# Introdução - Python

Prof. Igor Avila Pereira
igor.pereira@riogrande.ifrs.edu.br

Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul (IFRS)
Câmpus Rio Grande
Divisão de Computação

# Agenda

1. Introdução
2. Instalação
3. Tipagem Dinâmica
4. Sintaxe
5. Controle de Fluxo
   - Laços
6. Tipos
   - Números
   - Texto
   - Listas
   - Tuplas
   - Dicionário
   - Verdadeiro, falso e nulo
   - Operadores booleanos
7. Funções
8. Exercícios

# Introdução

Python é uma linguagem de altíssimo nível orientada a objeto, de tipagem dinâmica e forte, interpretada e interativa.

Multiparadigma, a linguagem suporta programação modular e funcional, além da orientação a objetos

Python é um software de código aberto, permitindo que seja inclusive incorporado em produtos proprietários

# Introdução

Softwares que utilizam Python:

- BrOffice.org
- PostgreSQL
- Blender
- GIMP
- Inkscape

É uma linguagem bem aceita na indústria, tais como:

- Google
- Yahoo
- Microsoft
- Nokia
- Disney

# Instalação

A versão estável mais recente está disponível para *download* no endereço: `http://www.python.org/download/`

## Windows

Para a plataforma Windows, basta executar o instalador. Para outras plataformas, como em sistemas Linux, geralmente o Python já faz parte do sistema, porém em alguns casos pode ser necessário compilar e instalar o interpretador a partir dos arquivos fonte.

# Instalação

**Windows 64 bits**

1. **Instalar o PostgreSQL:** http://www.postgresql.org/download/
2. **Instalar o Python 2.7:** https://www.python.org/downloads/
   Obs: Habilite a opção de adicionar ao PATH.
3. Instalar o PIP:
   `https://pip.pypa.io/en/latest/installing/`
4. **Trabalhar com PostgreSQL com Python:** Instale o psycopg2 (.exe)
   http://www.stickpeople.com/projects/python/win-psycopg/

# Instalação

## Ubuntu

1. **Instalar o PostgreSQL:**

   sudo apt-get install postgresql-9.4

   sudo apt-get install pgadmin3

2. **Instalar Python 2.7:**
   http://askubuntu.com/questions/101591/how-do-i-install-python-2-7-2-on-ubuntu

3. **Instalar o Flask:**

   Abra o terminal e digite "pip install Flask"

4. **Trabalhar com PostgreSQL com Python:**

   sudo apt-get install python-psycopg2

# Tipagem Dinâmica

Python utiliza tipagem dinâmica, o que significa que o tipo de uma variável é inferido pelo interpretador em tempo de execução.

- No momento em que uma variável é criada através de atribuição, o interpretador define um tipo para a variável, com as operações podem ser aplicadas.

# Tipagem Dinâmica

A tipagem do Python é forte, ou seja, o interpretador verifica se as operações são válidas e não faz coerções automáticas entre tipos incompatíveis.

- Para realizar a operação entre tipos não compatíveis, é necessário converter explicitamente o tipo da variável ou variáveis antes da operação

# Sintaxe

## Comentário de uma linha

- #

## Comentário funcionais

- São usados para:
  - alterar a codificação do arquivo fonte do programa:
    - Ex: #-* coding: utf8 -*#-
  - Definir o interpretador
    - Ex: #!/usr/bin/env/python

# Controle de Fluxo

## If

```
if <condição>:
    <bloco de código>
elif <condição>:
    <bloco de código>
elif <condição>:
    <bloco de código>
else:
    <bloco de código>
```

# Controle de Fluxo

## Entrada de Dados

```
temp = int(raw_input('Entre com a temperatura: '))

if temp < 0:
    print 'Congelando...'
elif 0 <= temp <= 20:
    print 'Frio'
elif 21 <= temp <= 25:
    print 'Normal'
elif 26 <= temp <= 35:
    print 'Quente'
else:
    print 'Muito quente!'
```

# Laços

## While

```
while <condição>:
    <bloco de código>
    continue
    break
else:
    <bloco de código>
```

# Laços

## While: Exemplo

```
# Soma de 0 a 99
s = 0
x = 1

while x < 100:
    s = s + x
    x = x + 1
print s
```

# Laços

## For

```
for <referência> in <sequência>:
    <bloco de código>
    continue
    break
else:
    <bloco de código>
```

# Laços

**For: Exemplo**

```
# Soma de 0 a 99
s = 0
for x in range(1, 100):
    s = s + x
print s
```

# Tipos

Variáveis no interpretador Python são criadas através da atribuição e destruídas pelo coletor de lixo, quando não existem mais referência a elas.

# Tipos

Existem vários tipos simples de dados pré-definidos no Python, tais como:

- Números (inteiros, reais, complexos....)
- Texto

Além disso, existem tipos que funcionam como coleções. Os principais são:

- Lista
- Tupla
- Dicionário

# Tipos

Os tipos no Python podem ser:

- **Mutáveis:** permite que o conteúdo das variáveis seja alterado
- **Imutáveis:** não permite que o conteúdo das variáveis seja alterado

# Números

Python oferece alguns tipos numéricos:

- **Inteiro** (*int*): i = 1
- **Real de ponto flutuante** (*float*): f = 3.14
- **Complexo** (*complex*): c = 3 + 4j

# Números

```
# -*- coding: latin1 -*-

# Convertendo de real para inteiro
print 'int(3.14) =', int(3.14)

# Convertendo de inteiro para real
print 'float(5) =', float(5)

# Calculo entre inteiro e real resulta em real
print '5.0 / 2 + 3 = ', 5.0 / 2 + 3

# Inteiros em outra base
print "int('20', 8) =", int('20', 8) # base 8
print "int('20', 16) =", int('20', 16) # base 16

# Operações com números complexos
c = 3 + 4j
print 'c =', c
print 'Parte real:', c.real
print 'Parte imaginária:', c.imag
print 'Conjugado:', c.conjugate()
```

# Números

Operações Aritméticas:

- **Soma** (+)
- **Diferença** (-)
- **Multiplicação** (*)
- **Divisão** (/)
- **Divisão inteira** (//)
  - o resultado é truncado para o inteiro imediatamente inferior, mesmo quando aplicado em números reais, porém neste caso o resultado será real também
- **Módulo** (%)
- **Potência** (**). Ex: 100**0.5
- **Positivo** (+)
- **Negativo** (-)

# Números

Operações lógicas:

- Menor ($<$)
- Maior ($>$)
- Menor ou igual ($<=$)
- Maior ou igual ($>=$)
- Igual ($==$)
- Diferente ($!=$)

# Números

Além dos operadores, também existem algumas funções que lidam com tipos numéricos:

- *abs()*: valor absoluto
- *pow()*: que eleva um número por outro
- *round()*: que retorna um número real com o arrendondamento especificado

# Texto

- As *strings* no Python são utilizadas para armazenar texto.
- Como imutáveis, não é possível adicionar, remover ou mesmo modificar algum caractere de uma *string*.
- Para realizar essas operações, o Python precisa criar uma nova *string*

A inicialização de *strings* pode ser:

- Com aspas simples ou duplas
- Em várias linhas consecutivas, desde que seja entre três aspas simples ou duplas

# Texto

```
# -*- coding: latin1 -*-

s = 'Camel'

# Concatenação
print 'The ' + s + ' run away!'

# Interpolação
print 'tamanho de %s => %d' % (s, len(s))

# String tratada como sequência
for ch in s: print ch

# Strings são objetos
if s.startswith('C'): print s.upper()

# o que acontecerá?
print 3 * s
# 3 * s é consistente com s + s + s
```

# Texto

Fatias de *strings* podem ser obtidas colocando índices entre colchetes após a *string*

Os índices no Python:

- Começam em zero
- Contam a partir do fim se forem negativos
- Podem ser definidos como trechos, na forma [início:fim +1:intervalo].
  - Se não for definido o início, será considerado como zero. Se não for definido o fim + 1, será considerado o tamanho do objeto. O intervalo (entre os caracteres), se não for definido, será 1.
- É possível inverter *strings* usando um intervalo negativo

# Listas

- Listas são coleções heterogêneas de objetos, que podem ser de qualquer tipo, inclusive outras listas.
- As listas no Python são mutáveis, podendo ser alteradas a qualquer momento.
- Listas podem ser fatiadas da mesma forma que as strings, mas como as listas são mutáveis, é possível fazer atribuições a itens da lista.

# Listas

```
# Uma nova lista: Brit Progs dos anos 70
progs = ['Yes', 'Genesis', 'Pink Floyd', 'ELP']

# Varrendo a lista inteira
for prog in progs:
    print prog

# Trocando o último elemento
progs[-1] = 'King Crimson'

# Incluindo
progs.append('Camel')

# Removendo
progs.remove('Pink Floyd')

# Ordena a lista
progs.sort()

# Inverte a lista
progs.reverse()

# Imprime numerado
for i, prog in enumerate(progs):
    print i + 1, '=>', prog

# Imprime do segundo item em diante
print progs[1:]
```

# Tuplas

Semelhantes as listas, porém são imutáveis: não se pode acrescentar, apagar ou fazer atribuições aos itens.

# Tuplas

Sintaxe:

```
tupla = (a, b, ..., z)
```

Os parênteses são opcionais.

Particularidade: tupla com apenas um elemento é representada como:

```
t1 = (1,)
```

Os elementos de uma tupla podem ser referenciados da mesma forma que os elementos de uma lista:

```
primeiro_elemento = tupla[0]
```

Listas podem ser convertidas em tuplas:

```
tupla = tuple(lista)
```

E tuplas podem ser convertidas em listas:

```
lista = list(tupla)
```

# Dicionário

- Um dicionário é uma lista de associações compostas por uma chave única e estruturas correspondentes.
- Dicionários são mutáveis, tais como as listas.
- A chave precisa ser de um tipo imutável, geralmente são usadas strings, mas também podem ser tuplas ou tipos numéricos.
- Já os itens dos dicionários podem ser tanto mutáveis quanto imutáveis. O dicionário do Python não fornece garantia de que as chaves estarão ordenadas.

# Dicionário

```
# Progs e seus albuns
progs = {'Yes': ['Close To The Edge', 'Fragile'],
    'Genesis': ['Foxtrot', 'The Nursery Crime'],
    'ELP': ['Brain Salad Surgery']}

# Mais progs
progs['King Crimson'] = ['Red', 'Discipline']

# items() retorna uma lista de
# tuplas com a chave e o valor
for prog, albuns in progs.items():
    print prog, '=>', albuns

# Se tiver 'ELP', deleta
if progs.has_key('ELP'):
    del progs['ELP']
```

# Verdadeiro, falso e nulo

Em Python, o tipo booleano (bool) é uma especialização do tipo inteiro (int).

O verdadeiro é chamado **True** e é igual a 1, enquanto o falso é chamado **False** e é igual a zero.

# Verdadeiro, falso e nulo

Os seguintes valores são considerados falsos:

- False (falso).
- None (nulo).
- 0 (zero).
- ” (string vazia).
- [] (lista vazia).
- () (tupla vazia).
- (dicionário vazio).
- Outras estruturas com o tamanho igual a zero.

# Verdadeiro, falso e nulo

São considerados verdadeiros todos os outros objetos fora dessa lista.

O objeto **None**, que é do tipo NoneType, do Python representa o nulo e é avaliado como falso pelo interpretador.

# Operadores booleanos

Com operadores lógicos é possível construir condições mais complexas para controlar desvios condicionais e laços.

Os operadores booleanos no Python são: **and**, **or**, **not**, **is** e **in**.

# Operadores booleanos

- **and:** retorna um valor verdadeiro se e somente se receber duas expressões que forem verdadeiras.
- **or:** retorna um valor falso se e somente se receber duas expressões que forem falsas.
- **not:** retorna falso se receber uma expressão verdadeira e vice-versa.
- **is:** retorna verdadeiro se receber duas referências ao mesmo objeto e falso em caso contrário.
- **in:** retorna verdadeiro se receber um item e uma lista e o item ocorrer uma ou mais vezes na lista e falso em caso contrário

# Operadores booleanos

```
print 2 or 3 # Mostra 2

print not 0 # Mostra True
print not 2 # Mostra False
print 2 in (2, 3) # Mostra True
print 2 is 3 # Mostra False
```

# Funções

Funções são blocos de código identificados por um nome, que podem receber parâmetros pré-determinados.

# Funções

```
def fatorial(n):

    n = n if n > 1 else 1
    j = 1
    for i in range(1, n + 1):
        j = j * i
    return j

# Testando...
for i in range(1, 6):
    print i, '->', fatorial(i)
```

# Exercícios

1. Implementar duas funções:
   - Uma que converta temperatura em graus Celsius para Fahrenheit.
   - Outra que converta temperatura em graus Fahrenheit para Celsius.
   - Lembrando que: $F = 9/5 * C + 32$
2. Implementar uma função que retorne verdadeiro se o número for primo (falso caso contrário). Testar de 1 a 100.
3. Implementar uma função que receba uma lista de listas de comprimentos quaisquer e retorne uma lista de uma dimensão.
4. Implementar uma função que receba um dicionário e retorne a soma, a média e a variação dos valores.

# Exercícios

5. Escreva uma função que:
   - Receba uma frase como parâmetro.
   - Retorne uma nova frase com cada palavra com as letras invertidas.
6. Crie uma função que:
   - Receba uma lista de tuplas (dados), um inteiro (chave, zero por padrão igual) e um booleano (reverso, falso por padrão).
   - Retorne dados ordenados pelo item indicado pela chave e em ordem decrescente se reverso for verdadeiro.

# Introdução - Python

Prof. Igor Avila Pereira
igor.pereira@riogrande.ifrs.edu.br

Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul (IFRS)
Câmpus Rio Grande
Divisão de Computação