

# Strings

- As STRINGS são formadas por caracteres da tabela ASCII (ou UNICODE)
- Para transformar um valor de qualquer outro tipo em STRING basta usar o comando

*str(VALOR)*

Exemplo

```
>>> str(True)
```

```
'True'
```

```
>>> str(3)
```

```
'3'
```

# Strings

- Todas as classes/tipos que possuem ideia de ordenação (como a classe STRING) podem ser acessadas por partes
- Cada elemento da STRING possui um número em sequência
- Ex: nome = 'PYTHON'

P	Y	T	H	O	N
0	1	2	3	4	5
-6	-5	-4	-3	-2	-1

# Strings

- Para acessar a letra 'P'  
***nome[0] ou nome[-6]***
- Para acessar a letra 'N'  
***nome[5] ou nome[-1]***
- Para acessar 'YTH'  
***nome[1:4]***
- Para acessar os 5 primeiros caracteres ('PYTHO')  
***nome[:5]***

# Strings

- Para pegar os caracteres alternados (PTO)  
`nome[0:5:2]`  
`nome[:5:2]`  
`nome[-6:-1:2]`  
`nome[-6::2]`
- O comando `len (nome)` retorna o tamanho da STRING  

```
>>> len(nome)  
6
```

# *ord()* e *chr()*

- *ord(x)*

- Recebe uma string formada por um ÚNICO caractere e retorna seu número na tabela ASCII

```
>>> ord('a')
```

```
97
```

```
>>> ord('1')
```

```
49
```

- *chr(x)*

- Recebe um número inteiro e retorna o caractere representado na tabela ASCII

```
>>> chr(49)
```

```
'1'
```

```
>>> chr(97)
```

```
'a'
```

# Programa

- Fazer um programa para ler uma cadeia e imprimir a soma dos valores ASCII dos caracteres

```
cadeia = raw_input('entre com uma cadeia: ')  
soma = 0  
for i in range(0,len(cadeia)):  
    soma+= ord(cadeia[i])  
print 'SOMA: ', soma
```

# Métodos da Classe string

- Métodos são parecidos com funções
- ***count***
  - Conta quantas vezes uma string *b* aparece em uma string *a*
  - ***a.count(b,[inicio,fim])***

Exemplo:

```
>>> a = 'abacaxi'
```

```
>>> a.count('a')
```

```
3
```

```
>>> a.count('ac')
```

```
1
```

```
>>> a.count('ix')
```

```
0
```

# Métodos da Classe string

- *lower*

- Retorna uma cópia em letras minúsculas

- *a.lower()*

```
>>> a = 'ABACAXI'
```

```
>>> a.lower()
```

```
'abacaxi'
```

- *upper*

- Retorna uma cópia em letras maiúsculas

- *a.upper()*

```
>>> a = 'brasil'
```

```
>>> a.upper()
```

```
'BRASIL'
```



# Métodos da Classe string

- *isalnum*

- Retorna **True** se TODOS os caracteres da STRING forem alfanúmericos
- *a.isalnum()*

```
>>> a = 'ABACAXI123'
```

```
>>> a.isalnum()
```

```
True
```

# Programa

- Fazer um programa para ler uma STRING e imprimir o total de letras, o total de números e a quantidade de outros caracteres.

```
cadeia = raw_input('entre com uma cadeia: ')
contA = 0
contN = 0
contO = 0
for i in range(0,len(cadeia)):
    if cadeia[i].isalpha():
        contA+=1
    elif cadeia[i].isdigit():
        contN+=1
    else:
        contO += 1
print 'LETRAS: ', contA, ' NUMEROS: ', contN, 'OUTROS: ', contO
```

# Métodos da Classe string

- *replace*
  - Retorna uma nova string substituindo na string a todas as ocorrências de uma string b por uma nova string c
  - Pode ser usada também um parâmetro opcional qtd que limita a quantidade de substituições a serem feitas
  - `a.replace(b,c[,qtd])`

Exemplo

```
>>> a = 'brasil'
>>> a.replace('s','z')
'brasil'
```

# Métodos da Classe string

- `split`
  - Separa a string *a* toda vez que for encontrada uma string *b*
  - Cada fracionamento da string será transformado em um item de uma lista
  - Pode ser usada também um parâmetro opcional *qtd* que limita a quantidade de fracionamentos a serem feitos
  - *a.split(b[,qtd])*

Exemplo

```
>>> a = 'brasil'
>>> a.split('a')
['br','sil']
```

# Programa

- Fazer um programa para ler uma sequência de DNA e imprimir a sua sequência complementar.
  - Ex: DNA = 'ATTGCA'  
COMP='TAACGT'

```
dna = raw_input('entre com DNA: ')\ndna = dna.upper()\nprint 'DNA: ', dna
```

```
dna = dna.replace('A','@')\ndna = dna.replace('T','A')\ndna = dna.replace('@','T')
```

```
dna = dna.replace('C','@')\ndna = dna.replace('G','C')\ndna = dna.replace('@','G')\nprint 'DNA COMPLEMENTAR: ', dna
```

# Formatação de *Strings*

- Pular linhas => `'\n'`
  - Ex: `a = input('\n\n digite valor: \n')`
- Tabulação horizontal => `'\t'`
  - Ex: `print '\t oi'`
- Para colocar um `'` ou um `"` dentro da string tem que usar a `\`
  - Ex: `print 'testando \' e \''`

# Formatação de *Strings*

- Métodos:
- `rjust` => define o espaçamento que será dado a direita
  - Ex: `a = 'um'`
  - `print a.rjust(3), a.rjust(3)`
- `ljust` => define o espaçamento que será dado a esquerda
- `center` => centraliza a string de acordo com o tamanho passado como parâmetro
  - `a.center(40)`

# Formatação de *Strings*

- Fazer um programa imprimir os 20 primeiros múltiplos de 7 com 4 espaços. Além disso imprimir o título “MULT 7” centralizado.

```
print "MULT 7".center(80)  
for i in range(0,20):  
    print str(i*7).rjust(4)
```



# Arquivos

- Até agora, quando terminamos de executar um programa  
TODOS OS DADOS SÃO PERDIDOS
- Isso acontece pois as variáveis, vetores, matrizes e listas são armazenadas na memória principal
  - A memória principal (RAM) é volátil
- Através do uso dos arquivos podemos guardar os dados em memória secundária
  - A memória secundária é persistente
  - Desse modo, quando terminamos de executar o programa os dados são mantidos nos arquivos e podemos recuperá-los em uma nova execução

# Arquivos

- Para trabalhar com arquivo usamos os seguintes métodos
  - ***open(diretorio\_nome\_arquivo,modo)***
    - O modo pode ser:
      - ‘r’ => *para abrir o arquivo somente para leitura*
      - ‘w’ => *para abrir o arquivo somente para escrita. Caso tenha algum dado dentro do arquivo ele será perdido*
      - ‘a’ => *o arquivo será aberto para adicionar dados ao final do arquivo*
      - *Podemos também acrescentar o modo ‘b’ aos três modos anteriores para o arquivo ser tratado como binário*
    - Ex: ***f = open('c:\exemplo.txt','w')***

# Arquivos

- ***close()***
  - O método ***close*** é utilizado para fechar um arquivo e liberar recursos.
  - Vale ressaltar que qualquer tentativa de acessar o arquivo novamente resultará em falha.
  - Esse método normalmente não recebe argumentos e é chamado da seguinte forma:
    - ***nome\_do\_arquivo.close()***
    - ***f.close()***
- ***write (string)***
  - O método ***write*** é usado para escrever uma string em um arquivo
    - ***nome\_do\_arquivo.write(string)***
    - ***f.write('nome:' + frase)***

# Programa

- Fazer um programa para ler 5 nomes e 5 notas e guardar em um arquivo

```
f = open('teste.txt','w')
for i in range(0,5):
    nome = raw_input('nome:')
    nota = input('nota:')
    f.write(nome + ' ' + str(nota)+"\n")
f.close()
```

*alex 3*  
*bianca 7.5*  
*carlos 2.0*  
*dado 7*  
*erica 1.5*

# Arquivos

- ***read()***
  - O método ***read*** é utilizado para ler uma determinada quantidade de dados.
  - Esse método recebe um argumento opcional chamado ***size***, ou tamanho, que caso não seja especificado será considerado como o tamanho total do arquivo.
    - ***nome\_do\_arquivo.read()***
    - ***f.read()***
- ***readline()***
  - O método ***readline*** lê uma única linha do arquivo, avaliado através do caractere de retorno de linha (***\n***)
  - Em sua segunda chamada lê a segunda linha do arquivo, em sua terceira, a terceira linha e assim sucessivamente até encontrar o ***EOF***.

# Programa

- Fazer um programa para ler um arquivo e imprimir o conteúdo deste arquivo na tela

```
f = open('teste.txt','r')  
cadeia = f.read()  
print cadeia  
f.close()
```

# Programa

- Fazer um programa para ler o arquivo criado e imprimir o nome do aluno que tirou a maior nota e a média da turma

```

soma = 0
cont = 0
maiorNota = -1
f = open('teste.txt','r')
for linha in f:
    [nome,nota] = linha.split()
    if float(nota) > maiorNota:
        maiorNota = float(nota)
        maiorNome = nome
    soma += float(nota)
    cont += 1
    print nota
print 'media:',soma/cont
print 'maior nota:', maiorNome
f.close()

```

```

soma = 0
cont = 0
maiorNota = -1
f = open('teste.txt','r')
fim = False
while not fim:
    linha = f.readline()
    if len(linha) == 0:
        fim = True
    else:
        [nome,nota] = linha.split()
        if float(nota) > maiorNota:
            maiorNota = float(nota)
            maiorNome = nome
        soma += float(nota)
        cont += 1
        print nota
print 'media:',soma/cont
print 'maior nota:', maiorNome
f.close()

```



# Programa

- Fazer um programa para ler o arquivo criado e criar um outro arquivo contendo apenas os nomes dos alunos que tiveram nota acima da média da turma

```
soma = 0
cont = 0
maiorNota = -1
f = open('teste.txt','r')
for linha in f:
    [nome,nota] = linha.split()
    soma += float(nota)
    cont += 1
f2 = open('teste2.txt','w')
f = open('teste.txt','r')
media = soma/cont
for linha in f:
    [nome,nota] = linha.split()
    if float(nota) > media:
        f2.write(nome+'\n')
        print 'nome:', nome
f.close()
f2.close()
```