



Universidade Federal do Piauí - UFPI

Sistemas de Informação

Programação Orientada a Objetos I - POO

Introdução ao python

Prof. Flávio Araújo - UFPI - Picos PI

História

- Iniciou em 1989;
- Holandês Guido van Rossum;
- Homenagem ao programa humorístico Monty Python (adorado pelos Nerds)
- Principal site: <http://www.python.org/>

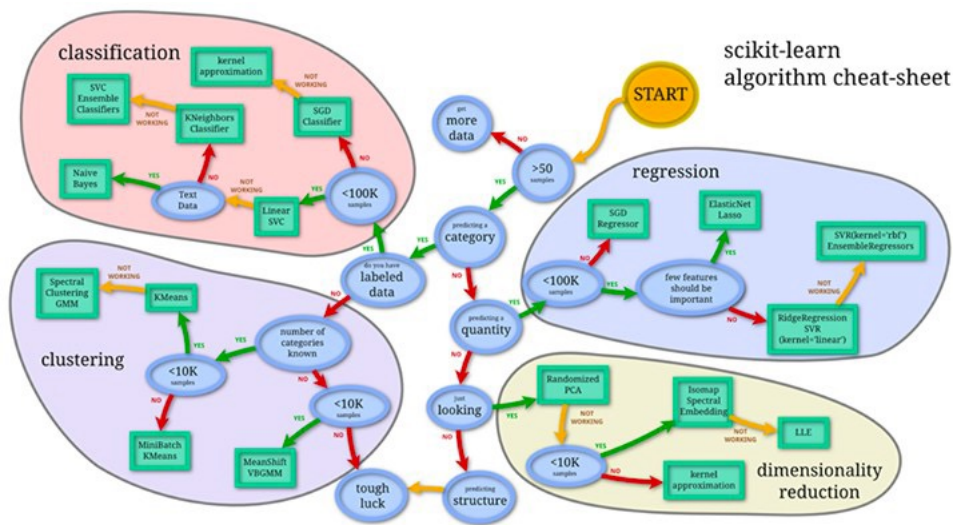


Diversos usos do python

- Python tem muitas bibliotecas e estruturas:
 - Desenvolvimento web, configuração em nuvem, análise de dados, machine learning, dentre outros.
- Python e a ciência de dados;
- Python no desenvolvimento web;
- Python é universal:
 - Windows, MacOS, Linux (Raspberry), Unix, ...
- Uma grande comunidade;

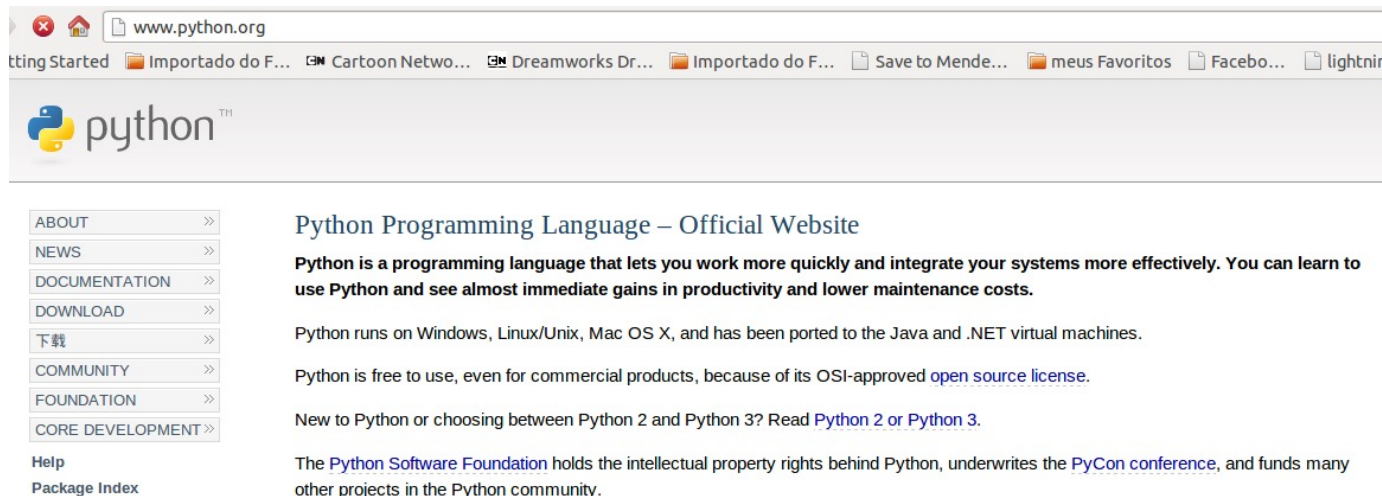
Diversos usos do python

Scikit-learn é um dos mais populares projetos para algoritmos de aprendizagem de máquina em python e é claro open-source.

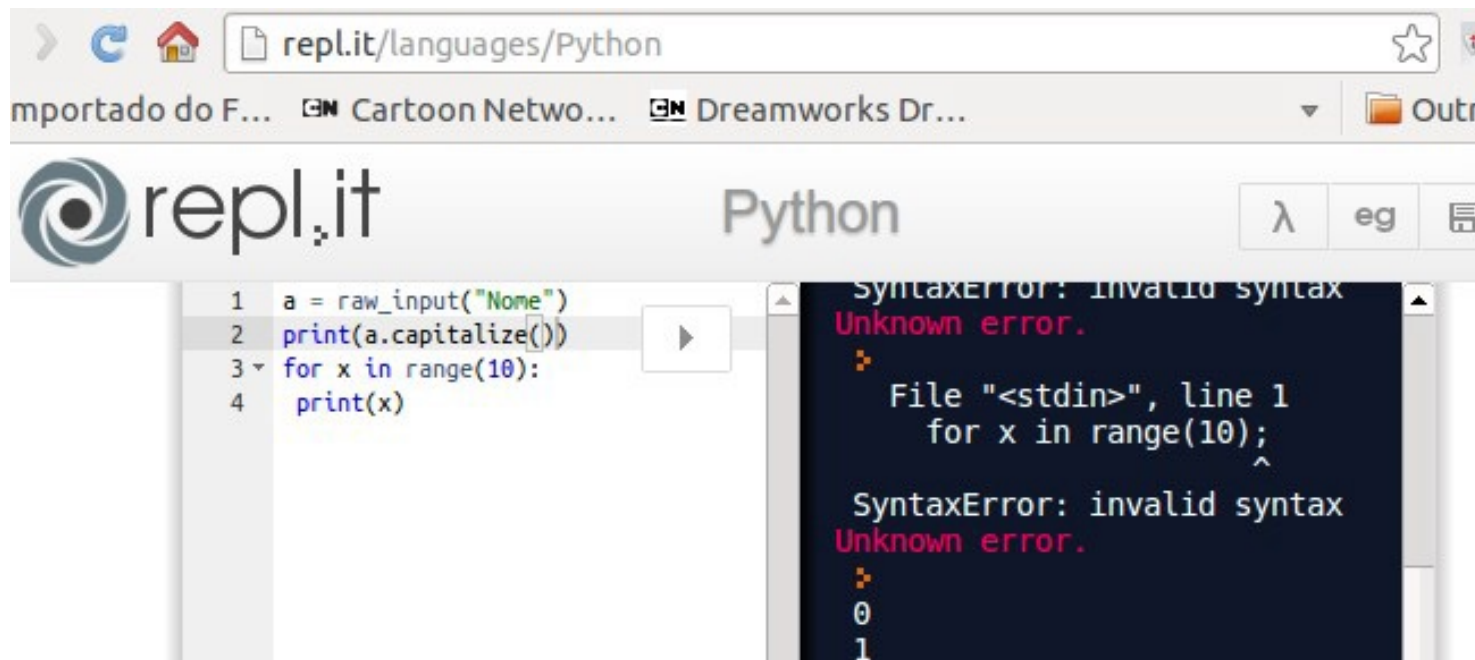


Instalação

- Linux:
 - Na maioria das distribuições já vem instalado.
- Windows e MacOS:
 - Fazer o download do site e executar o instalador



Interpretador OnLine



The screenshot shows the repl.it online Python interpreter interface. The browser address bar displays `repl.it/languages/Python`. The page header includes the `repl.it` logo, the word `Python`, and buttons for `λ`, `eg`, and a save icon. The code editor on the left contains the following Python code:

```
1 a = raw_input("Nome")
2 print(a.capitalize())
3 for x in range(10):
4     print(x)
```

A right-pointing arrow button is located between the code editor and the output console. The output console on the right displays a `SyntaxError` message:

```
SyntaxError: Invalid syntax
Unknown error.
File "<stdin>", line 1
    for x in range(10);
                        ^
SyntaxError: invalid syntax
Unknown error.
0
1
```

IDEs

- PyCharm, Eclipse, ...
- Qualquer editor de texto:
 - Notepad
 - Vim
 - Nano



Primeiras Observações

- É uma linguagem INTERPRETATIVA;
- O aninhamento é fundamental;
- Um comando por linha;
- Executando na linha de comando...

Observações

- Não misture espaços com TABs;
- Comentário: #
- Docstrings: são strings normais ou multilinhas (""" """) que tem a função de documentar módulos, classes, funções e métodos.

```
exemplo_docstring.py x
#!/usr/bin/env python
# -*- coding:utf-8 -*-

# isto é um comentário de apenas uma linha
"""
    isto é um comentário de várias linhas
"""

def span():
    "Este DOCSTRING serve para dizer que esta função faz nada"
    pass

def dobro(x):
    "retorna o dobro do parametro x"
    return x * x
```

Números

Existem quatro tipos numéricos em python: inteiros, ponto flutuante, booleano e complexo;

- Inteiros (int) - tamanho limitado pela memória do computador:
 - 32 bits = $-2^{16} .. + 2^{16}$
 - 64 bits = $-2^{32} .. + 2^{32}$
- Flutuante (float) = tamanho limitado pela memória;
- Tipos lógicos (bool) - True e False podem ser expressos por 1 e 0.
- Tipo complexo (complex): $4 + 3j$

Entrada do usuário

```
nome = input("Digite seu nome: ")  
print(nome)
```

Conversão de tipos:

```
n = input("Digite um numero: ")  
print(type(n))
```

```
n_int = int(n)  
n_float = float(n)
```

```
print(type(n_int))  
print(type(n_float))
```

Saída para o usuário

```
nome = 'flavio'
```

```
a = 2
```

```
b = 3.3456
```

```
print ('Nome:', nome, '\nValor de a:', a, '\nValor de b:', b)
```

```
print ('Nome = %s, inteiro = %d, float = %.2f' %(nome, a, b))
```

Operadores

Aritméticos		Bit a bit		Lógico	
+	Adição	&	AND	==	Igualdade
-	Subtração		OR	!=	Diferença
*	Multiplicação	^	XOR	<	Menor
/	Divisão	~	Inversor	>	Maior
//	Divisão Inteira	>>	Deslocamento Dir	<=	Menor igual
%	Módulo (resto)	<<	Deslocamento Esq	>=	Maior igual
**	Potência			in	Está contido
				Not in	Não está contido
				is	é
				Is not	Não é
				and	E lógico
				or	OU lógico
				not	Inversor lógico

Operadores

Operação	Descrição
$a \text{ is } b$	True se a e b são idênticos
$a \text{ is not } b$	True se a e b não são idênticos
$a \text{ in } b$	True se a é membro de b
$a \text{ not in } b$	True se a não é membro de b

Operador acumulativo	Substitui
$ += V$	$ = + V$
$ -= V$	$ = - V$
$ *= V$	$ = * V$
Mesma lógica para todos os outros operadores	

Comandos básicos

Comando IF

if expressão1:

Comandos1

elif expressão N*:

ComandosN*

else:

Comandos2

Expressão if (operador ternário)

Valor1 **if** expressão **else** valor2

Ex: “verdadeiro” if 4 > 3 else “falso”

Comandos básicos

```
"""  
    Exemplo do uso do IF  
"""  
  
a = input("Idade de A?")  
b = input("Idade de B?")  
  
if a < b:  
    print("A e mais novo que B")  
elif a == b:  
    print("A tem a mesma idade de B")  
else:  
    print("A e mais velho que B")
```


Exercício

Dado três lados de um triângulo verifique se é possível formar um triângulo. Em seguida diga se ele é equilátero, isóseles ou escaleno.

Lembre-se: Não é possível formar triângulo quando um lado é maior que a soma dos outros dois!

Lembrete:

Equilátero: todos lados iguais;

Isóceles: um lado diferente;

Escaleno: todos os lados diferentes.

Exercício

```
a = float(input('Tamanho do lado A: '))
b = float(input('Tamanho do lado B: '))
c = float(input('Tamanho do lado C: '))

if (a >= (b + c)) or (b >= (a + c)) or (c >= (a + b)):
    print('Nao eh possivel formar um triangulo!')
elif a == b and b == c:
    print('Triangulo equilatero!')
elif a != b and b != c and a != c:
    print('Triangulo escaleno!')
else:
    print('Triangulo isoceles!')
```

Comandos de Repetição

Comando while

while expressão1:
 Comandos1

```
while True:  
    print ('Loop infinito')
```

Exercício

Faça um programa que escreva de 0 até 40. Para múltiplos de 4 ou 10 escreva PIN na tela, quando for o último número escreva “FIM”.

Exemplo:

PIN

1

2

3

PIN

...

39

FIM

Dica de matemática: zero é múltiplo de todos os números pois qualquer número vezes 0 é 0.

Exercício

```
valor_inicial = 0
valor_final = 40

while valor_inicial <= valor_final:
    if valor_inicial % 4 == 0 or valor_inicial % 10 == 0:
        print ('PIN')
    elif valor_inicial == 40:
        print 'FIM'
    else:
        print (valor_inicial)
    valor_inicial += 1
```

Comandos de Repetição

Comando for

for elemento1, [elementosN] **in** iterável:

Comandos1

```
cidades = ['Picos', 'Teresina']
```

```
for x in cidades:
```

```
    print (x)
```

```
numeros = [1, 2, 3, 4]
```

```
for n in numeros:
```

```
    print (n)
```

```
for n in range(4):
```

```
    print (n)
```

Listas são iteráveis

Adiante será estudado Listas

Comandos de Repetição

- Comando range()
 - O range(n) cria uma lista com os valores do intervalo, mantendo apenas um valor na memória de cada vez, sendo este valor obtido de acordo com o índice solicitado.
 - Range(1, 10) -> Lista de 1 até 9, não inclui o 10.

Exercício

Faça a tabela de multiplicação dos números de 1 a 9.

Não precisa desenhar as bordas, cada tabela é sequencial.

TABUADA DO UM AO NOVE COM RESULTADOS

1	x	1	=	1
1	x	2	=	2
1	x	3	=	3
1	x	4	=	4
1	x	5	=	5
1	x	6	=	6
1	x	7	=	7
1	x	8	=	8
1	x	9	=	9
1	x	10	=	10

2	x	1	=	2
2	x	2	=	4
2	x	3	=	6
2	x	4	=	8
2	x	5	=	10
2	x	6	=	12
2	x	7	=	14
2	x	8	=	16
2	x	9	=	18
2	x	10	=	20

3	x	1	=	3
3	x	2	=	6
3	x	3	=	9
3	x	4	=	12
3	x	5	=	15
3	x	6	=	18
3	x	7	=	21
3	x	8	=	24
3	x	9	=	27
3	x	10	=	30

4	x	1	=	4
4	x	2	=	8
4	x	3	=	12
4	x	4	=	16
4	x	5	=	20
4	x	6	=	24
4	x	7	=	28
4	x	8	=	32
4	x	9	=	36
4	x	10	=	40

5	x	1	=	5
5	x	2	=	10
5	x	3	=	15
5	x	4	=	20
5	x	5	=	25
5	x	6	=	30
5	x	7	=	35
5	x	8	=	40
5	x	9	=	45
5	x	10	=	50

6	x	1	=	6
6	x	2	=	12
6	x	3	=	18
6	x	4	=	24
6	x	5	=	30
6	x	6	=	36
6	x	7	=	42
6	x	8	=	48
6	x	9	=	54
6	x	10	=	60

7	x	1	=	7
7	x	2	=	14
7	x	3	=	21
7	x	4	=	28
7	x	5	=	35
7	x	6	=	42
7	x	7	=	49
7	x	8	=	56
7	x	9	=	63
7	x	10	=	70

8	x	1	=	8
8	x	2	=	16
8	x	3	=	24
8	x	4	=	32
8	x	5	=	40
8	x	6	=	48
8	x	7	=	56
8	x	8	=	64
8	x	9	=	72
8	x	10	=	80

9	x	1	=	9
9	x	2	=	18
9	x	3	=	27
9	x	4	=	36
9	x	5	=	45
9	x	6	=	54
9	x	7	=	63
9	x	8	=	72
9	x	9	=	81
9	x	10	=	90

Exercício

```
for x in range(1,10):  
    for y in range(1,11):  
        print(x, ' * ', y, '=', x*y)  
    print('\n')
```

Quebra de Repetição

While e **for** aceitam os comandos **BREAK** e **CONTINUE**;

1. break: força a quebra da execução do bloco de repetição;
2. continue: desvia para o início do bloco de repetição.

Comandos diversos

É muito comum o uso do **try** para tratar exceções nos códigos:

```
a = input('Digite um numero: ')
b = input('Digite outro numero: ')

try:
    resultado = float(a)/b
    print (resultado)
except:
    print ('Nao existe divisao por zero!')
```

Comandos diversos: assert

O comando **assert** permite fazer testes em tempo de execução, quando não satisfeito a condição ele gera **AssertionError**. Geralmente é utilizado com **isinstance(valor, tipo)**.

```
def dobra(n):  
    return n**2
```

```
a = '2'
```

```
try:  
    assert isinstance(a, int)  
except AssertionError:  
    a = int(a)
```

```
print(dobra(a))
```

Funções/Métodos

```
def nome_da_funcao([parametro1], [parametros*]):  
    comando1  
    comando2  
    return <valor>
```

```
def multiplica(a, b):  
    return a * b
```

Depurar um programa

Depurar ou debugar o código é uma boa forma para encontrar erros:

```
a = input('Digite um numero: ') a: 3
b = input('Digite outro numero: ') b: 4

try:
    resultado = float(a)/b resultado: 0.75
    print (resultado)
except:
    print ('Nao existe divisao por zero!')
```

Variables

+	01 a = {int} 3
-	01 b = {int} 4
	01 resultado = {float} 0.75
▶	Special Variables

Atividades

1 - Faça um Programa que peça a temperatura em graus Farenheit, transforme e mostre a temperatura em graus Celsius:

$$C = (5 * (F-32) / 9).$$

2 - João Papo-de-Pescador, homem de bem, comprou um microcomputador para controlar o rendimento diário de seu trabalho. Toda vez que ele traz um peso de peixes maior que o estabelecido pelo regulamento de pesca do estado de São Paulo (50 quilos) deve pagar uma multa de R\$ 4,00 por quilo excedente. João precisa que você faça um programa que leia a variável *peso* (peso de peixes) e calcule o excesso. Gravar na variável *excesso* a quantidade de quilos além do limite e na variável *multa* o valor da multa que João deverá pagar. Imprima os dados do programa com as mensagens adequadas.

3 - Faça um Programa que pergunte quanto você ganha por hora e o número de horas trabalhadas no mês. Calcule e mostre o total do seu salário no referido mês, sabendo-se que são descontados 11% para o Imposto de Renda, 8% para o INSS e 5% para o sindicato, faça um programa que nos dê:

salário bruto.

quanto pagou ao INSS.

quanto pagou ao sindicato.

o salário líquido.

calcule os descontos e o salário líquido, conforme a tabela ao lado:

```
+ Salário Bruto : R$  
- IR (11%) : R$  
- INSS (8%) : R$  
- Sindicato ( 5%) : R$  
= Salário Líquido : R$
```

Atividades

4 - Um posto está vendendo combustíveis com a seguinte tabela de descontos:

Álcool:

- até 20 litros, desconto de 3% por litro
- acima de 20 litros, desconto de 5% por litro

Gasolina:

- até 20 litros, desconto de 4% por litro
- acima de 20 litros, desconto de 6% por litro

Escreva um algoritmo que leia o número de litros vendidos, o tipo de combustível (A para álcool e G para gasolina), calcule e imprima o valor a ser pago pelo cliente sabendo-se que o preço do litro da gasolina é R\$ 4,53 o preço do litro do álcool é R\$ 3,45.

Atividades

5 - Faça um programa que peça dois números, base e expoente, calcule e mostre o primeiro número elevado ao segundo número. Não utilize a função de potência da linguagem.

6 - Faça um programa que calcule o fatorial de um número inteiro fornecido pelo usuário. Ex.: $5! = 5 * 4 * 3 * 2 * 1 = 120$

7 - Altere o programa de cálculo do fatorial, permitindo ao usuário calcular o fatorial várias vezes (até que o usuário digite um número negativo) e limitando o fatorial a números inteiros positivos e menores que 16.

8 - Desenvolva um programa que faça a tabuada de um número qualquer inteiro que será digitado pelo usuário, mas a tabuada não deve necessariamente iniciar em 1 e terminar em 10, o valor inicial e final devem ser informados também pelo usuário, conforme exemplo abaixo:

```
Montar a tabuada de: 5
```

```
Começar por: 4
```

```
Terminar em: 7
```

```
Vou montar a tabuada de 5 começando em 4 e terminando em 7:
```

```
5 X 4 = 20
```

```
5 X 5 = 25
```

```
5 X 6 = 30
```

```
5 X 7 = 35
```

Atividades

9 - Em matemática, a Sucessão de Fibonacci (também Sequência de Fibonacci), é uma sequência de números inteiros, começando normalmente por 0 e 1, na qual, cada termo subsequente (numero de Fibonacci) corresponde a soma dos dois anteriores. A sequência recebeu o nome do matemático italiano Leonardo de Pisa, mais conhecido por Fibonacci, que descreveu, no ano de 1202, o crescimento de uma população de coelhos, a partir desta tal sequência, que já era, no entanto, conhecida na antiguidade. Os números de Fibonacci são os números que compõem a seguinte sequência: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, ... (podendo ser omitido o zero inicial). Faça um algoritmo que calcule os termos dessa sequencia, sendo a quantidade de termos indicada pelo usuário.