SeTel 2010

Semana de Telecomunicações da UFF

Software Livre e Desenvolvimento Ágil

Organização

PET Tele

Apoio

Minicurso de Linguagem de Programação Python.

Informações Iniciais:

- Objetivos do Curso;
- Linguagem de Programação;
- Histórico do Python;
- Principais características;
- Vantagens e desvantagens;
- Principais Aplicações;
- Quem usa Python;
- Obtenção e Instalação;
- Variáveis;
- Operações matemáticas;
- Entrada de dados;
- Listas;
- Estruturas de controle;
- Dicionário;
- Funções;
- Módulos;
- Expressões booleanas.

Objetivos do Curso:

- Mostrar de forma simples e clara o que é a linguagem Python, suas diversas aplicações e suas principais características.

Linguagem de Programação:

- Pensamento X Linguagem;
- Código Binário;
- Linguagem de Programação;
- Linguagens de Programação: Pontes;
- Tipos: Baixo Nível e Alto Nível.

Histórico do Python:

- Criado em 1989 pelo holandês Guido van Rossum;
- Herdou muitas características da Linguagem ABC;
- Necessidade de uma linguagem de alto nível para aplicações em projetos do Grupo Amoeba;
- Atualmente Python é mantido pela *Python Software Foundation,* sendo 3.1 a versão mais atual, coberta pela licença *GPL (GNU Public License).*

Principais Características:

- Fácil aprendizagem;
- Delimitação de bloco por identação;
- Interpretada;
- Possui tipagem dinâmica;
- É de alto nível;
- Orientada a objeto;
- Multiplataforma.

Vantagens:

- Código de leitura mais amigável;
- Maior rapidez no desenvolvimento;
- Pequena curva de aprendizagem;
- Módulos expandem as funcionalidades da linguagem;
- MultiPlataforma ;
- Portabilidade das aplicações.

Desvantagens:

- Requer um interpretador.
- Mais lento do que linguagens compiladas.
- Pouca popularidade.
- É necessário distribuir o interpretador junto com o programa ,caso o sistema não disponha de um.

Principais Aplicações:

- Aplicações Acadêmicas;
- Desenvolvimento de Páginas da Internet;
- Computação Gráfica (Jogos, Filmes e Animações);
- Monitoramento de Redes;
- Construção de Aplicativos;
- Aplicações Matemáticas.

Empresas que usam Python:

Obtenção e Instalação:

- Windows:
  - *Download* do IDLE em http://www.python.org
- Linux:
  - Terminal > “python”

Orientação a Objeto:

- Forma conceitual de se estruturar um programa.
- Objeto = atributos (variáveis) + métodos (funções).

Sintaxe:

objeto.método(argumentos)

Variáveis:

- Classificadas em três tipos:
  - *int* – um número inteiro;
  - *float* – um ponto flutuante;
  - *string* – uma sequência de caractéres.
- Variáveis não precisam ser declaradas:

Exemplos:

a=3 b=3.0 c="olá"

Strings:

- Sequência imutável de caracteres que permitem o trabalho com textos:

Exemplo:

```
>>> a=“Bom dia”
>>> print a
Bom dia
```

Strings (cont.):

Exemplo 2:

>>>b=‘O lema do governo JK era:\n “
Cinquenta anos em cinco”’

>>>print b

O Lema do governo JK era:

“Cinquenta anos em cinco”

Strings (cont.):

- Caracter de formatação: \t
- Mais uma aplicação para as aspas:
  - d= '''Times do Rio:

Botafogo

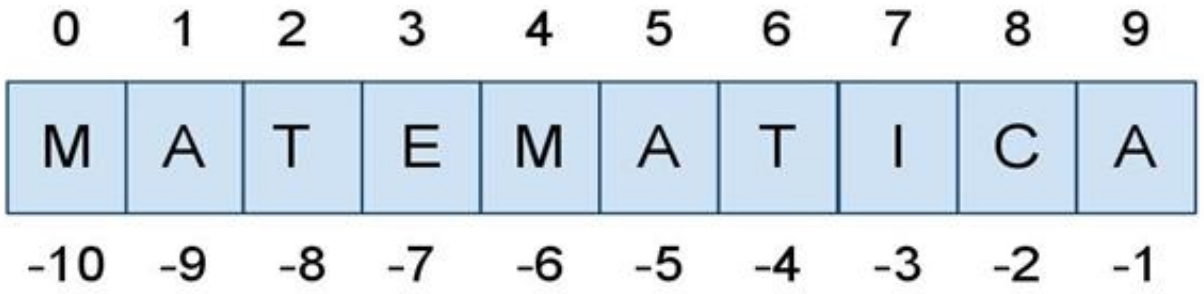
Vasco

Fluminense

Flamengo'''

Strings (cont.):

- Manipulação de Strings:
  - Indexação:



❑ Considerando a palavra acima, construir uma nova palavra. Exemplo: TEMA

Strings (cont.):

- Operador %:
  - %s – serve para substituir uma string;
  - %d – serve para subtituir números inteiros em frases destinadas a um print;
  - %f – serve para substituir floats.

Exemplo:

```
>>>compra=‘maça’
>>>tipo=‘verde’
>>>quilos=1.5
>>>print ‘Maria comprou %f quilos de %s %s.’ %(quilos,compra,tipo)
```

Operações Matemáticas:

- Soma (+);
- Subtração (-);
- Multiplicação (*);
- Divisão (/)
- Potenciação (**);
    - Radiciação (**x/y).

Entradas de Dados:

Comandos:

- raw_input();
- len();
- input();
- type();

Listas:

- Sequências de caracteres mutáveis. Sua indexação segue a mesma das strings.

Exemplo:

```
>>>lista=[7,8,9]
>>> print lista[0]
7
```

Listas (cont.):

- Alguns comandos e métodos referentes às listas:
  - len(lista)
  - lista.append(x)
  - lista.extend([4,5,6])
  - lista[y]=x

OBS.: comando for.

Exemplo:

>>>for valor in lista

Listas (cont.):

- Mais alguns comandos referentes às listas:
  - del lista[x];
  - lista.remove(y);

OBS.: função range.

Exemplo:

>>>vetor = range(início,fim+1,passo)

Exercícios:

1) Faça um programa que leia 4 notas, mostre as notas e a sua média na tela.

2) Faça um programa que leia um vetor de 3 números inteiros, escolhidos pelo usuário e mostre-os em seguida.

3) Faça um programa que leia dois vetores com 3 elementos cada. Gere um terceiro vetor de 6 elementos, cujos valores deverão ser compostos pelos elementos intercalados dos dois outros vetores.

Estruturas de Controle:

- Permitem que o programador modifique a ordem em que cada comando será executado, bem como se ele será ou não executado.

Estruturas de Controle (cont.):

1) Estrutura If:

Direciona o computador a uma tomada de decisão de acordo com as condições pré-estabelecidas.

Sintaxe:

```
... #bloco de comandos1
>>>if #condição1:
                    #bloco de comandos 2
... (continuação do programa)
...                 # bloco de comandos3
```

Estruturas de Controle (cont.):

2) Estrutura While:

Responsável pela execução de um determinado comando ou conjunto de comandos, enquanto uma determinada condição seja verdadeira.

Sintaxe:

```
>>>while #condição for verdadeira:
        #bloco de comandos pertencentes ao while

>>> # continuação do programa
```

Estruturas de Controle (cont.):

Exercício 4:

Faça um programa que o usuário tenha a tarefa de adivinhar um número determinado pelo programador. A cada escolha que o usuário faz, o programa deve dizer se o número escolhido está acima ou abaixo do verdadeiro valor. É necessário, também, avisar quando o usuário acertar na escolha do número.

Estruturas de Controle (cont.):

Possibilidade de Solução:

```
>>>num=23
>>>adv=0
>>>while adv!=num:
        adv=input('Insira um número')
        if adv<num:
                print 'É maior!'
        elif adv>num:
                print 'É menor!'
        else:
                print 'Você acertou!'
```

Estruturas de Controle (cont.):

Exercício 5:

Faça um programa que calcule o fatorial de um número.

Estruturas de Controle (cont.):

Possibilidade de Solução:

```
>>>Resp=1
>>>num=input('Entre com um número inteiro:')
Entre com um número inteiro: 6
>>> while num>1:
                    resp=resp*num
                    num=num-1
>>> print num,'! é igual a ', resp
```

Estruturas de Controle (cont.):

Execício 6:

Faça um programa que já possua uma lista com 4 nomes e que mostre na tela esses nomes e suas respectivas quantidades de letras.

Estruturas de Controle (cont.):

Possibilidade de Solução:

```
>>> nomes=['Carlos', 'Joana', 'Lara', 'Alex']
>>> for x in nomes:
        print '%s tem %i letras' %(x,len(x))
```

```
Carlos tem 6 letras
Joana tem 5 letras
Lara tem 4 letras
Alex tem 4 letras
```

Dicionário:

- É um conjunto de pares delimitados por chaves. São eles, as chaves e os valores. Para cada chave temos um valor correspondente.

Formatação:

```
>>> dicionário = {chave1:valor1, chave2:’valor2’}
```

Dicionário (cont.):

Exemplo:

>>> calculo={1:’primeiro período’, 2: ‘segundo período’, 4:’terceiro período’, 8:’quinto período’}

>>> print calculo

{1:’primeiro período’, 2:’segundo período’, 4: terceiro período, 8:’quinto período’}

>>>calculo[4]=‘números complexos’

Dicionário (cont.):

Exercício 7:

Faça um dicionário que contenha suas refeições e um alimento que esteja contido em cada uma delas. Mostre na tela. Após isso, mude os alimentos pelos seus alimentos favoritos.

Dicionário (cont.):

- Alguns Métodos dos dicionários:
  - .items();
  - .keys();
  - .values();
  - .get(chave);
  - .has_key(chave);
  - .update(dicionario).

Funções:

- Tem por objetivo agilizar o processo de construção de um programa, pois se torna responsável em executar uma ação que será usada muitas vezes num mesmo programa.

Sintaxe:

```
>>> def func(argumento):
        #bloco de controle
        return
```

Funções (cont.):

Exemplo:

```
>>>def raiz_quadrada(x):
        return x**(1/2)
```

Existe algum erro??

Funções (cont.):

Exercícios:

8) Faça um programa, com uma função que necessite de três argumentos, e que forneça a soma e o produto desses três argumentos.

9) Faça um programa, com uma função que calcule e forneça o resultado de um número “n” elevado a outro “m”. Ambos deverão ser escolhidos pelo usuário.

10) Faça um programa, com uma função que necessite de um argumento. A função retorna o valor do caractere “P”, se seu argumento for positivo, e “N”, se seu argumento for negativo e “Z” caso seja zero.

Módulos:

- Módulos são programas para serem reaproveitados futuramente.
- Eles contêm funções, classes e objetos funcionais para a criação de um novo programa.
- É necessário importar o módulo através do comando**:**

import *nome_do _modulo*

- Para utilizarmos uma função do módulo utilizamos:

from *nome_do_modulo* import *funcao*

Módulos (cont.):

- Módulo Math:

Tabalha com funções matemáticas.

Exemplo:

- math.factorial(x);
- math.exp(x);
- math.log(x,base);
- math.sqrt(x);
- math.degrees(x);
- math.radians(x).

Módulos (cont.):

Criando um Módulo:

• Para criarmos um módulo, precisamos colocá-lo num diretório do sistema.

• Como saber quais os diretórios estão disponíveis?

```
>>>import sys
>>>for i in sys.path:
        print i
```

Módulos (cont.):

Criando um Módulo:

• Criar, num editor de texto de sua escolha, um arquivo com extensão .py que contenha as funções pertencentes ao módulo.

• Testar o módulo:

```
>>> import nome_do_módulo
>>>  nome_do_módulo.função(paramentro1, parametro2)
```

Expressões booleanas:

- Sentenças lógicas que seguem as leis da Álgebra de Boole.
- Álgebra de Boole trabalha com valores lógicos, sendo uma operação FALSA representada pelo valor 0, e uma operação VERDADEIRA representada pelo valor 1.

Expressões booleanas (cont.):

- Alguns operadores lógicos:

- Analogia com circuitos elétricos.

expression	result
true and true	true
true and false	false
false and true	false
false and false	false
not true	false
not false	true
true or true	true
true or false	true
false or true	true
false or false	false

Tabela1: Alguns operadores lógicos.

Expressões booleanas (cont.):

- Alguns testes:

```
>>> a=6
>>> b=7
>>> c=42
>>> print 1,a==6
>>> print 2,a==7
>>> print 3,a==6 and b==7
>>> print 4,a==7 and b==7
>>> print 5, not a==7 and b==7
>>> print 6, a==7 or b==7
>>> print 7, a==7 or b==6
>>> print 8, not(a==7 and b==6)
>>> print 9, not a==7 and b==6
```

Considerações Finais:

- Python;
- UFF;
- PET-Tele.

Contatos:

Website: http:// *www.telecom.uff.br/pet/*

E-mail: pet.tele@telecom.uff.br

Local: Bloco E, sala 249 , Tel: (21) 2629-5606

Obrigado!