Python

Estruturas de dados

Prof. Dr. Dieval Guizelini

Analista e Desenvolvedor de Sistemas

Mestre em Bioinformática e Doutor em Ciências-Bioquímica

dieval at ufpr.br / dievalg at gmail.com

Estrutura de Dados

- As estruturas disponíveis em Python são: listas (lists), registros (tuples) e dicionários (dictionaries – map)
- Conjuntos (sets) estão disponíveis na biblioteca sets (a partir da versão 2.5)
- As listas são como vetores/matrizes unidimensionais, mas pode-se utilizar listas de listas
- Os dicionários são matrizes associativas
- As tuplas são conjuntos de campos pré-definidos.

Sobre vetores

- O índice zero [0] indica o primeiro elemento
- Vetores são coleções unidimensionais de elementos de um mesmo tipo.
- Valores negativos, referencia a lista de trás para frente (-1 é o último elemento)
- Ex:
  v = [1, ["texto 1","lista"], ("a","registro")]
- Onde e[0] retorna 1
  e[1] retorna ["texto 1","lista"]
  e[1][1] retorna “lista”
  e[2][1] retorna “registro”

Listas

- É a estrutura mais versátil do Python
- Declaração:
  list1 = ['physics', 'chemistry', 1997, 2000];
  list2 = [1, 2, 3, 4, 5];
  list3 = ["a", "b", "c", "d"]
- Parecem com vetores, mas permitem elementos com diferentes tipos
- Para acessar os valores de cada elemento:
  list1[0] => physics
  list2[1:5] => [2, 3, 4, 5]
- Podem ter os valores modificados
  list[2] = 2001;
- Elementos podem ser removidos:
  del list[2]
- Funções:
  cmp(list1,list2)
  len(list1)
  max(list1),min(list1)
  list(seq)

Comprehensions (comportamento funcional)

- Utilizada para gerar listas...
- Sintaxe geral: [*expression* for *variable* in *list*]
- Exemplos:

```
myList=[0 for i in range(10)]      # [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
myList=[i for i in range(10)]      # [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
myList=[i*i for i in range(10)]    # [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]

myArray=[[0 for j in range(3)] for i in range(3)]
                                   # [[0, 0, 0], [0, 0, 0], [0, 0, 0]]
```

Registros (tuple)

- “tuples” são sequências de objetos “imutáveis”

- Criação

```
tup1 = (1,2,3)
tup2 = ()
tup2 = (1,)
tupA = ('physics', 'chemistry', 1997, 2000)
tupB = (1, 2, 3, 4, 5 )
tupC = "a", "b", "c", "d"
```

Tuplas: acessando os elementos

- Da mesma forma que vetores:
  print(tupB[0])
  print(tupB[1:3])
- Atualizando (criando novos registros a partir de um existente)
  tup3 = tup1 + tup2
- Excluindo tuplas
  del tup3
- Os operadores + e * funcionam como nas Strings
- Outras expressões:

len(tup1)	retorna o número de elementos
3 in (1,2,3)	retorna verdadeiro
for x in (1,2,3): print(x)	percorre um a um os elementos
cmp(tup1,tup2)	compara as duas tuplas

Dicionário

- Vetores associativos
- O índice do vetor é um nome que corresponde a um valor
- Exemplo:

```
dicio = {"Key 1": "Value 1", 2: 3, "pi": 3.14}
dicio[“Key 1”]    #  retorna “Value 1”
dicio[2]              # retorna 3
dicio[“pi”]          # retorna 3.14
```

Funções para dicionários

Função/Método	Finalidade
d.clear()	Limpar o dicionário
d.copy()	Copiar o dicionário
d.fromkeys(seq [,values])	Cria um novo dicionário
d.get(key,default=None)	Retorna o valor de um dicionário ou “default”
d.has_key(key)	Retorna VERDADEIRO se tiver uma chave
d.items()	Retorna uma lista de tuplas (key,value)
d.keys()	Retorna uma lista de chaves
d.setdefault(key,default=None)	Retorna o valor do item com a chave especificada. Se a chave não existir, insira a chave, com o valor indicado.
d.update(dict2)	Atualiza o dicionário
d.values()	Retorna uma lista com os valores

Formatando a saída

- Chamamos de formatação o recurso que permite controlar a “forma” que o conteúdo das variáveis são apresentadas em tela.
- A função de saída padrão é o print e a sintaxe geral é:

```
Type "help", "copyright", "credits" or "license()" for more information.
>>> print(
        print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)
```

- Onde:
  - value pode ser um variável (objeto) ou uma String com a configuração da formatação.
  - ... O conjunto de variáveis (opcional)
  - E os parâmetros nomeados, sep, end, file e flush, respectivamente, indicam a string de separação entre os valores das variáveis passadas anteriormente, o end indica o símbolo a impresso após o conteúdo informado, file o nome do arquivo ou o dispositivo padrão de saída (sys.stdout) e se o buffer deve ser esvaziado após a execução da função.
- O Python utiliza o estilo do C para formatação, em outras palavras, utilizamos os %<algo> para interpolar o conteúdo das variáveis com a String de formatação.

Exemplos de utilização:

- Para os exemplos abaixo, considere as definições de variáveis abaixo:

```
nome = ‘Mr Bean’
numMensagens = 10
tab = list()
tab.append( ('João',  5,  9.2) )
tab.append( ('Lucas', 7,  9.3) )
tab.append( ('Pedro', 3,  4.2) )
```

Criando algumas variáveis para nossos exemplos.
nome e numMensagens são, respectivamente, do tipo String e int.
A variável tab é uma estrutura de dados – list.

- Apresentando o conteúdo de uma variável:

```
print("Bom dia, %s!" % nome )
```

%s indica a posição em que o conteúdo da variável nome será inserido na String.

- Trabalhando com duas ou mais variáveis:

```
print("%s - %d" % (nome,numMensagens)      )
Mr Bean - 10
```

Observe a presença do %s e do %d na String de formatação, respectivamente, eles representam uma string e um inteiro decimal.

Para duas ou mais variáveis, lembre-se de usar as TUPLAS, representadas pelos parênteses.

Exemplos de utilização:

- Para os exemplos abaixo, considere as definições de variáveis abaixo:

```
nome = ‘Mr Bean’
numMensagens = 10
tab = list()
tab.append( ('João',  5,  9.2) )
tab.append( ('Lucas', 7,  9.3) )
tab.append( ('Pedro', 3,  4.2) )
```

%20s indica que o conteúdo de nome será ajustado para 20 posições. Porém alinhado a direita.

- Criando uma saída em formato de tabela com coluna fixa:

```
print("%20s | %5d" % (nome,numMensagens) )
                Mr Bean |    10
print("%-20s | %5d" % (nome,numMensagens) )
Mr Bean                 |    10
```

Ao indicar -20, voltamos o alinhamento a esquerda.

- Criando uma saída em formato de tabela delimitada por tabulação (\t):

```
print("%-20s\t%5d" % (nome,numMensagens) )
Mr Bean                         10
```

Exemplos de utilização:

- Para os exemplos abaixo, considere as definições de variáveis abaixo:

```
nome = ‘Mr Bean’
numMensagens = 10
tab = list()
tab.append( ('João',  5,  9.2) )
tab.append( ('Lucas', 7,  9.3) )
tab.append( ('Pedro', 3,  4.2) )
```

- Criando uma saída em formato de tabela delimitada por tabulação (\t):

```
for reg in tab:
        print("%-20s\t%5d\t%3.2f" % reg )
```

```
João                    5    9.20
Lucas                   7    9.30
Pedro                   3    4.20
```

Formatando String

Símbolo	Conversão/saída
%c	caractere
%s	String, conversão realizada pela função str
%i	Inteiro decimal com sinal
%d	Inteiro decimal com sinal
%u	Inteiro decimal sem sinal
%o	Valor em base octal
%x	Valor em hexadecimal (letras em minúsculas)
%X	Valor em hexadecimal (letras em maiúsculas)
%e	Notação exponencial
%E	Notação exponencial (maiúscula)
%f	Número real (ponto flutuante)
%g	A versão menor dos formatos %f e %e
%G	A versão menor dos formatos %f e %e (maiúscula)

Referências

- PYTHON SOFTWARE FOUNDATION. Documentation (6.1. string – Common string operations). Disponível em: https://docs.python.org/3.4/library/string.html