December 16, 2009

1 Lista de exercícios do minicurso de python

1.1 Instale softwares

Instale o ipython no seu computador.

Sistema operacional Windows: Se você está no grupo de sistemas complexos e usa windows você não deve ser um cara muito esperto, por isso vou lhe passar o passo a passo: instalar o python (sugiro a versão 2.6).

`http://www.python.org/download/`

instalar o ipython.

`http://ipython.scipy.org/moin/`

instale o PyReadline (habilita a função de tab-complete no ipython).

`http://ipython.scipy.org/moin/PyReadline/Intro`

mais ajuda para usuários do windows:

`http://www.python.org/doc/faq/windows/`

Sistema operacional linux: O teu sistema já possui python, bastar instalar o pacote do ipython. apt-get install ipython no debian, ou pacmam -S ipython no arch)

Instale um bom editor de texto: No windows eu recomendo o Scite:

`http://www.scintilla.org/SciTE.html`

Ou notepad++:

`http://notepad-plus.sourceforge.net/`.

No linux, se você não usa o emacs ou vi, sugiro começar com o gedit. Para quem usa IDEs o code::blocks, eclipse e netbeans possuem plugins para python.

1.2 usando o python:

Abra o ipython e aprenda a usar a função help(). Basicamente mande help('node_da_função') e você obtem o help daquela função. Por exemplo. help('print') ou help('sum').

Se você digitar:

>>> zipcode = 02492[1]

O python retornará:

SyntaxError: invalid token

Se você digitar no terminal:

[1] vamos usar o símbolo $ para identificar um comando no terminal e o símbolo >>> para identificar um comando dentro ipython

```
>>> zipcode = 02132
>>> print zipcode
1114
```

Vc é capaz de explicar o que está ocorrendo?

1.3 escrevendo scripts:

Crie um arquivo converte.py, e copie o seguinte código: # Início do script

```
# -*- coding: utf-8 -*-
def bin2dec(n):
    """
    Recebe uma string em binário e converte para um inteiro em hexadecimal
    """
    raise NotImplementedError('Essa função ainda não foi implementada!')
def dec2bin(n):
   """
   Recebe um decimal e converte uma string em binário
   """
   raise NotImplementedError('Essa função ainda não foi implementada!')
if __name__ == "__main__":
    print 'Esse módulo foi chamado como main!'
# Final do script
```

Salve esse código no mesmo diretório no qual você chamou o ipython.
Agora no ipython:

```
>>> import converte
>>> converte.dec2bin?
>>> dec2bin('1010')
```

Entenda o output de cada um desses comandos. Feche o ipython. Abra o ipython novamente e mande:

```
>>> from converte import dec2bin
>>> dec2bin?
```

Pesquise no google e entenda a diferença entre os comandos **import modulo** e **from modulo import função** No terminal do linux (ou no prompt de comando do windows), digite:

```
$ python converte.py
```

Entenda o que ocorreu.
Pesquise no google o que significa as linhas:

```
# -*- coding: utf-8 -*-
raise NotImplementedError('Essa função ainda nao foi implementada!')
if __name__ == "__main__":
```

Finalmente, utilizando a função int() e bin(), escreva código para implementar as funções bin2dec e dec2bin.

1.4 Código modular, funções, tratamento de erros e passagem de parâmetros.

Escreva uma script que imprime:

```
+ - - - - + - - - - +
I         I         I
I         I         I
I         I         I
I         I         I
+ - - - - + - - - - +
I         I         I
I         I         I
I         I         I
I         I         I
+ - - - - + - - - - +
```

Dica: para imprimir sem pular linha: print '+', '-'

Generalize o seu código para imprimir um grid NxN. Dica: divida seu código em pequenas funções especializadas em gerar cada linha da figura, e faça loops para gerar todas a figura final.

Aprenda a usar o modulo sys `http://docs.python.org/library/sys.html` para passar o argumento por linha de comando. Dica: Faça um script teste_sys.py com o código: # inicio do código import sis print sys.argv #fim do código E

chame-o no terminal. $ python teste_sys.py 1 2 3 aloalo Pegue o seu script que

recebe o argumento por linha de comando e mande: $python print_grid.py o

python deve retornar: Traceback (most recent call last):

```
File "print_grid.py", line 34, in <module>
N = int(sys.argv[1])
IndexError: list index out of range
```

Ou então, podemos fazer algo pior:

$python print_grid.py a

Traceback (most recent call last):
File "print_grid.py", line 34, in <module>
N = int(sys.argv[1])
ValueError: invalid literal for int() with base 10: 'a'

Aprenda sobre tratamento de erros em `http://docs.python.org/tutorial/errors.html` E escreva um bloco try / except para transformar essas mensagens de erros do python em algo claro para o usuário, como:

$ python print_grid.py
$ Erro: Idiota! Vc deve passar o tamanho do Grid!
$python print_grid.py a
$ Erro: Animal! O tamanho do grid deve ser um inteiro!

1.5 Embaralhamento de sequências.

Usando seus conhecimentos de geradores de números aleatórios, crie uma função que embaralha uma lista. Esse programa funciona para strings?

Implemente também um teste que certifica que esse embaralhamento é razoável. Dica: Use um dicionário para fazer o histograma das possíveis permutações.

1.6 Permutações de uma sequência.

a- Escreva uma programa que gera todas as permutações de uma lista. Esse programa funciona para strings? Esse programa possui menos de 10 linhas? dica: use recursividade e o comando yield estude: `http://docs.python.org/tutorial/classes.html` tópico: generators