Python para Aplicações em Eletrônica

Aula 03: Strings

Strings

Tipos Básicos

- Uso básico
  - **>> a='a casa de fulano'**
  - **>> a='''a casa**
    De fulano'''
- Acesso(**[]** e **[:]**):

```
1 a = 'a casa de fulano'
2
3 print a[2]
4 print a[2:6]
5 print a[7:]
```

```
c
casa
de fulano
```

- Método **len(str)**:
  - **>> print len(a)**
  - **>> 16**

Operadores Básicos

- Os operadores básicos sobre *strings* são:
  - `+` e `*`: Operadores de concatenação
  - `[]` e `[:]`: Operadores de acesso
  - **`in`** e **`not in`**: Operador de pertinência.

```
1 a = 'a casa de fulano'
2 b = ' eh bonita'
3 print 'O tamanho de a eh:', len(a)
4 print a+b
5 print b*3
6 print 'x in a:: ', 'x' in a
7 print 'c not in b:: ', 'c' not in b
```

```
O tamanho de a eh: 16
a casa de fulano eh bonita
 eh bonita eh bonita eh bonita
x in a::  False
c not in b::  True
```

Caracteres “não-printáveis”

- Existem vários caracteres que correspondem a comandos e não são exibidos.
- Alguns dos mais comuns são:
  - **`\n`**: Quebra de linha
  - **`\t`**: Tab horizontal
  - **`\v`**: Tab vertical

```
1 print 'Texto com\n quebra'
2 print '\t texto com\t tabs'
3 print '\v texto com\v tabs'
```

```
Texto com
 quebra
        texto com       tabs

 texto com
          tabs
```

Formatação de Strings (%)

- É possível inserir valores em posições especificas da *string*.
- A definição de um formato é particulamente útil para exibição e armazenamento de dados.
- Até o python 2.7 era feito utilizando o operador **%**
- É necessário informar o tipo de dado que será inserido.
- Exemplo:
  - **>> print 'a turma %s possui %d alunos!!'% ('A',32)**
  - **>> a turma A possui 32 alunos!!**

Formatação de Strings (`%`)

- Os principais tipos de dados são:
  - **`%c`**: Caracter
  - **`%s`**: String
  - **`%i`** ou **`%d`**: Inteiro com sinal
  - **`%u`**: Inteiro sem sinal
  - **`%f`**: Número real
  - **`%.[n]f`**: Número real. **`n`** é o número de casas decimais que serão usadas
  - **`%e`**: Número no formato de base 10 (**`43.214 = 4.3214x10^1`**)
  - **`%.[n]e`**: Número na notação exponecial. **`n`** é o número de casas decimais que serão usadas

Formatação de Strings (%)

```
1 print 'a turma %c possui %d alunos!!'%('A',32)
2 print 'a turma %s teve media %f !!'%('A02',6.542322)
3 print 'a turma %s teve media %.2f !!'%('A02',6.542322)
4 print 'a sala %s tem %e metros quadrdos!!'%('105',42.76584)
5 print 'a sala %s tem %.2e metros quadrdos!!'%('105',42.76584)
```

```
a turma A possui 32 alunos!!
a turma A02 teve media 6.542322 !!
a turma A02 teve media 6.54 !!
a sala 105 tem 4.276584e+01 metros quadrdos!!
a sala 105 tem 4.28e+01 metros quadrdos!!
```

Formatação de Strings (`.format()`)

- A partir do python 3. (e de versões mais novas do 2.7), foi definido o método **`.format()`**
- Este método torna desnecessário conhecer o tipo de variável que será inserida.
- Pode ser utilizada na forma de sequência, lista numerada ou “dicionário”

Formatação de Strings (`.format()`)

- Sequência:
  - `>> print 'a turma {} possui {} alunos!!'.format('A01',32)`
  - `>> a turma A01 possui 32 alunos!!`
- Lista numerada:
  - `>> print 'a turma {0} possui {1} alunos!!'.format('A01',32)`
  - `>> a turma A01 possui 32 alunos!!`
  - `>> print 'a turma {1} tem {0} alunos e media {2}!!'.format(32,'A01',6.54)`
  - `>> a turma A01 tem 32 alunos e media 6.54!!`
- Dicionário:
  - `>> print 'a turma {nome} possui {nAlunos} alunos!!'.format(nome='A01',nAlunos=32)`
  - `>> a turma A01 possui 32 alunos!!`

Formatação de Strings (`.format()`)

```
1 print 'a turma {} possui {} alunos!!'.format('A01',32)
2 print 'a turma {1} tem {0} alunos e media {2}!!'.format(32,'A01',6.54)
3 print 'valor:{1} valor:{0} valor:{1} valor:{0}!!'.format(32,'A01',6.54)
4 print 'a turma {nome} possui {nAlunos} alunos!!'.format(nome='A01',nAlunos=32)
5
6 print 'nome:{0}  idade:{1}  altura:{2:.2f}'.format('Jose Gilmar',31,1.75343)
7 print 'nome:{}  idade:{}  altura:{:.2f}'.format('Jose Gilmar',31,1.75343)
8 print 'idade:{idade}  altura:{alt:.2f}'.format(idade=31,alt=1.75343)
```

```
a turma A01 possui 32 alunos!!
a turma A01 tem 32 alunos e media 6.54!!
valor:A01 valor:32 valor:A01 valor:32!!
a turma A01 possui 32 alunos!!
nome:Jose Gilmar  idade:31  altura:1.75
nome:Jose Gilmar  idade:31  altura:1.75
idade:31  altura:1.75
```

Formatação de Strings (`.format()`)

```
1 # Programa para calcular a media de um aluno
2 print('Programa para calcular a media de um aluno\n')
3
4 nome = raw_input('Entre com o nome do aluno: ')
5 nota1 = float(raw_input("Entre com a primeira nota: "))
6 nota2 = float(raw_input("Entre com a segunda nota: "))
7 media = (nota1 + nota2)/2
8 print('{0} teve media igual a: {1:.2f}'.format(nome, media))
```

```
Programa para calcular a media de um aluno

Entre com o nome do aluno: José Gilmar
Entre com a primeira nota: 7.5342
Entre com a segunda nota: 8.2344
José Gilmar teve media igual a: 7.88
```

Métodos da classe *string*

- A classe *string* possui vários métodos notivos implementados.
- Estes métodos podem ser classificados nas seguintes áreas: formatação; remoção e alteração; verificação e busca.
- A seguir, são apresentados os principais métodos de cada área.

Métodos: Formatação

- **`center()`**, **`ljust()`** e **`rjust()`**: Métodos para alinhamento.
- **`capitalize()`**, **`title()`**, **`lower()`**, **`upper()`** e **`swapcase()`**: Alteração de padrão das letras.
- **`zfill()`**: Preenchimento com zeros

```
 1 texto = 'forMaTAcao de sTRIngs'
 2 print 'center:', texto.center(30)
 3 print 'center:',texto.center(30,'#')
 4 print 'ljust :',texto.ljust(30,'!')
 5 print 'rjust :',texto.rjust(30,'-')
 6 print 'capit :',texto.capitalize()
 7 print 'title :',texto.title()
 8 print 'lower :',texto.lower()
 9 print 'upper :',texto.upper()
10 print 'swapc :',texto.swapcase()
11 print 'zfill :',texto.zfill(30)
```

```
center:      forMaTAcao de sTRIngs
center: ####forMaTAcao de sTRIngs#####
ljust : forMaTAcao de sTRIngs!!!!!!!!!
rjust : ---------forMaTAcao de sTRIngs
capit : Formatacao de strings
title : Formatacao De Strings
lower : formatacao de strings
upper : FORMATACAO DE STRINGS
swapc : FORmAtaCAO DE StriNGS
zfill : 000000000forMaTAcao de sTRIngs
```

Métodos: Remoção e Alteração

- **`strip()`**, **`lstrip()`** e **`rstrip()`**: Remoção de caracteres no início e no fim da string.
- **`replace()`**, **`maketrans()`** e **`translate()`**: Substituição e remoção de letras.

Métodos: Remoção e Alteração

```
 1 textoA = '    eliminacao    e    alteracao     '
 2 textoB = '9999eliminacao9999e9999alteracao99999'
 3 print 'strip :', textoA.strip()
 4 print 'lstrip:', textoA.lstrip()
 5 print 'rstrip:', textoA.rstrip()
 6 print 'strip: ', textoB.strip('9')
 7 print 'lstrip:', textoB.lstrip('9')
 8 print 'rstrip:', textoB.rstrip('9')
 9
10 print 'replace: ', textoB.replace('9',' ')
11 print 'replace: ', textoB.replace('9',' ',2)
12
13 from string import maketrans
14 intab = "aeiosh"
15 outtab = "@31057"
16 trantab = maketrans(intab, outtab)
17
18 str = "Que texto mais lindo!!!";
19 print str
20 print str.translate(trantab)
21 print str.translate(trantab, 'xl')
```

```
strip : eliminacao    e    alteracao
lstrip: eliminacao    e    alteracao
rstrip:     eliminacao    e    alteracao
strip : eliminacao9999e9999alteracao
lstrip: eliminacao9999e9999alteracao99999
rstrip: 9999eliminacao9999e9999alteracao
replace:     eliminacao    e    alteracao
replace:  99eliminacao9999e9999alteracao99999
Que texto mais lindo!!!
Qu3 t3xt0 m@15 l1nd0!!!
Qu3 t3t0 m@15 1nd0!!!
```

Métodos: Verificação

- **`isalnun()`** e **`isalpha()`**: Verifica se a *string* possui apenas caracteres alfanuméricos ou letras.
- **`isnumeric()`**, **`isdigit()`** ou **`isdecimal()`**: Verifica se a *string* possui apenas números.
- **`isspace()`**: Verifica se a *string* possui apenas caracteres espaço.
- **`islower()`**, **`isupper()`** e **`istitle()`**: Verifica se a *string* está formatada nos respectivos tipos.

Métodos: Verificação

```
1 a='a02';b='Acasa'; c=u'1235'; d=u'12.35'
2 print 'a:{} b:{} c:{} d:{}'.format(a,b,c,d)
3 print 'a:: AlNum:{} Alpha:{}'.format(a.isalnum(), a.isalpha())
4 print 'b:: AlNum:{} Alpha:{}'.format(b.isalnum(), b.isalpha())
5 print 'c:: Di:{} Nu:{} De:{}'.format(c.isdigit(),c.isnumeric(), c.isdecimal())
6 print 'd:: Di:{} Nu:{} De:{}'.format(d.isdigit(),d.isnumeric(), d.isdecimal())
7 e='1235'
8 print 'e:{} :: Di:{}'.format(e,e.isdigit())
```

```
a:a02 b:Acasa c:1235 d:12.35
a:: AlNum:True Alpha:False
b:: AlNum:True Alpha:True
c:: Di:True Nu:True De:True
d:: Di:False Nu:False De:False
e:1235 :: Di:True
```

Métodos: Busca

- **`count()`**: Conta quantas vezes uma *substring* aparece na string.
- **`endswith()`** e **`startswith()`**: Verifica se a *string* termina ou inicia com uma determinada *substring*.
- **`find()`**, **`rfind()`**, **`index()`** e **`rindex()`**: Encontra o índice da primeira aparição da *substring* (pela esquerda ou pela direita).

Métodos: Busca

```
1 texto='Casa bola casa Casa caso Bola dolar'
2 print 'casa:{} bola:{}'.format(texto.count('casa'),texto.count('bola'))
3 print 'casa:{} bola:{}'.format(texto.lower().count('casa'),texto.lower().count('bola'))
4 print 'Start:Cas:{} Ends:lar:{}'.format(texto.startswith('Cas'),texto.endswith('lar'))
5 print 'Start:sa:{}'.format(texto.startswith('sa',2,10))
6 print 'find:: as:{} ol:{}'.format(texto.find('as'),texto.find('ol'))
7 print 'find:: as:{} ol:{}'.format(texto.rfind('as'),texto.rfind('ol'))
8 print 'find:: as:{} ol:{}'.format(texto.rfind('as',2,20),texto.rfind('ol',2,20))
```

```
casa:1 bola:1
casa:3 bola:2
Start:Cas:True Ends:lar:True
Start:sa:True
find:: as:1 ol:6
find:: as:21 ol:31
find:: as:16 ol:6
```

Métodos: Outros

- **`min()`** e **`max()`**: Identifica o elemento de “maior” e “menor” valor na *string*.
- **`join(seq)`**: Cria *string* a partir de sequência.
- **`split()`** e **`splitlines()`**: Cria sequência a partir de *string*.

Métodos: Busca

```
 1 texto='123asczk#'
 2 print 'min:{} max:{}'.format(min(texto),max(texto))
 3
 4 seq=['1','2','3','4','5']
 5 s='-'
 6 print 'join: ', s.join(seq)
 7 str1='0 1 2 3 4 as 4 v asa'
 8 str2='0 1 2-3 4 as-4-v asa'
 9 print 'lista1:{}'.format(str1.split())
10 print 'lista1:{}  :max 3'.format(str1.split(' ',3))
11 print 'lista2:{}'.format(str2.split())
12 print 'lista2:{}  :com -'.format(str2.split('-'))
```

```
min:# max:z
join:  1-2-3-4-5
lista1:['0', '1', '2', '3', '4', 'as', '4', 'v', 'asa']
lista1:['0', '1', '2', '3 4 as 4 v asa']  :max 3
lista2:['0', '1', '2-3', '4', 'as-4-v', 'asa']
lista2:['0 1 2', '3 4 as', '4', 'v asa']  :com -
```

Tópicos especiais

Lendo Arquivo

- Salvando dados em um arquivo de texto

```
 1 a = ['lista A', 1,2,3,4,5,6,7,8]
 2 b = ['lista B', 9,8,7,6,5,4,3,2];s = ' '
 3 ## Escrevendo em arquivo
 4 #w - escrita (do zero)
 5 #r - leitura
 6 #a - escrita (adicionando ao arquivo)
 7 file = open('dados.txt','w')
 8 file.write(s.join(str(it_a) for it_a in a))
 9 file.write('\n')
10 file.write(s.join(str(it_b) for it_b in b))
11 file.close()
```

Lendo Arquivo

- Lendo os dados

```
12 file = open('dados.txt','r')
13 linhas=file.readlines()
14 print linhas
15 lista1 = linhas[0].split()
16 lista2 = linhas[1].split()
17 print 'lista1::', lista1
18 print 'lista2::', lista2
19 dadosA=[];dadosB=[]
20 for i in range(2,len(lista1)):
21     dadosA.append(float(lista1[i]))
22     dadosB.append(float(lista2[i]))
23 print dadosA
24 print dadosB
```

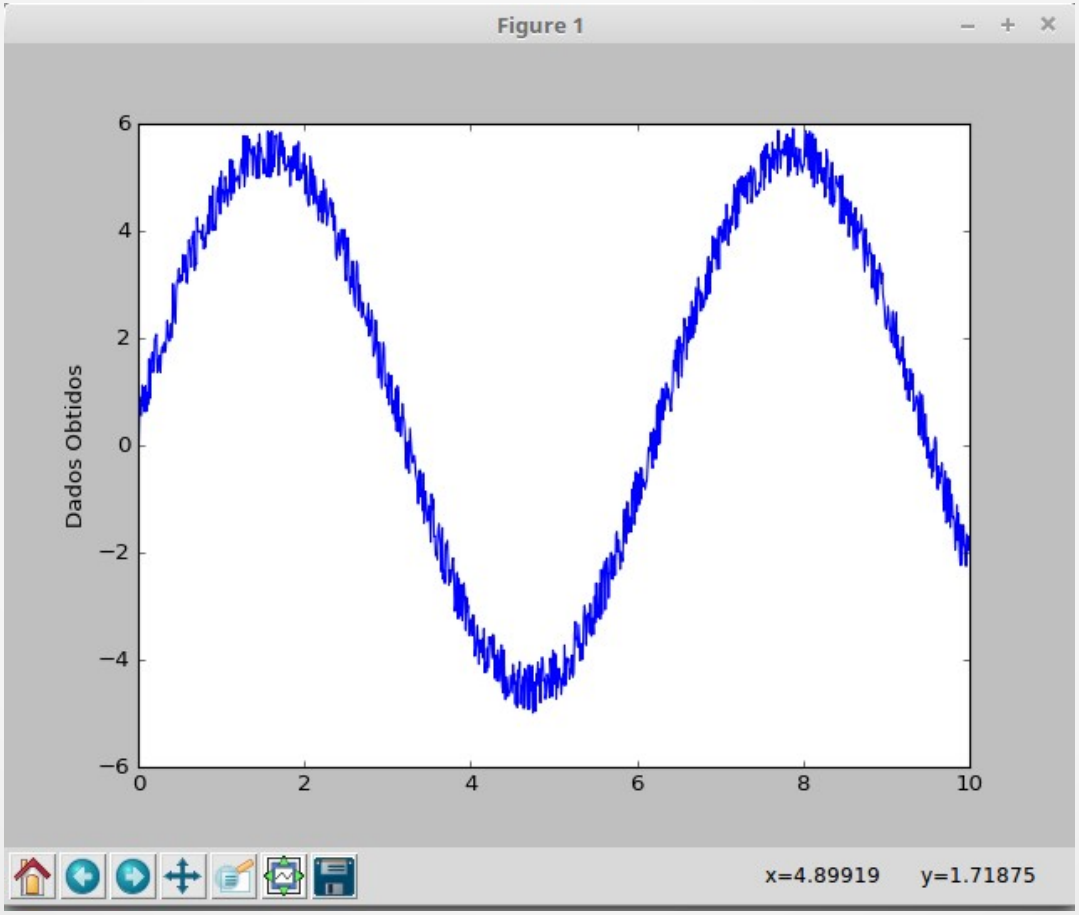
```
['lista A 1 2 3 4 5 6 7 8\n', 'lista B 9 8 7 6 5 4 3 2']
lista1:: ['lista', 'A', '1', '2', '3', '4', '5', '6', '7', '8']
lista2:: ['lista', 'B', '9', '8', '7', '6', '5', '4', '3', '2']
[1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0]
[9.0, 8.0, 7.0, 6.0, 5.0, 4.0, 3.0, 2.0]
```

Plotando Gráficos

- É possível plotar gráficos utilizando a função **`plot`** da biblioteca **`matplotlib`** do python.

```
 1 import matplotlib.pyplot as plt
 2 import math
 3 import random
 4 t = range(0,1000)
 5 y=[];x=[]
 6 for tt in t:
 7     x.append(tt*0.01)
 8     y.append(5*math.sin(tt*0.01)+random.random())
 9 plt.plot(x,y)
10 plt.ylabel('Dados Obtidos')
11 plt.show()
```

Plotando Gráficos



Exercícios

Exercícios

1) Criar um código que simule diferentes dados (valores numéricos) sendo obtidos, por exemplo, a partir da leitura de sensores.

- *Salve os dados em arquivo*

Exercícios

2) Crie um código para ler dados gravados em arquivo .txt, fazer a analise dos dados e plotá-los em gráficos.

- *Utilize* ***`timestamps`*** *como os instantes de plot.*