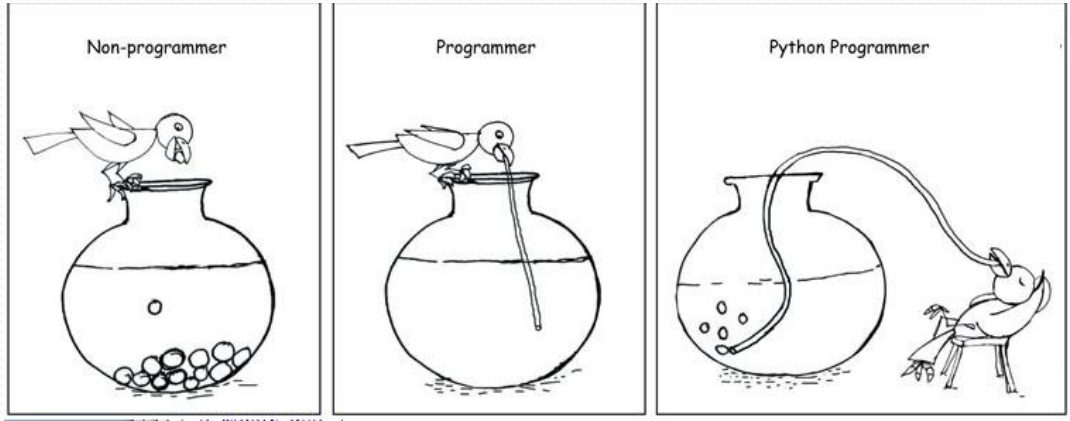
Introdução a Python

Introdução

- Python foi desenvolvido para ser portável e extensível.
- Sua sintexa promove boas práticas de programação e tende a agilizar o tempo de desenvolvimento sem sacrificar a escalabilidade e a manutenção.
- Apresenta estruturas de dados de alto nivel e uma abordagem simples para a programação orientada a objetos.
- Python é facil de aprender e ainda assim uma linguagem poderosa!



Por que Python?

- Python se tornou rapidamente uma das linguagens de programação mais populares do mundo.
  - Agora é particularmente popular para educação e computação científica.
- Há um grande número de aplicativos Python de código aberto gratuitos.
- Produtividade, extensas bibliotecas padrão e milhares de bibliotecas de código aberto de terceiros.

Por que Python?

- Ele oferece suporte a paradigmas de programação populares — procedimental, funcional, orientado a objetos e reflexiva.

- É popular em inteligência artificial, que está desfrutando de um crescimento explosivo, em parte por causa de sua relação especial com a ciência de dados.

- Há um amplo mercado de trabalho para programadores Python em muitas disciplinas, especialmente em cargos orientados para ciência de dados.

Hello World em Python

```
# Imprimindo uma linha de texto
print("Hello World!")
```

- # - Comentários em linha em Python
  - Python não tem símbolo para comentário para mais de uma linha; como /* ... */ em Java.
- Em Python, a maioria das sentenças termina sem pontuação.

Rodando Python

- Sentenças em Python podem ser executadas de duas maneiras:

1. Modo tradicional: digitando-as em um editor e salvando o arquivo como .py
    1. Interpretador começa da primeira sentença e as executa, sequencialmente, uma a uma.
    2. Para executar o programa chama-se o interpretador Python (DOS/Shell) da seguinte forma

        python *arquivo*.py

2. Modo iterativo: programador digita as sentenças diretamente para o interpretador (IDLE), que as executa uma por vez.

Mais sobre print

```
# usando , para não mudar de linha
print(“Hello”),
print (“World!”)
```

Hello World!

```
# imprimindo em multiplas
# linhas usando \
print(“Hello\nWorld!“)
```

Hello

World!

Caracteres especiais

Caracteres especiais	Significado
\n	Move o cursor para o início da nova linha.
\t	Move o cursor para o próximo tab.
\r	Move o cursor para o início da linha corrente; não avança para a próxima linha.
\b	Move o cursor um espaço para traz.
\a	Emite som de alerta.
\\	Imprime o caractere \.
\”	Aspas dupla “
\’	Aspas simples ‘

Adicionando inteiros

```
# prompt user for input
integer1 = input( "Entre o primeiro inteiro:\n" ) # lê uma string
integer1 = int( integer1 ) # converte string para inteiro

integer2 = input( " Entre o segundo inteiro :\n" ) # lê uma string
integer2 = int( integer2 ) # converte string para inteiro

soma = integer1 + integer2   # calcula e atribui a soma
print "Soma é", soma
```

input(<String>) – função do Python para ler uma String;

- Toma como argumento uma string, usada para solicitar uma entrada
- E retorna a entrada como uma string

int (var) – converte var em inteiro.

Variáveis em Python

- Variáveis podem consistir de **letras, digitos e** underscores (_)
  - **inteiro_1**
  - **_int2**
- **Não podem iniciar com um dígito**
  - **1_int**
- Python é *case sensitive*—***maiúsculas e minúsculas são diferentes***
  - **a1** e **A1**
  - **inT1** e **int1**

Variáveis em Python

- Em Phyton, variáveis são vistas como objetos.
  - Um objeto pode ter multiplos nomes, chamado identificadores. Cada identificador referencia (aponta para) um objeto na memória.

- Em linguagens como C++ e Java, o programador deve declarar o tipo do objeto (variável) antes de usá-lo no programa.

- Python, no entanto, usa tipagem dinâmica, o que significa que Python determina o tipo do objeto durante a execução do programa.

- Exemplo:
  - Se um objeto **a é inicializado com 2, então o tipo do objeto é** "integer" (pois 2 é um inteiro).

Aritmética

- Operadores aritméticos em Python

Operação	Operador Python	Expressão aritmética	Expressão Python
Adição	+	x + y	x + y
Subtração	–	x – y	x – y
Multiplicação	*	xy	x * y
Exponenciação	**	x^y	x ** y
Divisão	/	x / y	x / y
Divisão inteira	//	x div y	x // y
Módulo	%	x mod y	x % y

Ordem de precedência

Operador	Operação	Ordem
()	Parênteses	Avaliados primeiro. Quando aninhados, inicia-se pelo mais interno.
**	Exponenciação	Avaliados em segundo
*, /, //, %	Multiplicação, divisão, divisão inteira e módulo	Avaliados em terceiro
+, -	Adição, subtração	Avaliados por último

Formatação de Strings

- String é um tipo definido em Python, diferente de Java, por exemplo.
  - Permite operações poderosas sobre texto de maneira facilitada.
  - Ex. de string e aspas

```
# Criando strings e usando aspas.

print("String com \"aspas dupla.\"“)
print('Outra string com "aspas dupla."‘)
print('Esta é uma string com \'aspas simples.\'‘)
print("Outra string com 'aspas simples.'“)
```

Aspas triplas

- **Aspas triplas (“””) ou (’’’) são usadas para criar:**
  - Strings em multiplas linhas;
  - strings contendo aspas simples ou duplas
  - **docstrings**, recomendado para documentar o propósito de certos componentes de um programa.

Formatação de Strings

- Aspas simples e duplas em uma string formada por aspas triplas, não precisam do caractere ‘\’.

- Strings formadas por aspas triplas também são úteis para grandes blocos de texto, pois permitem multiplas linhas.

```
print("""Esta string tem "aspas dupla" e 'aspas simples'.
E até permite multiplas linhas."""“)

print ('''Esta string também tem aspas "dupla" e 'simples'.'''“)
```

Ignorando a quebra de linha

- Para ignorar a quebra de linha pode-se usar o caractere \

Como o ultimo caractere em um linha:

- **In []: print('this is a longer string, so we **
  : split it over two lines')

this is a longer string, so we split it over two lines.

- O interpretador remonta as partes da string em uma única string, sem quebra de linha!

Ignorando a quebra de linha

- Também é possível escrever uma sentença grande, separando-a em sentenças menores separadas por vírgula.

- **In []: print('this is a longer string, so we' ,**
  : 'split it over two lines')

this is a longer string, so we split it over two lines.

Operadores relacionais

Operador algébrico	Operador Python	Exemplo
=	==	x== y
≠	!= , <>	x!=y, x<>y
>	>	x > y
<	<	x<y
≥	>=	x >= y
≤	<=	x <= y

if e relacionais

```
‘’’ comparando inteiros usando if ’’’

# lê o primeiro numero
number1 = input( "Entre com o primeiro inteiro: " )
number1 = int( number1 )

# Lê o segundo inteiro
number2 = input( "Entre com o segundo inteiro: " )
number2 = int( number2 )

if number1 == number2:
    print(number1, ‘ é igual a ’, number2 )

if number1 != number2:
    print(number1, ‘ é diferente de ‘, number2)

if number1 < number2:
    print(number1, ‘ é menor que ‘ number2)

if number1 > number2:
    print(number1, ‘ é maior que ‘, number2 )
```

```
Entre com o primeiro inteiro: 1
Entre com o segundo inteiro: 2
1 é diferente de 2
1 é menor que 2
>>>
```

if e relacionais

docstring

```
‘’’ comparando inteiros usando if ’’’

# lê o primeiro numero
number1 = input( "Entre com o primeiro inteiro: " )
number1 = int( number1 )

# Lê o segundo inteiro
number2 = input( "Entre com o segundo inteiro: " )
number2 = int( number2 )

if number1 == number2:
    print(number1, ‘ é igual a ’, number2 )

if number1 != number2:
    print(number1, ‘ é diferente de ‘, number2)

if number1 < number2:
    print(number1, ‘ é menor que ‘ number2)

if number1 > number2:
    print(number1, ‘ é maior que ‘, number2 )
```

Use **comentários** para documentar seu código e melhorar a legibilidade.

Comentários também ajuda **você** e a outros programadores a ler e compreender seu código.

Não interfere na execução do programa.

O comando if

- A estrutura **if consiste da palavra if, a condição a ser testada, seguida de dois pontos(:).**

- Uma estrutura **if** também pode ter um corpo que em Python é delimitado **através da identação.**
  - *Diferente de Java e C++ que usam chaves* { }.

```
if number1 != number2:
        print(number1, 'é diferente de ', number2 )
        print('Dentro do corpo do if')
```

```
if number1 != number2:
        print(number1, ' é diferente de ', number2 )
print('Fora do corpo do if')
```

if e relacionais

```
#lê o primeiro numero
number1 =int( input("Entre com o primeiro inteiro: " ))

# Lê o segundo inteiro
number2 = input( "Entre com o segundo inteiro: " )
number2 = int( number2 )

if number1 == number2:
    print(number1, ' é igual a ', number2 )

    if number1 != number2:
        print (number1, ' é diferente de ', number2 )

if number1 < number2:
    print(number1, ' é menor que ', number2 )

if number1 > number2:
    print(number1, ' é maior que ', number2 )
```

dentro do corpo do if anterior

```
Entre com o primeiro inteiro: 1
Entre com o segundo inteiro: 2
1 é menor que 2
>>>
```

Exercício

- Escreva um programa que leia dois inteiros e determine se o primeiro é multiplo do segundo.

if/else e if/elif/else

```
if grade >= 90:
    print "A"
else:
    if grade >= 80:
        print "B"
    else:
        if grade >= 70:
            print "C"
        else:
            if grade >= 60:
                print "D"
            else:
                print "F"
```

=

```
if grade >= 90:
    print "A"
elif grade >= 80:
    print "B"
elif: grade >= 70:
    print "C"
elif: grade >= 60:
    print "D"
else:
    print "F"
```

Encadeando comparações

- Você pode encadear comparações para verificar se um valor está em um intervalo, por exemplo.

- A seguinte comparação determina se x está no intervalo de 1 a 5, inclusive:

```
In [1]: x = 3

In [2]: 1 <= x <= 5
Out[2]: True

In [3]: x = 10
In [4]: 1 <= x <= 5

Out[4]: False
```

Objetos e tipagem dinâmica

- Valores como 7 (inteiro), 4.1 (ponto-flutuante), ‘gato' (string) são objetos.
  - Todo objeto tem um tipo e um valor!

```
In [1]: type(7)
Out[1]: int

In [2]: type(4.1)
Out[2]: float

In [3]: type('dog')
Out[3]: str
```

  - *type*() pode ser usado para obter o tipo do objeto

Objetos e tipagem dinâmica

- **Python usa tipagem dinâmica** - determina o tipo do objeto a qual uma variável se refere durante a execução do código:

```
In [9]: type(x)
Out[9]: int

In [10]: x = 4.1

In [11]: type(x)
Out[11]: float

In [12]: x = 'dog'

In [13]: type(x)
Out[13]: str
```

O laço while

- Controlando o laço por contador

```
total = 0 # soma das notas
contaNotas = 1 # numero de notas

while contaNotas <= 10:
        nota= input( “Digite a nota: " )
        nota = int( nota )
        total = total + nota
        contaNotas = contaNotas + 1

media = total / 10
print(“A média da classe é ‘, media)
```

O laço while

- Controlando o laço por sentinela

```
total = 0 #soma das notas
contaNota = 0 # numero de notas

nota = input( "Digite a nota, -1 para sair: " )
nota = int( nota ) # converte string em inteiro

while nota != -1:    # sentinela
    total = total + nota
    contaNota = contaNota + 1
    nota = input( "Digite a nota, -1 para sair: " )
    nota = int( nota )

if contaNota <> 0:
    media = float( total ) / contaNota
    print("Média da classe foi", media)
else:
    print( "Nenhuma nota digitada“)
```

Símbolos de atribuição estendidos

Símbolo	Expressão	Explicação
+=	c += 7	c = c + 7
-=	d -= 4	d = d - 4
*=	e *= 5	e = e * 5
**=	f **= 3	f = f ** 3
/=	g /= 3	g = g / 3
%=	h %= 9	h = h % 9

A função *range*

- Define um intervalo de valores e pode tomar 1, 2 ou 3 argumentos
  1. Quando é passado 1 argumento (*fim*), este é o final do intervalo; [0,1,…,*fim* -1]
     - range (5) = [0, 1, 2, 3, 4]
  2. Quando são passados 2 argumentos (*inicio*, *fim*), estes são o inicio e o fim do intervalo [*inicio*, *inicio+1*,…, *fim-1*]
     - range(2,5) = [2, 3, 4]
  3. Com 3 argumentos, (inicio, fim, inc) estes são o inicio o fim e o incremento; [*inicio*, *inicio+inc*,…, *fim*-1]
     - range(-5,5,3) = [-5, -3, -1, 1, 3]

O laço for

- **for** contador **in range**(1, 101):
  - # varia contador de 1 a 100 com incremento de 1
- **for** contador **in range**(100, 0, –1):
  - # varia contador de 100 a 1 com decremento de 1
- **for** contador **in range**(7, 78, 7):
  - # varia contador de 7 a 77 com incremento de 7
- **for** contador **in range**(20, 1, -2):
  - # varia contador de 20 a 2 com decremento de 2

break

- break – quando executado dentro de um laço (for ou while) causa a saída imediata do laço.
  - A execução do programa continua na sentença seguite ao laço.

```
for x in range( 1, 11 ):
    if x == 5:
        break;
    print x,

print "\nbreak executado; x =", x
```

continue

- ***continue*** quando executado dentro de um laço (for ou while), pula o restante das sentenças do corpo do laço e procede para a próxima iteração do mesmo.
  - No **while**, o teste de contiação do laço é avaliado imediatamente após a execução do continue.
  - No **for**, a variável controle assume o próximo valor da sequencia (se esta possui mais valores).

```
for x in range( 1, 11 ):
    if x == 5:
        continue
    print x,
print "\Uso de continue para não imprimir o valor 5"
```

Operadores lógicos

- ***and – E lógico***
- ***or – Ou lógico (inclusivo)***
- ***not – Não lógico (negação)***

```
if animal == “Dog" and age >= 13:
    seniorDogs += 1
```

```
if not value == 10:
    print(“Valor  não é 10”)
```

Argumentos de print

- Vamos exibir 'Programação' com seus caracteres separados por dois espaços:

```
for ch in 'programação':
    ...:     print(ch, end=' ')
    ...:
p r o g r a m a ç ã o
```

A função **print** exibe seu(s) argumento(s) e move o cursor para a próximo linha. Você pode alterar esse comportamento com o argumento **end**.

Tente executar **print(ch)**

listas e iteradores

- A sequência à direita da palavra-chave da instrução ***for*** deve ser iterável.
  - Um iterável é um objeto do qual a instrução ***for*** pode tirar um item de cada vez.
    - Python tem outros tipos de sequência iteráveis além de strings.

```
In [3]: total = 0

In [4]: for number in [2, -3, 0, 17, 9]:
...: total = total + number

In [5]: total
Out[5]: 25
```

Introdução à ciência dos dados: Estatística descritiva básica

- Na ciência de dados, costuma-se usar estatísticas para descrever e resumir os dados. Algumas estatísticas descritivas são:
  - mínimo - o menor valor em uma coleção de valores.
  - máximo - o maior valor em uma coleção de valores.
  - contagem - o número de valores em uma coleção.
  - soma - o total dos valores em uma coleção.

Introdução à ciência dos dados: Estatística descritiva básica

- Python tem muitas funções integradas para realizar tarefas comuns. Funções integradas min e max calculam o mínimo e o máximo, respectivamente, de uma coleção de valores:

```
In [1]: min(36, 27, 12)
Out[1]: 12

In [2]: max(36, 27, 12)
Out[2]: 36
```

```
In [3]: len([0,2,3,7])
Out[3]: 4

In [4]: sum([0,2,3,7])
Out[4]: 12
```

Medidas de tendência central - média, mediana e moda

- Aqui continuamos nossa discussão sobre o uso de estatísticas para analisar dados com várias estatísticas descritivas, incluindo:
  - média - o valor médio em um conjunto de valores.
  - mediana - o valor médio quando todos os valores estão ordenados.
  - moda - o valor que ocorre com mais frequência.

- Estas são medidas de tendência central - cada uma é uma maneira de produzir um único valor que representa um valor "central" em um conjunto de valores, ou seja, um valor que é, em certo sentido, típico de os outros.

O módulo *statistics*

- O módulo de estatísticas da biblioteca padrão Python fornece funções para calcular a média, a mediana e a moda. Para usar esses recursos, primeiro importe o módulo de estatísticas:

```
In [3]: import statistics
```

- Então, você pode acessar as funções do módulo com "statistics.” seguido pelo nome da função para chamar

```
In [1]: grades = [85, 93, 45, 89, 85]

In [4]: statistics.mean(grades)
Out[4]: 79.4
In [5]: statistics.median(grades)
Out[5]: 85
In [6]: statistics.mode(grades)
Out[6]: 85
```

- Para confirmar que a mediana e o modo estão corretos, você pode usar a função ordenada integrada para obter um cópia das notas com seus valores organizados em ordem crescente:

```
In [7]: sorted(grades)
Out[7]: [45, 85, 85, 89, 93]
```

A moda

- Estudando os valores classificados, você pode ver que 85 é a moda porque ocorre com mais frequência (duas vezes).

- A função de moda causa um **StatisticsError** para listas como [85, 93, 45, 89, 85, 93] em que existem dois ou mais valores “mais frequentes”.
  - Esse conjunto de valores é considerado bimodal.

Exercício

- Escreva um programa que leia o valor do raio de um círculo e calcule o diâmetro e a área do círculo. Considere $\pi = 3.14159$.

Exercício

- Uma palindrome é um número ou um texto cuja leitura é a mesma tanto de frente para traz como de traz para frente. Por exemplo, cada um dos seguintes números de 5 dígitos são palindromes: 12321, 55555, 45554 e 11611. Escreva um script que leia um número de 5 dígitos e determine se este é ou não uma palindrome. Se o número não for de 5 dígitos, mostre um alerta ao usuário indicando o problema e permita que o usuário entre com um número correto após a emissão do alerta. Não deve ser usado vetores (array).

Exercício

- Um triângulo retangulo pode ter lados que são inteiros. O conjunto desses três valores é chamado tripla Pitagorea. Estes lados devem satisfazer a relação de que a soma dos quadrados dos lados é igual ao quadrado da hipotenusa.

- Encontre todas as triplas Pitagoreas, cujos valores para os lados e para a hipotenusa sejam menores que 20.

- Use um for tripo aninhado para tentar todas as possibilidades. Este método é chamado **computação por força bruta**. Para diversos problemas computacionais não existe outro algoritmo senão a força bruta!