- O estilo pep8 orienta definir em 4 espaços para indentação padrão.
- Se você estiver usando Tab para endentar, tenha certeza que seu editor converta para 4 espaços.

```
if True:
    print('Yes it is')
```

IF SIMPLES

```
x = int(input())  # Lembre de usar raw_input() no Python 2
if x > 5:
    msg = 'Higher than five'
elif x == 5:
    msg = 'Equels five'
else:
    msg = 'Lower than five'
print(msg)
```

OPERADOR TERNÁRIO

```
x = int(input())
msg = 'Higher than five' if x > 5 else 'Equel or lower than five'
print(msg)
```

EXERCÍCIOS

- If-else:

https://www.hackerrank.com/challenges/py-if-else/problem