Python – Biblioteca *math e Strings*

Importação de módulos

- O PYTHON possui muitas bibliotecas ou módulos que NÃO fazem parte do núcleo do PYTHON
- Nesse caso, é necessário IMPORTÁ-LAS
- SINTAXE DO COMANDO DE IMPORTAÇÃO

  import NOME_MODULO

  Exemplo

  >>> import math
- Para utilizar alguma função da biblioteca é necessário colocar primeiro o nome da biblioteca

  Ex: math.sin(0)

Biblioteca *math*

- O módulo math contém várias funções matemáticas úteis para resolver problemas matemáticos mais rapidamente

- Funções Trigonométricas

Em todas a funções x é um arco medido em radianos

- cos(x) => calcula o cosseno de x
- sin(x) => calcula seno de x
- tan(x) => calcula a tangente de x
- acos(x) => calcula o arco cosseno de x
- asin(x) => calcula o arco seno de x
- atan(x) => calcula o arco tangente de x

Biblioteca *math*

- Funções importantes
  - ***exp(x)*** => calcula e ** x
  - ***log(x,y)*** => calcula o log de x na base y
  - ***factorial(x)*** => calcula o fatorial de x
  - ***sqrt(x)*** => calcula a raiz quadrada de x
  - ***abs(x)*** => retorna o valor absoluto de um inteiro x
  - ***fabs(x)*** => retorna o valor absoluto de um float x
  - ***ceil(x)*** => retorna o menor número inteiro maior do que o float x
  - ***floor(x)*** => retorna o maior número inteiro menor do que o float x

Exercício

- Fazer um programa em PYTHON para ler os valores x e y e imprima a expressão abaixo:

$$\sum_{i=1}^{50} \left(\frac{x+\sqrt{y+2}}{3} \right) + i^2 - y^5$$

Programa

Programa

```
import math
e = 0
x = input('entre com x: ')
y = input('entre com y: ')
for i in range(1,51):
    e = e + (x + math.sqrt(y+2))/3 + i**2 - y**5
print 'e:', e
```

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 acesar 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

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')

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.

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')
'brazil'

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’

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:  ')
dna = dna.upper()
print 'DNA: ', dna

dna = dna.replace('A','@')
dna = dna.replace('T','A')
dna = dna.replace('@','T')

dna = dna.replace('C','@')
dna = dna.replace('G','C')
dna = dna.replace('@','G’)
print '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 “ dentr 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.

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)
```