Introdução ao Python

Python

Márcio Palheta
marcio.palheta@gmail.com

O que é Python?

- **Linguagem de altíssimo nível**
  - Suporte nativo a estruturas de dados de alto nivel
- **Multiplataforma**
  - Unix, Windows, Symbian, Solaris, etc...
- **Multiparadigma**
  - Procedural, OO, funcional
- **Opensource**
- **Dinâmica e Forte**
- **Joga com outras linguagens**
  - (.NET) IronPython, (Java) Jython, C e C++

Por que Python?

- Aprendizado fácil
- Sintaxe limpa e de fácil leitura
- Forte suporte da comunidade
- Bem documentada
- Biblioteca
- Divertida
- Mais com menos *[código]*
- Liberdade

Quem usa Python?

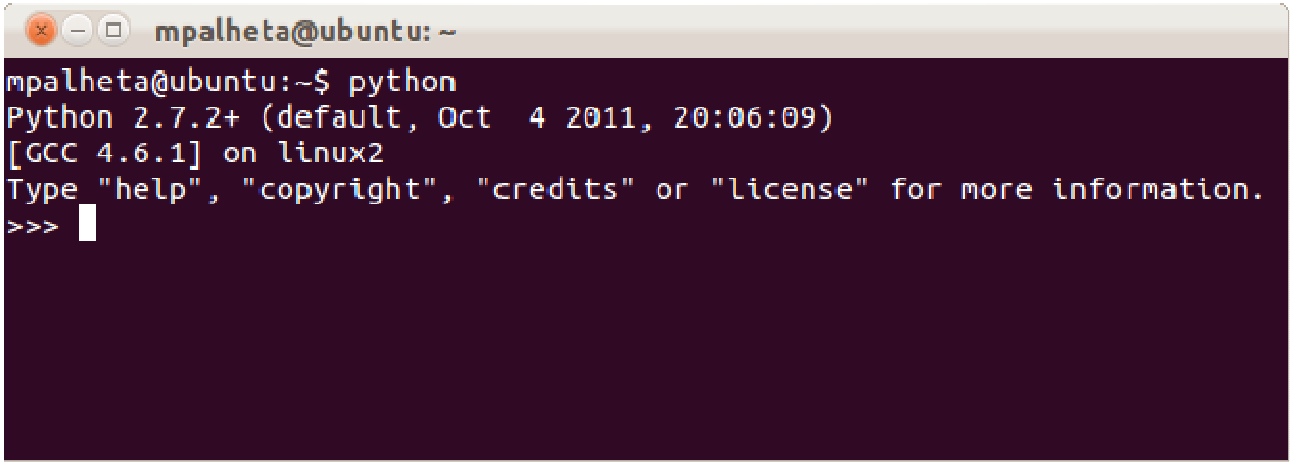
- Google
- NASA
- Nokia
- Gimp/Inkscape/Blender
- Governo (brasil.gov.br)
- Portal G1 (g1.globo.com)
- Entre outras...

http://www.python.org/about/success/

http://www.python.org.br/wiki/EmpresasPython

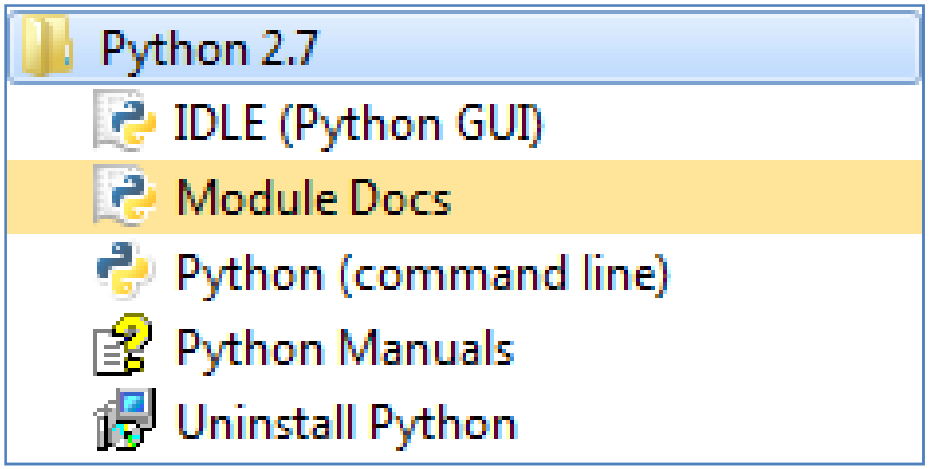
Instalação do Python - Linux

- Já vem por padrão instalado nas distribuições;
- Opção de instalação:
  - $ sudo apt-get install python
- Execução via terminal:
  - $ python



Instalação do Python

- **Windows**
  - Download:http://www.python.org/getit/windows/
  - Execute o programa baixado. Ex: python-2.7.3.msi
  - Pacotes instalados:



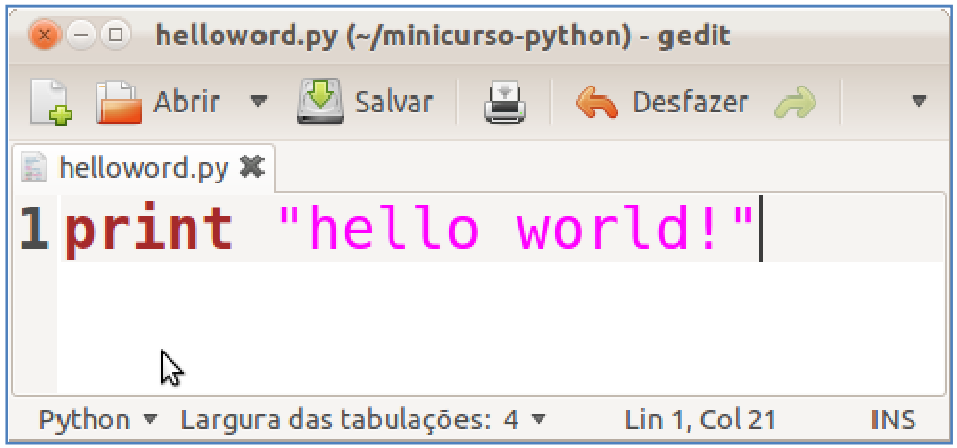
Interpretador Interativo

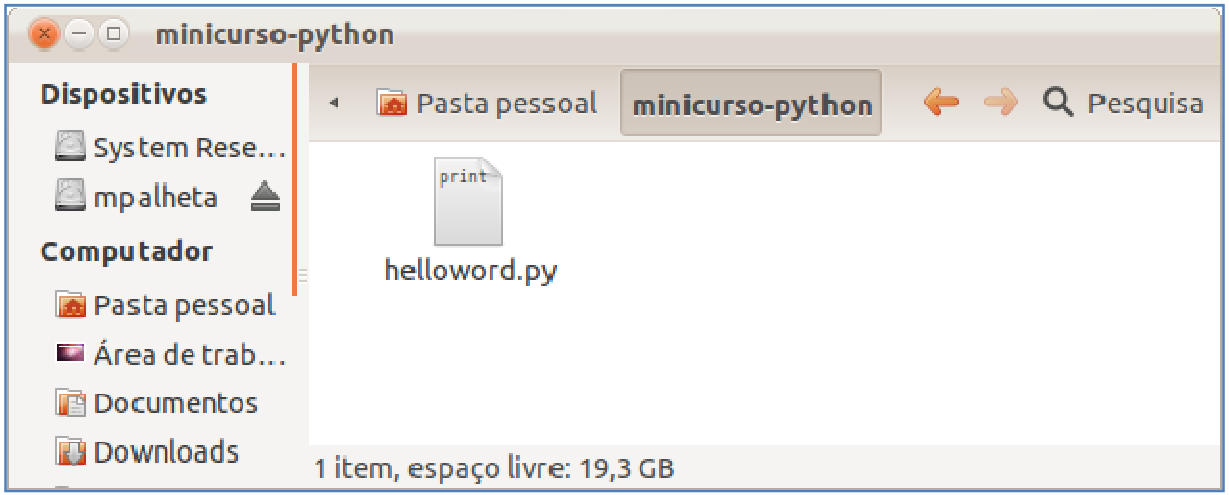
print “Hello, World!”

```
mpalheta@ubuntu: ~
mpalheta@ubuntu:~$ python
Python 2.7.2+ (default, Oct  4 2011, 20:06:09)
[GCC 4.6.1] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> print "hello world!"
hello world!
>>>
```

Executando arquivos *.py

- Use um editor de texto;
- Escreva o código para imprimir uma mensagem



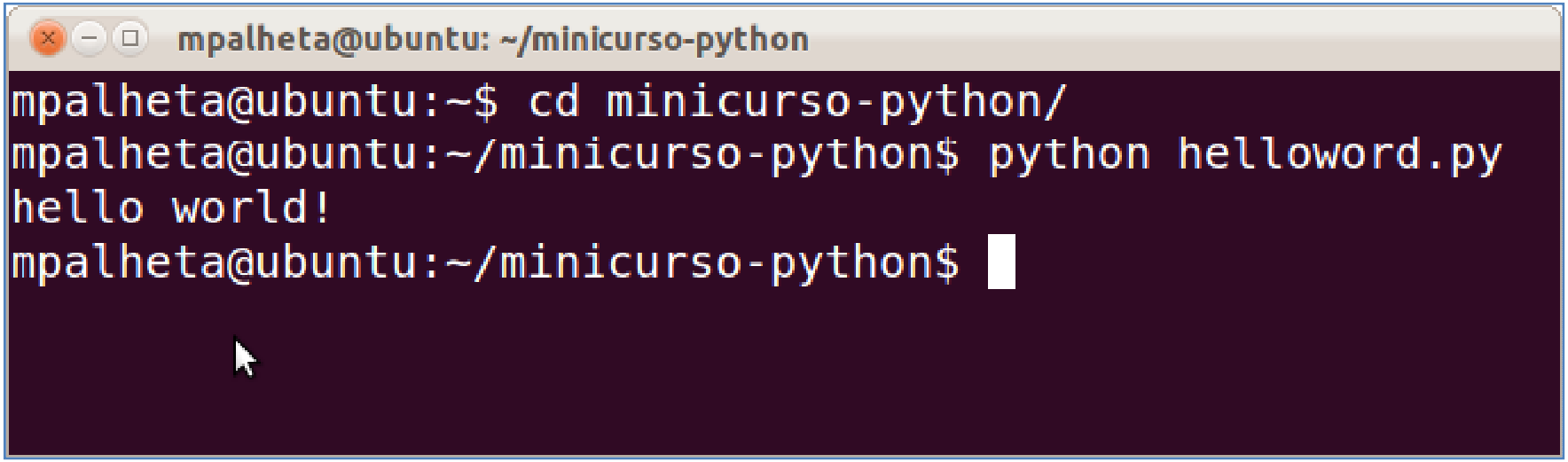


- Salve o arquivo:
- Helloworld.py

Executando arquivos *.py

- No terminal, entre no diretório onde salvou o arquivo heloworld.py;
- Digite o comando para execução de arquivos python:

$ python helloworld.py



Conceitos Básicos

- Case sensitive

```
This ≠ this
```

- Blocos por endentação

```
se condição:
    bloco
```

ERRADO

```
se condição:
bloco
```

- Tipagem dinâmica

Integer

```
a = 2
```

String

```
a = “alguma coisa”
```

Float

```
a = 2.3
```

- Tudo é objeto
- Não tem ponto e virgula no final (‘;’)
- Comentários começam com #

Por dentro da API

- Os comandos mais importantes!

- **dir**(objeto)
  - Retorna uma lista de atributos e métodos do objeto

- **help**(objeto)
  - Mostra a documentação do objeto

Variáveis e Tipos Básicos

- Atribuição:
  - nome_da_variavel = valor
- Inteiros, Inteiros Longos, Reais, Strings e Booleanos

```
>>> a = 10000000000000000000000000000000000000005646789467
>>> type(a)
<type 'long'>
```

```
>>> a = 4
>>> type(a)
<type 'int'>
```

```
>>> a = 5.3209
>>> type(a)
<type 'float'>
```

```
>>> a = True
>>> b = False
>>> type(a)
<type 'bool'>
>>> type(b)
<type 'bool'>
```

```
>>> a = 'texto'
>>> b = "texto"
>>> type(a)
<type 'str'>
>>> type(b)
<type 'str'>
```

Tente usar **dir**(a) e **help**(a)

Conversão dos Tipos Básicos

- int(), float(), str(), bool(), long()

```
>>> int(3.1415)
3
```

```
>>> float(3)
3.0
```

```
>>> str(25)
'25'
```

```
>>> str(True)
'True'
```

```
>>> bool(1)
True
```

```
>>> bool(0)
False
```

```
>>> bool(43)
True
```

```
>>> bool('lala')
True
```

```
>>> bool('')
False
```

```
>>> float(233333333333337777777777788888888899999999999933334421L)
2.3333333333333778e+56
```

```
>>> long(2.56)
2L
```

Teste no interpretador!

CUIDADO!

- Cuidado com os binários, octais e hexadecimais

```
>>> a = 0b10
>>> a
2
```

Binário

```
>>> a = 032
>>> a
26
```

Octal

```
>>> a = 0x32
>>> a
50
```

Hexadecimal

```
>>> bin(2)
'0b10'
```

```
>>> oct(26)
'032'
```

```
>>> hex(50)
'0x32'
```

Operadores Aritméticos

- +, -, *, /, //, **, %

Divisão com números inteiros resulta em um número inteiro

// - divisão inteira
** - exponenciação
% - resto da divisão

Teste no interpretador

```
>>> a = 2
>>> b = 3
>>> a + b
5
>>> a - b
-1
>>> a * b
6
>>> a / b
0
>>> a // b
0
>>> a ** b
8
```

```
>>> a = 2
>>> b = 3.5
>>> a + b
5.5
>>> a - b
-1.5
>>> a * b
7.0
>>> a / b
0.5714285714285714
>>> a // b
0.0
>>> a ** b
11.313708498984761
```

```
>>> 4 // 1.3
3.0
>>> 10 // 1.3
7.0
>>> 10 // 3.3
3.0
```

```
>>> 10%3
1
>>> 10%2
0
>>> 5%3
2
```

Python como calculadora

```
>>> 2+2
4
>>> # This is a comment
... 2+2
4
>>> 2+2  # and a comment on the same line as code
4
>>> (50-5*6)/4
5
>>> # Integer division returns the floor:
... 7/3
2
>>> 7/-3
-3
```

```
>>> width = 20
>>> height = 5*9
>>> width * height
900
```

Operadores Lógicos

- and, or, not

```
>>> True and True
True
>>> True and False
False
>>> False and False
False
>>> 1 and True
True
>>> 0 and True
0
```

```
>>> True or True
True
>>> False or True
True
>>> False or False
False
>>> 1 or False
1
>>> 0 or False
False
```

```
>>> not True
False
>>> not False
True
>>> not 1
False
```

```
>>> not ((a or False) and (False or False))
True
```

Operadores Relacionais

- >, <, >=, <=, ==, !=, <>

!= e <> significam “diferente”

É possível usar vários operadores na relação:
1<2<=3<4<5>4>3>=2!=1

- indica um comentário

```
>>> 2 > 3
False
>>> 2 < 3
True
>>> 3 >= 3
True
>>> 4 <= 3
False
```

```
>>> 2 == 2
True
>>> 2 == 1
False
>>> 3 != 2
True
>>> 3 != 3
False
>>> 3 <> 3
False
>>> 2 <> 2
False
```

```
>>> x = 3
>>> 2 < x < 4
True
>>> 7 > x > 1
True
>>> 3 <= x <4
True
```

Manipulação de Strings

- Podemos representar Strings com (') ou (' ')

```
>>> 'spam eggs'
'spam eggs'
>>> 'doesn\'t'
"doesn't"
>>> "doesn't"
"doesn't"
>>> '"Yes," he said.'
'"Yes," he said.'
>>> "\"Yes,\" he said."
'"Yes," he said.'
>>> '"Isn\'t," she said.'
'"Isn\'t," she said.'
```

Textos em várias linhas

Definição:

```
hello = "This is a rather long string containing\n\
several lines of text just as you would do in C.\n\
    Note that whitespace at the beginning of the line is\
 significant."

print hello
```

Resultado:

```
This is a rather long string containing
several lines of text just as you would do in C.
    Note that whitespace at the beginning of the line is significant.
```

Operações com Strings

- Operações:
  - + : Concatenação
  - * : Replicação

```
>>> st = ‘curso '
>>> st + ‘python'
‘curso python'
```

```
>>> st = ‘Casa’
>>> st*3
‘CasaCasaCasa'
```

  - str[i]: retorna o caracter de índice i da string str;
  - str[inicio:fim] retorna uma substring de str

```
>>> st = 'arquivo.mp3'
>>> st[0]
'a'
>>> st[-1]
'3'
>>> st[-4]
'.'
```

```
>>> st = 'arquivo.mp3'
>>> st[2:]
'quivo.mp3'
>>> st[0:-4]
'arquivo'
>>> st[-3:]
'mp3’
```

Teste no interpretador

Conhecendo as Strings

dir(‘string ou variavel’)

```
>>> dir('string')
['__add__', '__class__', '__contains__', '__delattr__', '__doc__', '__eq__', '__format__',
'__ge__', '__getattribute__', '__getitem__', '__getnewargs__', '__getslice__', '__gt__',
'__hash__', '__init__', '__le__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__',
'__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__',
'__setattr__', '__sizeof__', '__str__', '__subclasshook__', '_formatter_field_name_split',
'_formatter_parser', 'capitalize', 'center', 'count', 'decode', 'encode', 'endswith',
'expandtabs', 'find', 'format', 'index', 'isalnum', 'isalpha', 'isdigit', 'islower', 'isspace',
'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'partition', 'replace', 'rfind', 'rindex', 'rjust',
'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title',
'translate', 'upper', 'zfill']
```

Métodos de Strings

- **split**(char)
  - Retorna uma lista com os elementos separados por char

```
>>> a = '1+2+3+4+5+6'
>>> a.split('+')
['1', '2', '3', '4', '5', '6']
```

- **len**(str)
  - Retorna o tamho da string str

```
>>> a = 'curso de python'
>>> len(a)
15
```

- **strip**(chars)
  - Retorna uma string onde os chars da direita e da esquerda foram removidos

```
>>> a = '  !!! STRING DE RETORNO!     !! !  !'
>>> a.strip(' !')
'STRING DE RETORNO'
```

Métodos de Strings

- **find**(substring)
  - Retorna a posição da primeira ocorrência da substring.

    Caso não seja encontrada, retorna -1

```
>>> a = 'curso de python'
>>> a.find('de')
6
```

```
>>> a = 'curso de python'
>>> a.find('casa')
-1
```

- **lower**(), **upper**()
  - Retornam uma string em minúsculo/maiúsculo

```
>>> a = 'CuRsO dE pYtHoN!'
>>> a.lower()
'curso de python!'
```

```
>>> a = 'CuRsO dE pYtHoN!'
>>> a.upper()
'CURSO DE PYTHON!'
```

Métodos de Strings

Procurem mais métodos de string com **dir**() e **help**()

Exercícios

- A partir da string “!! ! a;b;c;d;e;f;gh!########” gere o resultado:
  - [“a”, “b”, “c”, “d”, “e”, “f”, “g”]

- A partir da string “ring ring! - hello!“ gere o resultado:
  - “hello!”

- Transformar a string “isso deve ser bom” para “Isso Deve Ser Bom”
- Transformar a string “abacate azul” em “4b4c4te 4zul”

Dica: para os dois últimos, pesquise os métodos de string usando dir()

Exercícios

- A partir da string “!! ! a;b;c;d;e;f;gh!########” gere o resultado:
  - [“a”, “b”, “c”, “d”, “e”, “f”, “g”]

```
string = '!! ! a;b;c;d;e;f;gh!##########'
print string.strip(' !#h').split(';')
```

Exercícios

- A partir da string “ring ring! - hello!“ gere o resultado:
  - “hello!”

```
string = 'ring ring! - hello!'
print string[string.find('hello'):]
```

OU

```
string = 'ring ring! - hello!'
print string[13:]
```

Exercícios

- Transformar a string “isso deve ser bom” para “Isso Deve Ser Bom”

```
string = 'isso deve ser bom'
print string.title()
```

- Transformar a string “abacate azul” em “4b4c4te 4zul”

```
string = 'abacate azul'
print string.replace('a', '4')
```

Difícil?

Desvio Condicional

```
if condicao:
    comandos
elif condicao:
    comandos
else:
    comandos
```

```
>>> a = input('Digite um numero: ')
Digite um numero: 4
>>> if a%2 == 0:
...         print "Par"
...   else:
...         print "Impar"

Par
```

Não precisa de (parêntesis), mas podem ser usados

Desvio Condicional

- Não existe SWITCH/CASE, quando é necessário várias comparações, usamos **elif's** ou **dicionário** (explicado mais à frente)
- Pode ser escrito em uma linha:
  - SETRUE **if** condição **else** SEFALSE

```
>>> numero = input('entre com um numero: ')
entre com um numero: 5
>>> print 'par' if numero%2 == 0 else 'impar'
impar
```

Exercícios

- Leia dois números e imprima o maior deles
- Leia uma letra, se for “M” imprima “Masculino, se for “F” imprima “Feminino”, senão imprima “Sexo invalido”.

DICA:

variavel = input() *#Para números*

variavel = raw_input() *#Para strings*

Exercícios

- Leia dois números e imprima o maior deles

```
numero1 = input('Numero 1: ')
numero2 = input('Numero 2: ')
if numero1 > numero2:
    print numero1
elif numero1 < numero2:
    print numero2
else:
    print 'iguais'
```

Exercícios

- Leia uma letra, se for “M” imprima “Masculino, se for “F” imprima “Feminino”, senão imprima “Sexo invalido”.

```
letra = raw_input('Sexo: ')
if letra == 'M':
    print 'Masculino'
elif letra == 'F':
    print 'Feminino'
else:
    print 'Sexo invalido'
```

Loops

- O **While** é *quase* igual às outras linguagens:

while condição:
    comandos
else:
    comandos

```
>>> x = 0
>>> while x < 10:
        print x,
        x += 1

0 1 2 3 4 5 6 7 8 9
```

Loops

- O **While** é *quase* igual às outras linguagens:

while condição:
    comandos
else:
    comandos

```
>>> x = 0
>>> while x < 10:
        print x,
        x += 1

0 1 2 3 4 5 6 7 8 9
```

Loops

- O **else** só é executado se não ocorreu nenhum break durando o while, ou seja, se o while parar pela condição.

```
>>> numero = input('entre com um numero: ')
entre com um numero: 11
>>> x = 2
>>> while x < numero:
        if numero%x == 0:
            print 'nao é primo'
            break
        x += 1
    else:
        print numero, 'é primo‘

11 é primo
```

Loops

- O for em python é parecido com o *foreach* de outras linguagens:

for variavel **in** sequencia:

    comandos

else:

    comandos

```
>>> for letra in 'string':
        print letra

s
t
r
i
n
g
```

Loops

- Trabalhando com listas: [...].

```
>>> for elemento in [1,3,3,6,4,2,7]:
        print elemento*2


2
6
6
12
8
4
14
```

Loops

- a função **range**(numero) retorna uma lista de números

```
>>> range(10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
>>> range(5,10)
[5, 6, 7, 8, 9]
```

```
>>> range(0,10,2)
[0, 2, 4, 6, 8]
```

```
>>> for i in range(10):
        print i,

0 1 2 3 4 5 6 7 8 9
>>> for i in range(5,10):
        print i,

5 6 7 8 9
>>> for i in range(0,10,2):
        print i,

0 2 4 6 8
```

Exercícios

- Imprima a tabuada de multiplicação de um número digitado pelo usuário;
- Some os números impares de 0 a 100;
- Imprima uma frase digitada pelo usuário, continue lendo e imprimindo até o usuário digitar SAIR

Exercícios

- Imprima a tabuada de um número digitado pelo usuário

```
numero = input('Entre com um numero: ')
for i in range(11):
    print numero, 'x', i, '=', numero*i
```

- Imprima a soma dos números impares de 0 a 100

```
soma = 0
for i in range(1, 100, 2):
    soma += i
print soma
```

Exercícios

- Imprima a tabuada de um número digitado pelo usuário

```
numero = input('Entre com um numero: ')
for i in range(11):
    print numero, 'x', i, '=', numero*i
```

- Imprima a soma dos números impares de 0 a 100

```
soma = 0
for i in range(1, 100, 2):
    soma += i
print soma
```

OU

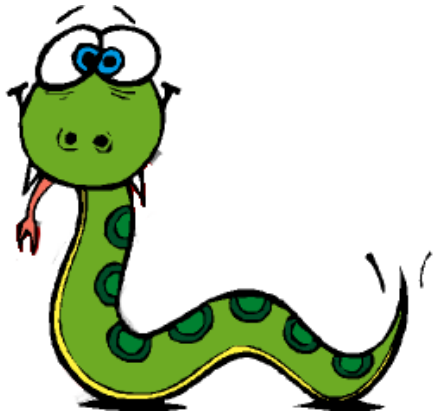
```
sum(range(1,100,2)
```

Exercícios

- Imprima uma frase digitada pelo usuário, continue lendo e imprimindo até o usuário digitar SAIR

```
while True:
    frase = raw_input('Frase: ')
    if frase == 'SAIR': break
    print 'FRASE: ', frase
```

Não tem problema usar *"While True"* em python! No exemplo o *"While True"* evita repetição de código!



Funções

- Para criar uma função:
  - **def** nome_da_funcao(parametros):

- Para retornar algum valor, usamos o **return**
- Uma função não precisa retornar algo.
- Não existe sobrecarga

```
>>> def fat(n):
        return 1 if n < 2 else fat(n-1)*n

>>> fat(5)
120
```

Funções

- Valor padrão de um parâmetro.
- Substitui a sobrecarga e evita repetição de código

```
def funcao(a, b=5):
        print 'a =',a
        print 'b =',b
```

```
>>> funcao(2)
a = 2
b = 5
>>> funcao(2,6)
a = 2
b = 6
>>> funcao(b=9, a=4)
a = 4
b = 9
```

```
public void funcao(int a) {
   System.out.println('a = '+a);
   System.out.println('b = 5');
}

public void funcao(int a, int b) {
   System.out.println('a = '+a);
   System.out.println('b = '+b);
}
```

Exercícios

- Faça uma função para calcular o fatorial de um número (função não recursiva)
- Faça uma função para verificar se um número é primo ou não

Exercícios

- Faça uma função para calcular o fatorial de um número (função não recursiva)

```
def fatorial(n):
    resultado = 1
    for i in range(2, n+1):
        resultado *= i
    return resultado
```

Exercícios

- Faça uma função para verificar se um número é primo ou não

```
def primo(n):
    for i in range(2, n):
        if n%i == 0:
            return False
    return True
```

Listas

- Não existe vetores em Python, nós usamos **listas**
- Uma lista guarda uma seqüência de itens
- São identificadas por colchetes “[ ]”

```
>>> lista = [1,2,3,4,5]
>>> print lista
[1, 2, 3, 4, 5]
```

- Os itens podem ser acessados como um vetor

```
>>> lista[0]
1
>>> lista[4]
5
```

Listas

Inserindo elementos

- **append**(obj)
  - Insere o objeto no fim da lista

```
>>> lista = [1,2,3]
>>> lista.append(0)
>>> lista
[1, 2, 3, 0]
```

- **insert**(posição, obj)
  - Insere o objeto na posição especificada

```
>>> lista = [1,2,3]
>>> lista.insert(1, 'objeto')
>>> lista
[1, 'objeto', 2, 3]
```

Listas

Removendo elementos

- **remove**(obj)
  - Remove o objeto da lista

```
>>> lista = ['a','b',3]
>>> lista.remove('b')
>>> lista
['a', 3]
```

- **pop**(posição)
  - Remove e retorna o objeto da posição especificada. O pop() sem parâmetros retira o último elemento

```
>>> lista = ['a', 2, 'b', 4, 'c']
>>> lista.pop(0)
'a'
>>> lista
[2, 'b', 4, 'c']
>>> lista.pop()
'c'
>>> lista
[2, 'b', 4]
```

Listas

Outros métodos

- **count**(obj)
  - Retorna a quantidade de elementos “obj” da lista
- **index**(obj)
  - Retorna a posição do “obj” na lista
- **reverse**()
  - Inverte a lista colocando os elementos ao contrário
- **sort**()
  - Ordena a lista

Usem dir() e help()

Tuplas

- Definidas por parêntesis “( )” ou objetos divididos por “,”

```
>>> tupla = (1,2,3)
>>> tupla
(1, 2, 3)
```

```
>>> tupla = 1,2,3
>>> tupla
(1, 2, 3)
```

- Parecidos com listas, mas são imutáveis

```
>>> lista = [1,2,3]
>>> lista[0] = 4
>>> lista
[4, 2, 3]
```

```
>>> tupla = (1,2,3)
>>> tupla[0] = 4
Traceback (most recent call last):
  File "<pyshell#40>", line 1, in <module>
    tupla[0] = 4
TypeError: 'tuple' object does not support item assignment
```

Dicionários

- O programador pode definir a chave do item
- Representados por chaves “{ }”

```
>>> dicionario = {'chave' : 'item', 2 : 4}
>>> dicionario['chave']
'item'
>>> dicionario[2]
4
```

Use dir() e help()

Dicionários

- Inserir item
  - dicionario[chave] = obj
- Remover item
  - dicionario.**pop**(chave)
    - Se não encontrar a chave retorna erro
  - dicionario.**pop**(chave, retorno)
    - Se não encontrar a chave retorna o valor de retorno
- Remover todos itens
  - dicionario.**clear**()

Dicionários

- Acessar item
  - dicionario[chave]
    - Se a chave não existir retorna erro
  - dicionario.**get**(chave)
    - Se a chave não existir retorna None
  - dicionario.**get**(chave, retorno)
    - Se a chave não existir retorna o valor de retorno
- Verificar se existe item
  - dicionario.**has_key**(chave)

Exercícios

- Função para calcular série Fibonacci até n
- Imprimir quantos números são maiores que 10 na lista:
  - [1, 4, 15, 26, 3, 6, 10, 13, 5, 3]
- Leia o nome e a nota de 5 alunos e guarde em um dicionário
- Função que receba uma string é imprima em escada, ex:
  - T
  - TE
  - TES
  - TEST
  - TESTE

DICA: Use a função len(objeto) para pegar o tamanho da string

Exercícios

- Calcular série Fibonacci até n

```
def fibonacci(n):
    fibs = [0, 1]
    for i in range(n-2):
        fibs.append(fibs[-2] + fibs[-1])
    return fibs
```

Exercícios

- Verificar quantos números são maiores que 10 na lista:
  - [1, 4, 15, 26, 3, 6, 10, 13, 5, 3]

```
lista = [1, 4, 15, 26, 3, 6, 10, 13, 5, 3]
soma = 0
for item in lista:
    if item > 10:
        soma += 1
print soma
```

Exercícios

- Leia o nome e a nota de 5 alunos e guarde em um dicionário

```
notas = {}
for i in range(5):
    nome = raw_input('nome: ')
    nota = input('nota: ')
    notas[nome] = nota
print notas
```

Exercícios

- Função que receba uma string é imprima em escada, ex:
  - T
  - TE
  - TES
  - TEST
  - TESTE

```
def funcao(palavra):
    for i in range(len(palavra)):
        print palavra[:i+1]
```

FIM

Contatos

- Márcio Palheta
  - marcio.palheta@gmail.com
  - @marciopalheta
  - http://sites.google.com/site/marciopalheta

Bibliografia

- LIVRO: Apress - Beginning Python From Novice to Professional
- LIVRO: O'Relly - Learning Python
- http://www.python.org
- http://www.python.org.br

- Mais exercícios:
  - http://www.python.org.br/wiki/ListaDeExercicios

- Documentação do python:
  - http://docs.python.org/

Introdução ao Python

Python

Márcio Palheta
marcio.palheta@gmail.com