Python para Aplicações em Eletrônica

Aula 02: Números e Strings

Números

Tipos Básicos

- Inteiros (**`int`** e **`long`**):
  - **`>> a=100`**
- Números reais (**`float`**):
  - **`>> b=12.75`**
- Números complexos (**`complex`**):
  - **`>> c=11.57+3.41j`**

Conversão em Números

- É possível converter *strings* ou outros tipos de números em formatos específicos.
- Função **`int(x)`**:
  - Converte um valor **`x`** em inteiro.
  - **`x`** pode ser um número real ou uma string de um número inteiro.
  - Não é possível converter um número complexo em **`int`** (nem **`long`** ou **`float`**)

```
1 x = 11.34
2 y = '11'
3 z = '11.34'
4
5 print int(x)
6 print int(y)
7 print int(z)
```

```
11
11
Traceback (most recent call last):
  File "ex01.py", line 7, in <module>
    print int(z)
ValueError: invalid literal for int() with base 10: '11.34'
```

Conversão em Números

- A função **`long(x)`** é similar à **`int(x)`**.
- Função **`float(x)`**:
  - Converte um valor **`x`** em um valor real.
  - **`x`** pode ser um número real ou uma string de um número real.
  - Não é possível converter um número complexo em **`float`**.
- Função complex**`(x)`**:
  - Converte um valor **`x`** em um valor complexo.
  - **`x`** pode ser um número complexo (inteiro ou real) ou uma string de um número.

Funções Matemáticas Básicas

Funções Básicas

- **`abs(x)`**: Cálcula o módulo de um número **`x`**
- **`cmp(x,y)`**: Retorna os seguintes valores
  - -1, se x<y;
  - 0, se x==y
  - 1, e x>y
- **`max(x,y, ...)`** ou **`max([x,y, ...])`**: Retorna o maior valor.
- **`min(x,y, ...)`** ou **`min([x,y, ...])`**: Retorna o menor valor.
- **`round(x)`** ou **`round(x,n)`**: Arredonda o valor de **`x`** em **`n`** casas decimais.

Funções Básicas

```
 1 #!/usr/bin/python
 2 a=-32
 3 b=12
 4 c =3+4j
 5 print '|a|:',abs(a),' |b|:', abs(b), ' ||c||:', abs(c)
 6
 7 print 'max(1,2,75,63,12,-5):',max(1,2,75,63,12,-5)
 8 l = [1,2,75,63,12,-5]
 9 print 'min de ', l, ':', min(l)
10
11 d = 13.6544346
12 print 'round(d):',round(d), ' round(d,2):', round(d,2)
```

```
|a|: 32  |b|: 12  ||c||: 5.0
max(1,2,75,63,12,-5): 75
min de  [1, 2, 75, 63, 12, -5] : -5
round(d): 14.0  round(d,2): 13.65
```

Biblioteca **`math`**

- A biblioteca **`math`** possui várias funções matemáticas básicas.
- Adicione a instrução **`import math`** para utilizar a biblioteca.
- As principais funções são:
  - **`ceil(x)`**: Arredondamento “para cima”
  - **`floor(x)`**: Arredondamento “para baixo”
  - **`exp(x)`**: Exponencial de x
  - **`log(x)`**: Logaritmo natural de x
  - **`log10(x)`**: Logaritmo de x na base 10
  - **`pow(x,y)`**: Similar a **`x**y`**
  - **`sqrt(x)`**: Raiz quadrada

Biblioteca **math**

```
 1 #!/usr/bin/python
 2 import math
 3
 4 a = 12.35
 5
 6 print 'math.ceil(a):',math.ceil(a), ' math.floor(a):', math.floor(a)
 7 print 'exp(2.2):', math.exp(2.2)
 8 print 'log(100):', math.log(100), 'log10(100):', math.log10(100)
 9 print 'pow(2,4):', math.pow(2,4), ' 2**4:', (2**4)
10 print 'sqrt(81):', math.sqrt(81)
11 print 'exp(2+3j):',math.exp(2+3j)
```

```
math.ceil(a): 13.0  math.floor(a): 12.0
exp(2.2): 9.02501349943
log(100): 4.60517018599 log10(100): 2.0
pow(2,4): 16.0  2**4: 16
sqrt(81): 9.0
exp(2+3j):
Traceback (most recent call last):
  File "ex03.py", line 11, in <module>
    print 'exp(2+3j):',math.exp(2+3j)
TypeError: can't convert complex to float
```

Biblioteca **`math`**

- A biblioteca math possui as principais funções trigonométricas.
  - **`cos(x)`**, **`sin(x)`** e **`tan(x)`**: **`x`** em radianos
  - **`acos(x)`**, **`asin(x)`** e **`atan(x)`**: cosseno, seno e tangente inversos, resultado em radianos
  - **`atan2(x,y)`**: Arco-tangente com informação de quadrante
  - **`degrees(x)`**: Converte x em graus
  - **`radians(x)`**: Converte x para radianos
  - **`pi`** e **`e`**: Constantes matemáticas
  - **`hypot(x,y)`**: Retorna **`sqrt(x*x+y*y)`**

Biblioteca **math**

```
 1 #!/usr/bin/python
 2 import math
 3
 4 ang_d = 45
 5 ang_r = math.radians(ang_d)
 6
 7 print 'ang_d:', ang_d, ' ang_r:', ang_r
 8 print 'cos(x):', math.cos(ang_r),' sin(x):', math.sin(ang_r)
 9 print 'tan(x):', math.tan(ang_r)
10
11 print 'acos(0.5):', math.acos(0.5),' asin(x):', math.asin(0.5)
12 print 'math.atan(-0.5):',math.atan(-0.5)
13 print 'math.atan2(-1,2):',math.atan2(-1,2)
14 print 'math.atan2(1,-2):',math.atan2(1,-2)
15 print 'pi:', math.pi, ' e:', math.e
```

```
ang_d: 45  ang_r: 0.785398163397
cos(x): 0.707106781187  sin(x): 0.707106781187
tan(x): 1.0
acos(0.5): 1.0471975512  asin(x): 0.523598775598
math.atan(-0.5): -0.463647609001
math.atan2(-1,2): -0.463647609001
math.atan2(1,-2): 2.67794504459
pi: 3.14159265359  e: 2.71828182846
```

Biblioteca **`random`**

- A biblioteca **`random`** possui vários funções para obtenção de valores aleatórios ou “bagunçar” listas.
- Deve ser adicionada a instrução **`import random`** no início do código.
- As principais funções são:
  - **`random()`**: gera um número real aleatório entre 0 e 1.
  - **`uniform(x,y)`**: gerar um número **`n`**, tal que **`x<= n <y`**
  - **`randrange([start,]stop,[step])`**: Seleciona um elemento (**`int`**) da lista formada pelos números de **`start`** até **`stop`**. Os números da lista são gerados a partir de **`start`** com passo de tamanho **`step`**.
  - **`choice(seq)`**: Seleciona um elemento de **`seq`**, que pode ser um lista, tupla ou string.
  - **`shuffle(lst)`**: Altera a posição dos elementos de uma lista aleatoriamente.

Biblioteca **random**

```
 1 #!/usr/bin/python
 2 import random
 3
 4 print 'random(): ', random.random()
 5 print 'uniform(3.56,72.1): ', random.uniform(3.56,7.1)
 6 print 'randrange(10): ', random.randrange(10)
 7 print '1->randrange(13,79,10): ', random.randrange(13,79,10)
 8 print '2->randrange(13,79,10): ', random.randrange(13,79,10)
 9
10 seq = [1,2,3,4,5,6,7,8,9,10]
11 print 'seq:', seq
12 print 'choice(seq): ', random.choice(seq)
13 print 'shuffle(seq): ', random.shuffle(seq)
14 print 'seq:', seq
```

```
random():  0.138896015543
uniform(3.56,72.1):  4.74745421084
randrange(10):  6
1->randrange(13,79,10):  73
2->randrange(13,79,10):  53
seq: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
choice(seq):  5
shuffle(seq):  None
seq: [6, 10, 9, 4, 2, 5, 8, 3, 1, 7]
```

Biblioteca **`random`**

- Não existe aleatoriedade real. Os comportamento aleatório obtido pela biblioteca **`random`** se dá pelo uso de um algoritmo determinístico e um número “semente” (*seed*).
- Comumente é utilizado o último valor gerado como *seed*. Para o primeiro valor gerado, o *seed* pode assumir um valor padrão.
- É possível definir o valor de seed antes começar a utilizar as funções da biblioteca.
  - **seed(x)**: Define o valor inteiro inicial utilizado para geração de números aleatórios.
- Uma boa prática é utilizar o instante de tempo atual como *seed*.

```
1 #!/usr/bin/python
2 import random
3 import time # Biblioteca necessaria para usar informacoes de tempo
4
5 n=time.time() # tempo em s (desde 01/01/1970)
6 print(n)
7 random.seed(n)
```

Tópicos especiais

Try Except

- Certos valores gerados, atribuídos ou lidos durante a execução do programa podem gerar erros.
- Esses erros costumam fechar o programa.
- É possível fazer o tratamento de tais erros usando **`try … except.`**
- Os principais usos para este tipo de tratamento são nas seguintes situações:
  - Entrada de dados de usuários ou em arquivos: podem vir em formatos diferentes do esperado
  - Criação ou escrita em arquivos;
  - Possibilidade de divisão por zero;
  - Etc.

Try Except

- O uso se é feito da seguinte forma:

```
try:
    # codigo a ser executado
except:
    # codigo executado se ocorrer qualquer excecao
else:
    # codigo executado se nao ocorrer excecao
```

- É possível fazer o tratamento de erros específicos. Os principais tipos são:
  - **`IndexError`**: Erro ao tentar acessar posição não encontrada em um sequência.
  - **`ValueError`**: O tipo usado como argumento está correto, mas a operação é inválida para o valor especificado
  - **`IOError`**: Erro em operações de entrada e saída.
  - **`KeyError`**: Similar ao anterior para dicionário.
  - **`TypeError`**: Erro ao tentar executar funções ou operações com tipos inválidos.
  - **`ArithmeticError`**: Qualquer erro de cálculo numérico

Try Except

```
1 #!/usr/bin/python
2
3 n=raw_input('Digite um valor:')
4 print n
5
6 try:
7     c = int(n)
8     print 'c:', c
9 except:
10    print 'Deu ruim!!!'
11 else:
12    print 'Deu bom!!'
```

```
Digite um valor:312
312
c: 312
Deu bom!!
```

```
Digite um valor:32.1
32.1
Deu ruim!!!
```

```
1 #!/usr/bin/python
2
3 n=raw_input('Digite um valor:')
4 print n
5 a=0
6 try:
7     c = int(n)
8     print 'c:', c
9     print c/a
10 except ValueError:
11    print 'Nao foi possivel converter o valor'
12 except:
13    print 'Alguma coisa deu ruim!!!'
14 else:
15    print 'Deu bom!!'
```

```
Digite um valor:32
32
c: 32
Alguma coisa deu ruim!!!
gilmar@gilmarPC ~/Dropbox/Ensino/Minicursos/Python/Aula02/Codigos $ python ex08.py
Digite um valor:32.1
32.1
Nao foi possivel converter o valor
```

Salvando em arquivo

- O seguinte código pode ser utilizado para salvar textos em arquivos

```
1 #!/usr/bin/python
2
3 file = open('text.txt', 'w')
4 file.write('whatever')
5 file.close()
```

Exercícios

Exercícios

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

- *No início da execução do programa, a taxa de amostragem.*
- *Defina uma condição de parada (número de amostras ou comando, por exemplo.)*
- *Utilize arredondamento em* **`n`** *(a ser definido no início do programa) casas decimais.*
- *Os dados devem ter algum componente aleatório.*
- *Utilize também funções como senoides com incertezas como fontes de sinal.*
- *Teste diferentes formas de aleatoriedade.*

Exercícios

2) Adicione ao código anterior a funcionalidade de gravar os dados coletados em um arquivo **`.txt`**.

- *Ao iniciar o programa, dever ser inserido o arquivo onde os dados serão gravados.*
- *Torne seu programe “o mais a prova de falhas” possível.*
- *Adicione **`timestamps`** aos dados coletados.*

Exercícios

3) Adicione ao código anterior a funcionalidade de análise de dados (valor máximo e mínimo, média, mediana, desvio padrão, etc.).

- *Após a condição de parada do processo de amostragem ser alcançada, analise os dados gerados e obtenha as métricas definidas acima.*
- *Grave a análise também em arquivo* **`.txt`**.