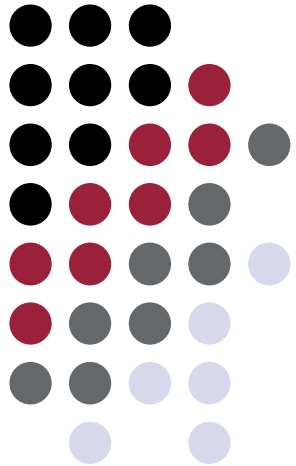
A10 Introdução ao Python

PEP300 – Técnicas Metaheurísticas para Otimização Combinatória

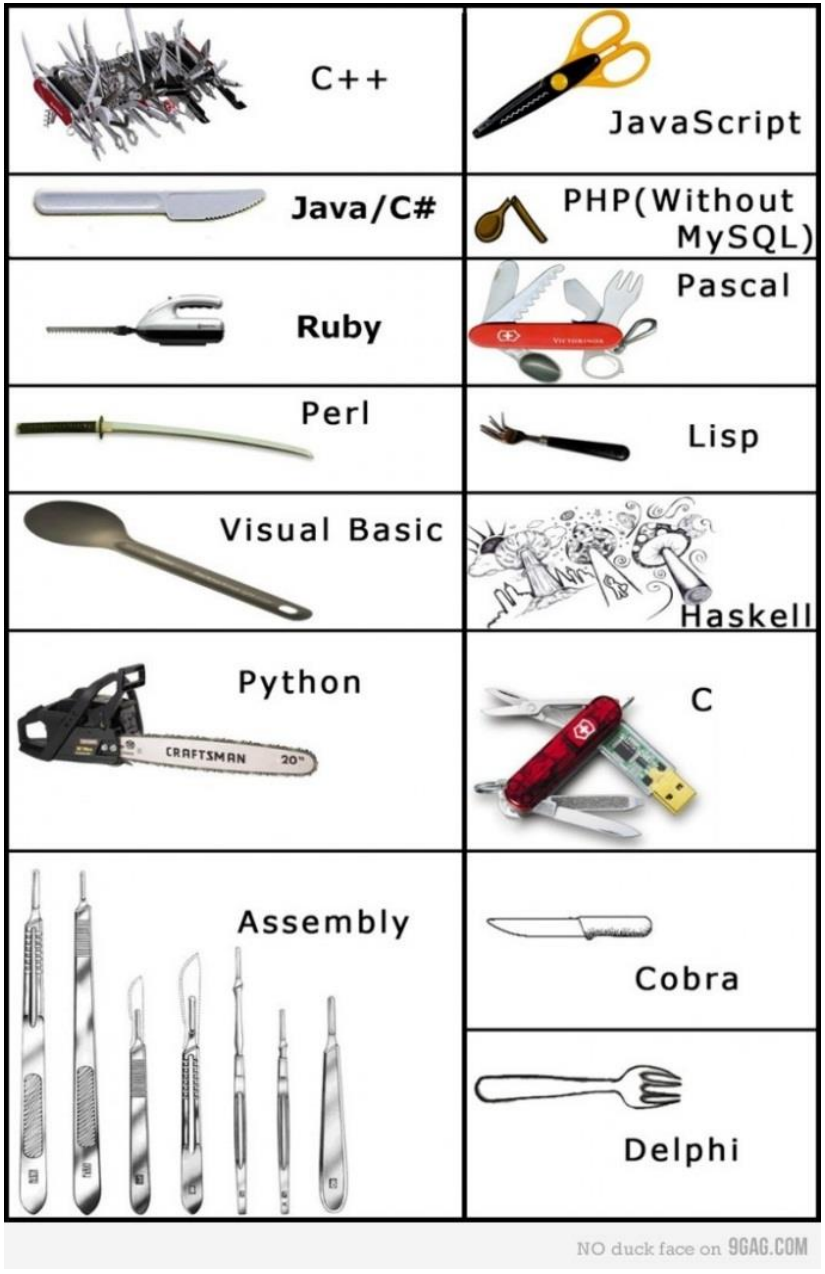
Prof. Dr. George H. G. Fonseca
Universidade Federal de Ouro Preto



Introdução

- Abordado
  - Visão geral (1, 2, 3)
  - Tipos de dados (4)
  - Operadores Básicos (5)
  - Controle de Fluxo (6, 7)
  - Funções e Módulos (14, 15)
  - Entrada / Saída (16)
  - Exceptions (17)
- Não abordado
  - Funções de tipos de dados (8 - 13)
  - Classes / Objects (18)
  - Expressões regulares (19)
  - Acesso a banco de dados (21)
  - Multithread (24)
  - Programação de GUIs (26)
  - Programação Web (20, 22, 23, 25)
  - Utilidades e extensões (26, 27)

Introdução


C++
JavaScript
Java/C#
PHP(Without MySQL)
Ruby
Pascal
Perl
Lisp
Visual Basic
Haskell
Python
C
Assembly
Cobra
Delphi
NO duck face on 9GAG.COM

Introdução

- Características
  - Facilidade de aprendizado – indicado a iniciantes em programação
  - Código enxuto – mais legível e fácil de manter
  - Ampla biblioteca – portável e compatível
  - Suporte a: orientação a objetos, banco de dados, GUIs, Web

Instalação

- Interpretador Python (recomendo a versão 3.7.X)
  - https://www.python.org/download
  - Pode ser configurado para acesso direto no terminal

- IDE's para Python
  - A IDE padrão de Python é o IDLE, que já vem no instalador. Basta clicar com o botão direito em algum arquivo .py e “Edit with IDLE”
  - NetBeans: https://blogs.oracle.com/geertjan/entry/python_in_netbeans_ide_8
  - Eclipse: http://pydev.org/

Sintaxe

- Identificadores

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9).

- Palavras reservadas

And	Exec	Not
Assert	Finally	Or
Break	For	Pass
Class	From	Print
Continue	Global	Raise
Def	if	Return
Del	import	Try
Elif	in	While
Else	is	With
Except	lambda	Yield

Sintaxe

- Endentação

One of the first caveats programmers encounter when learning Python is the fact that there are no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced.

The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. Both blocks in this example are fine:

```
if True:
    print "True"
else:
  print "False"
```

However, the second block in this example will generate an error:

```
if True:
    print "Answer"
    print "True"
else:
    print "Answer"
  print "False"
```

Variáveis

Python variables do not have to be explicitly declared to reserve memory space. The declaration happens automatically when you assign a value to a variable. The equal sign (=) is used to assign values to variables.

The operand to the left of the = operator is the name of the variable and the operand to the right of the = operator is the value stored in the variable. For example:

```
#!/usr/bin/python

counter = 100          # An integer assignment
miles   = 1000.0       # A floating point
name    = "John"       # A string

print counter
print miles
print name
```

Here, 100, 1000.0 and "John" are the values assigned to *counter*, *miles* and *name* variables, respectively. While running this program, this will produce the following result:

```
100
1000.0
John
```

Tipos de Dados

- Number

Here are some examples of numbers:

Int	Long	float	complex
10	51924361L	0.0	3.14j
100	-0x19323L	15.20	45.j
-786	0122L	-21.9	9.322e-36j
080	0xDEFABCECBDAECBFBAEI	32.3+e18	.876j
-0490	535633629843L	-90.	-.6545+0J
-0x260	-052318172735L	-32.54e100	3e+26J
0x69	-4721885298529L	70.2-E12	4.53e-7j

Tipos de Dados

- String

```
#!/usr/bin/python

str = 'Hello World!'

print str          # Prints complete string
print str[0]       # Prints first character of the string
print str[2:5]     # Prints characters starting from 3rd to 5th
print str[2:]      # Prints string starting from 3rd character
print str * 2      # Prints string two times
print str + "TEST" # Prints concatenated string
```

This will produce the following result:

```
Hello World!
H
llo
llo World!
Hello World!Hello World!
Hello World!TEST
```

Tipos de Dados

- List

```
#!/usr/bin/python

list = [ 'abcd', 786 , 2.23, 'john', 70.2 ]
tinylist = [123, 'john']

print list          # Prints complete list
print list[0]       # Prints first element of the list
print list[1:3]     # Prints elements starting from 2nd till 3rd
print list[2:]      # Prints elements starting from 3rd element
print tinylist * 2  # Prints list two times
print list + tinylist # Prints concatenated lists
```

This will produce the following result:

```
['abcd', 786, 2.23, 'john', 70.200000000000003]
abcd
[786, 2.23]
[2.23, 'john', 70.200000000000003]
[123, 'john', 123, 'john']
['abcd', 786, 2.23, 'john', 70.200000000000003, 123, 'john']
```

Tipos de Dados

- Tuple

```
#!/usr/bin/python

tuple = ( 'abcd', 786 , 2.23, 'john', 70.2  )
tinytuple = (123, 'john')

print tuple           # Prints complete list
print tuple[0]        # Prints first element of the list
print tuple[1:3]      # Prints elements starting from 2nd till 3rd
print tuple[2:]       # Prints elements starting from 3rd element
print tinytuple * 2   # Prints list two times
print tuple + tinytuple # Prints concatenated lists
```

This will produce the following result:

```
('abcd', 786, 2.23, 'john', 70.200000000000003)
abcd
(786, 2.23)
(2.23, 'john', 70.200000000000003)
(123, 'john', 123, 'john')
('abcd', 786, 2.23, 'john', 70.200000000000003, 123, 'john')
```

Following is invalid with tuple, because we attempted to update a tuple, which is not allowed. Similar case is possible with lists:

```
#!/usr/bin/python

tuple = ( 'abcd', 786 , 2.23, 'john', 70.2  )
list = [ 'abcd', 786 , 2.23, 'john', 70.2  ]
tuple[2] = 1000    # Invalid syntax with tuple
list[2] = 1000     # Valid syntax with list
```

Tipos de Dados

- Dictionary

```
#!/usr/bin/python

dict = {}
dict['one'] = "This is one"
dict[2]     = "This is two"

tinydict = {'name': 'john','code':6734, 'dept': 'sales'}

print dict['one']       # Prints value for 'one' key
print dict[2]           # Prints value for 2 key
print tinydict          # Prints complete dictionary
print tinydict.keys()   # Prints all the keys
print tinydict.values() # Prints all the values
```

This will produce the following result:

```
This is one
This is two
{'dept': 'sales', 'code': 6734, 'name': 'john'}
['dept', 'code', 'name']
['sales', 6734, 'john']
```

Operadores Básicos

- Aritméticos

Assume variable a holds 10 and variable b holds 20, then:

Operator	Description	Example
+	Addition - Adds values on either side of the operator	a + b will give 30
-	Subtraction - Subtracts right hand operand from left hand operand	a - b will give -10
*	Multiplication - Multiplies values on either side of the operator	a * b will give 200
/	Division - Divides left hand operand by right hand operand	b / a will give 2
%	Modulus - Divides left hand operand by right hand operand and returns remainder	b % a will give 0
**	Exponent - Performs exponential (power) calculation on operators	a**b will give 10 to the power 20
//	Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed.	9//2 is equal to 4 and 9.0//2.0 is equal to 4.0

Operadores Básicos

- Comparação

Following table shows all the comparison operators supported by Python language. Assume variable a holds 10 and variable b holds 20, then:

Operator	Description	Example
==	Checks if the value of two operands is equal or not, if yes then condition becomes true.	(a == b) is not true.
!=	Checks if the value of two operands is equal or not, if values are not equal then condition becomes true.	(a != b) is true.
<>	Checks if the value of two operands is equal or not, if values are not equal then condition becomes true.	(a <> b) is true. This is similar to != operator.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(a > b) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(a < b) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(a >= b) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(a <= b) is true.

Operadores Básicos

- Atribuição

Assume variable a holds 10 and variable b holds 20, then:

Operator	Description	Example
=	Simple assignment operator, Assigns values from right side operands to left side operand	c = a + b will assign value of a + b into c
+=	Add AND assignment operator, It adds right operand to the left operand and assigns the result to left operand	c += a is equivalent to c = c + a
-=	Subtract AND assignment operator, It subtracts right operand from the left operand and assigns the result to left operand	c -= a is equivalent to c = c - a
*=	Multiply AND assignment operator, It multiplies right operand with the left operand and assigns the result to left operand	c *= a is equivalent to c = c * a
/=	Divide AND assignment operator, It divides left operand with the right operand and assigns the result to left operand	c /= a is equivalent to c = c / a
%=	Modulus AND assignment operator, It takes modulus using two operands and assigns the result to left operand	c %= a is equivalent to c = c % a
**=	Exponent AND assignment operator, Performs exponential (power) calculation on operators and assigns value to the left operand	c **= a is equivalent to c = c ** a
//=	Floor Dividion and assigns a value, Performs floor division on operators and assigns value to the left operand	c //= a is equivalent to c = c // a

Operadores Básicos

- Operadores Lógicos

There are following logical operators supported by Python language. Assume variable a holds 10 and variable b holds 20, then:

Operator	Description	Example
and	Called Logical AND operator. If both the operands are true, then the condition becomes true.	(a and b) is true.
or	Called Logical OR Operator. If any of the two operands are non zero, then the condition becomes true.	(a or b) is true.
not	Called Logical NOT Operator. Used to reverse the logical state of its operand. If a condition is true, then Logical NOT operator will make it false.	not(a and b) is false.

Operadores Básicos

- Pertinência a conjunto

Operator	Description	Example
In	Evaluates to true if it finds a variable in the specified sequence and false otherwise.	x in y, here **In** results in a 1 if x is a member of sequence y.
not in	Evaluates to true if it does not finds a variable in the specified sequence and false otherwise.	x not in y, here **not In** results in a 1 if x is not a member of sequence y.

Operadores Básicos

- Precedência

The following table lists all operators from highest precedence to lowest:

Operator	Description
**	Exponentiation (raise to the power)
~ + -	Complement, unary plus and minus (method names for the last two are +@ and -@)
* / % //	Multiply, divide, modulo and floor division
+ -	Addition and subtraction
>> <<	Right and left bitwise shift
&	Bitwise 'AND'
^ |	Bitwise exclusive `OR' and regular `OR'
<= < > >=	Comparison operators
<> == !=	Equality operators
= %= /= //= -= += *= **=	Assignment operators
is is not	Identity operators
in not in	Membership operators
not or and	Logical operators

Controle de Fluxo

- if – elif - else

```
#!/usr/bin/python

var = 100
if var < 200:
   print "Expression value is less than 200"
   if var == 150:
      print "Which is 150"
   elif var == 100:
      print "Which is 100"
   elif var == 50:
      print "Which is 50"
elif var < 50:
   print "Expression value is less than 50"
else:
   print "Could not find true expression"

print "Good bye!"
```

When the above code is executed, it produces the following result:

```
Expression value is less than 200
Which is 100
Good bye!
```

Controle de Fluxo

- while

The following program uses a nested for loop to find the prime numbers from 2 to 100:

```
#!/usr/bin/python

i = 2
while(i < 100):
   j = 2
   while(j <= (i/j)):
      if not(i%j): break
      j = j + 1
   if (j > i/j) : print i, " is prime"
   i = i + 1

print "Good bye!"
```

When the above code is executed, it produces the following result:

```
2 is prime
3 is prime
5 is prime
7 is prime
```

Controle de Fluxo

- for

```
#!/usr/bin/python

for letter in 'Python':     # First Example
   print 'Current Letter :', letter

fruits = ['banana', 'apple',  'mango']
for fruit in fruits:        # Second Example
   print 'Current fruit :', fruit

print "Good bye!"
```

When the above code is executed, it produces the following result:

```
Current Letter : P
Current Letter : y
Current Letter : t
Current Letter : h
Current Letter : o
Current Letter : n
Current fruit : banana
Current fruit : apple
Current fruit : mango
```

Controle de Fluxo

- for

An alternative way of iterating through each item is by index offset into the sequence itself. Following is a simple example:

```
#!/usr/bin/python

fruits = ['banana', 'apple',  'mango']
for index in range(len(fruits)):
   print 'Current fruit :', fruits[index]

print "Good bye!"
```

When the above code is executed, it produces the following result:

```
Current fruit : banana
Current fruit : apple
Current fruit : mango
Good bye!
```

Funções

- Exemplo:

```
#!/usr/bin/python

# Function definition is here
def sum( arg1, arg2 ):
   # Add both the parameters and return them."
   total = arg1 + arg2
   print "Inside the function : ", total
   return total;

# Now you can call sum function
total = sum( 10, 20 );
print "Outside the function : ", total
```

Módulos

- Exemplo:

The Python code for a module named *aname* normally resides in a file named *aname.py*. Here's an example of a simple module, hello.py

```
def print_func( par ):
   print "Hello : ", par
   return
```

You can use any Python source file as a module by executing an import statement in some other Python source file. The *import* has the following syntax:

```
import module1[, module2[,... moduleN]
```

When the interpreter encounters an import statement, it imports the module if the module is present in the search path. A search path is a list of directories that the interpreter searches before importing a module. For example, to import the module hello.py, you need to put the following command at the top of the script:

```
#!/usr/bin/python

# Import module hello
import hello

# Now you can call defined function that module as follows
hello.print_func("Zara")
```

When the above code is executed, it produces the following result:

```
Hello : Zara
```

Entrada / Saída

- Terminal

```
#!/usr/bin/python

print "Python is really a great language,", "isn't it?";
```

This would produce the following result on your standard screen:

```
Python is really a great language, isn't it?
```

The *raw_input([prompt])* function reads one line from standard input and returns it as a string (removing the trailing newline).

```
#!/usr/bin/python

str = raw input("Enter your input: ");
print "Received input is : ", str
```

This would prompt you to enter any string and it would display same string on the screen. When I typed "Hello Python!", its output is like this:

```
Enter your input: Hello Python
Received input is :  Hello Python
```

Entrada / Saída

- Arquivos: open e close

```
#!/usr/bin/python

# Open a file
fo = open("foo.txt", "wb")
print "Name of the file: ", fo.name

# Close opened file
fo.close()
```

This would produce the following result:

```
Name of the file:  foo.txt
```

Entrada / Saída

- Arquivos: write

```
#!/usr/bin/python

# Open a file
fo = open("foo.txt", "wb")
fo.write( "Python is a great language.\nYeah its great!!\n");

# Close opened file
fo.close()
```

The above method would create *foo.txt* file and would write given content in that file and finally it would close that file. If you would open this file, it would have following content:

```
Python is a great language.
Yeah its great!!
```

Entrada / Saída

- Arquivos: read

Let's take a file *foo.txt*, which we have created above.

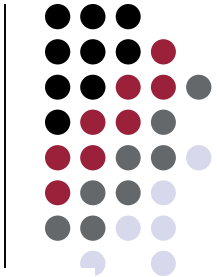
```
#!/usr/bin/python

# Open a file
fo = open("foo.txt", "r+")
str = fo.read(10);
print "Read String is : ", str
# Close opened file
fo.close()
```

This would produce the following result:

```
Read String is :  Python is
```

Exceptions



- Types

EXCEPTION NAME	DESCRIPTION
Exception	Base class for all exceptions
StopIteration	Raised when the next() method of an iterator does not point to any object.
SystemExit	Raised by the sys.exit() function.
StandardError	Base class for all built-in exceptions except StopIteration and SystemExit.
ArithmeticError	Base class for all errors that occur for numeric calculation.
OverflowError	Raised when a calculation exceeds maximum limit for a numeric type.
FloatingPointError	Raised when a floating point calculation fails.
ZeroDivisonError	Raised when division or modulo by zero takes place for all numeric types.
AssertionError	Raised in case of failure of the Assert statement.
AttributeError	Raised in case of failure of attribute reference or assignment.
EOFError	Raised when there is no input from either the raw_input() or input() function and the end of file is reached.
ImportError	Raised when an import statement fails.
KeyboardInterrupt	Raised when the user interrupts program execution, usually by pressing Ctrl+c.
LookupError	Base class for all lookup errors.
IndexError	Raised when an index is not found in a sequence.

Exceptions

- Types

KeyError	Raised when the specified key is not found in the dictionary.
NameError	Raised when an identifier is not found in the local or global namespace.
UnboundLocalError	Raised when trying to access a local variable in a function or method but no value has been assigned to it.
EnvironmentError	Base class for all exceptions that occur outside the Python environment.
IOError	Raised when an input/ output operation fails, such as the print statement or the open() function when trying to open a file that does not exist.
OSError	Raised for operating systemrelated errors.
SyntaxError	Raised when there is an error in Python syntax.
IndentationError	Raised when indentation is not specified properly.
SystemError	Raised when the interpreter finds an internal problem, but when this error is encountered the Python interpreter does not exit.
SystemExit	Raised when Python interpreter is quit by using the sys.exit() function. If not handled in the code, causes the interpreter to exit.
TypeError	Raised when an operation or function is attempted that is invalid for the specified data type.
ValueError	Raised when the built-in function for a data type has the valid type of arguments, but the arguments have invalid values specified.
RuntimeError	Raised when a generated error does not fall into any category.
NotImplementedError	Raised when an abstract method that needs to be implemented in an inherited class is not actually implemented.

Exceptions

- Sintax example

Here is one more simple example, which tries to open a file where you do not have permission to write in the file, so it raises an exception:

```
#!/usr/bin/python

try:
   fh = open("testfile", "w")
   fh.write("This is my test file for exception handling!!")
except IOError:
   print "Error: can\'t find file or read data"
else:
   print "Written content in the file successfully"
```

This will produce the following result:

```
Error: can't find file or read data
```

Exercícios

- Escreva um programa que leia o peso e a altura do usuário e calcule seu IMC:

$$IMC = \frac{peso}{altura^2}$$

posteriormente informe a qual faixa o usuário pertence:

- IMC < 18.5: abaixo do peso
- 18.5 ≤ IMC ≤ 25: peso ideal
- IMC > 25: sobrepeso

Exercícios

- Escreva um programa que leia os coeficientes a, b e c e calcula as raizes de uma equação do segundo grau de acordo com a formula de Bhaskara:

$$\Delta = b^2 - (4 \times a \times c)$$

$$x = \frac{-b \pm \sqrt{\Delta}}{2 \times a}$$

note que se $\Delta < 0$ não há raízes reais

Exercícios

- Escreva um programa que calcule a media dos elementos de uma lista

- Escreva um programa que recebe uma lista de inteiros e a separe em duas novas listas: uma dos números pares e outra dos números ímpares

- Escreva um programa que calcule o fatorial de um número inteiro fornecido pelo usuário. Ex.:

$$5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$$

Exercícios

- Escreva um programa que busca um elemento em uma lista. Caso o elemento seja encontrado exiba seu índice na lista; caso contrário, informe que o elemento não foi encontrado

- Escreva um programa que some os n termos da série a seguir:

$$s = 1/1 + 2/3 + 3/5 + 4/7 + 5/9 + \ldots + n/m$$

Exercícios

- Escreva um programa que peça dois números, base e expoente, calcule e mostre o primeiro número elevado ao segundo número. Não utilize a função de potência da linguagem.

- Escreva um programa que encontre o menor elemento de uma lista de inteiros e exiba seu índice

Exercícios

- Escreva um programa para ler uma lista de 10 números. Após isto, ler mais um número qualquer, calcular e escrever quantas vezes esse número aparece na lista.

- Um número primo é aquele que é divisível apenas por um e por ele mesmo. Escreva um programa que peça um número inteiro e determine se ele é ou não um número primo.

- Escreva um programa que imprime todos os números primos de 2 a 100

Exercícios

- Escreva um programa que ordene uma lista de inteiros

- Escreva um programa que leia um tamanho n e imprima o seguinte triângulo retângulo formado por asteriscos:

```
*
* *
* * *
* * * *
```

- Escreva um programa que leia um tamanho n e imprima o seguinte triângulo retângulo formado por asteriscos:

```
      *
    * *
  * * *
* * * *
```

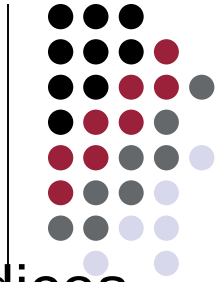
Exercícios

- Escreva um programa que faça e exiba a matriz resultante da soma de duas matrizes

- Escreva uma função que recebe dois parâmetros: taxaImposto, que é a quantia de imposto sobre vendas expressa em porcentagem e custo, que é o custo de um item antes do imposto. A função retorna o valor de custo para incluir o imposto sobre vendas.

- Escreva uma função que recebe uma matriz e retorna o seu menor e o seu maior elemento

Exercícios



- Escreva uma função que recebe uma lista e dois índices i e j e retorne a lista com as posições i e j trocadas

- Escreva uma função que informe a quantidade de dígitos de um determinado número inteiro informado.

- Escreva uma função que recebe dois vetores v1 e v2 de 10 números cada, calcula e retorna a quantidade de vezes que v1 e v2 possuem os mesmos números nas mesmas posições.

- Escreva uma função que recebe uma lista e retorna outro lista contendo seus elementos na ordem reversa

Exercícios

- Implemente uma classe Carro com os seguintes atributos:
  - consumo de combustível (km/l)
  - quantidade de combustível no tanque

  e dois métodos:
  - andar( ) que recebe uma quantidade de quilômetros e simula o ato de dirigir, reduzindo a quantia de combustível no tanque de acordo com o consumo
  - abastecer( ) que recebe uma quantidade de combustível e altera a quantidade de combustível no tanque

Exercícios

- Implemente uma classe BombaCombustivel com os seguintes atributos:
  - tipo de combustível
  - valor do litro
  - quantidade de combustível

  e cinco métodos:
  - abastecerValor( ) que recebe um valor e altera a quantidade de combustivel na bomba
  - abastecerLitros( ) que recebe um número de litros e altera a quantidade de combustível na bomba
  - alterarValor( ) que altera o valor do litro de combustível
  - alterarTipo( ) que altera o tipo de combustível
  - alterarQuantidade( ) que altera a quantidade de combustível na bomba

Bibliografia

- Rossum, G. V. *Python Tutorial.* Disponível em http://www.tutorialspoint.com/python/python_tutorial.pdf, acessado em Setembro de 2014.

- Não conseguiu algo? https://www.google.com.br Resolve seus problemas! =)