Python Básico

Por onde começo ?

... Criando nosso primeiro Hello World !

Hello World

... 'hello world' - Python X {Java, C, PHP, Pascal}

```
program helloworld;
begin
     writeln<'Hello World!'>;
end
```

```
class HelloWorld
{
    public static void main(String[] args)
    {
        System.out.println("Hello World!");
    }
}
```

```
#include <stdio.h>
int main (void)
{
    printf("Hello World");
    return 0;
}
```

```
<?php
    echo "Hello World";
?>
```

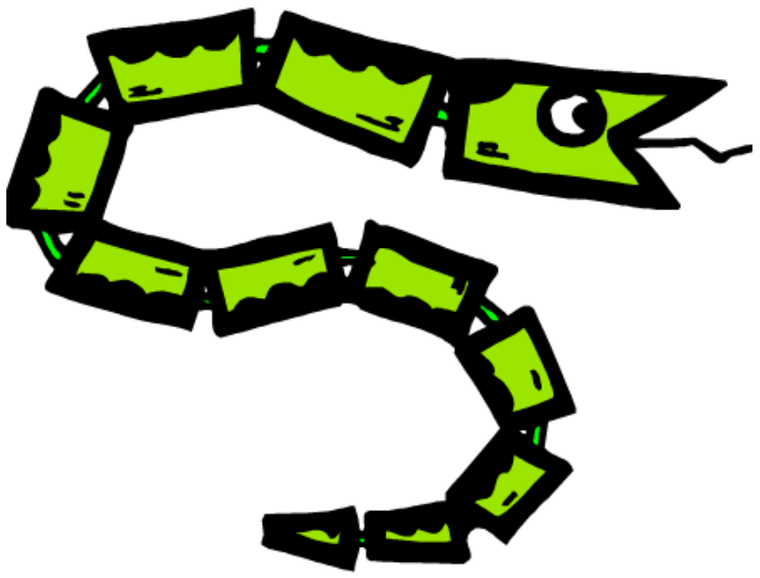
... em Python ...

print “Hello World”

Exemplo1.py

Tipos e operações

Vamos ver um trecho de código em Python!



Código Base

```
x = 34 - 23                    #Um comentário
y = "Hello"                    #Outro comentário
z = 3.45
if z == 3.45 or y == "Hello":
    x = x + 1
    y = y + "World"      #Concatenação de String
print x
print y
```

... entendendo o código...

- Atribuição utiliza = e comparação utiliza ==

... entendendo o código...

- Atribuição utiliza = e comparação utiliza ==

```
x = 34 - 23                    #Um comentário
y = "Hello"                    #Outro comentário
z = 3.45
if z == 3.45 or y == "Hello":
    x = x + 1
    y = y + "World"      #Concatenação de String
print x
print y
```

... entendendo o código...

- Números: + - * / % tem suas funções características
- + pode ser usado como concatenação de Strings;
- % pode ser usado para formatar Strings (assim como em C).

... entendendo o código...

- Números: + - * / % tem suas funções características
- + pode ser usado como concatenação de Strings;
- % pode ser usado para formatar Strings (assim como em C).

```
x = 34 - 23                 #Um comentário
y = "Hello"                 #Outro comentário
z = 3.45
if z == 3.45 or y == "Hello":
    x = x + 1
    y = y + "World"         #Concatenação de String
print x
print y
```

... entendendo o código...

- Operadores lógicos são palavras e não símbolos (||, &&)
- and, or, not

... entendendo o código...

- Operadores lógicos são palavras e não símbolos (||, &&)

- and, or, not

```
x = 34 - 23                    #Um comentário
y = "Hello"                    #Outro comentário
z = 3.45
if z == 3.45 or y == "Hello":
    x = x + 1
    y = y + "World"            #Concatenação de String
print x
print y
```

... entendendo o código...

- print é o comando básico para "impressão" na tela

... entendendo o código...

- print é o comando básico para "impressão" na tela

```
x = 34 - 23                    #Um comentário
y = "Hello"                    #Outro comentário
z = 3.45
if z == 3.45 or y == "Hello":
    x = x + 1
    y = y + "World"        #Concatenação de String
print x
print y
```

... entendo o código...

- E se você quiser receber uma entrada diretamente do usuário ?
  - *raw_input() - retorna uma string !*

```
>>> raw_input('Digite um valor')
```

Exemplo1

... entendendo o código...

- A primeira atribuição em uma variável também é responsável por cria-lá.
  - Os tipos das variáveis não precisam ser informados;
  - Python descobre o tipo da variável por conta própria!

... entendendo o código...

- A **primeira atribuição** em uma variável também é responsável por cria-lá.
  - Os tipos das variáveis não precisam ser informados;
  - Python descobre o tipo da variável por conta própria!

```
x = 34 - 23                 #Um comentário
y = "Hello"                 #Outro comentário
z = 3.45
if z == 3.45 or y == "Hello":
    x = x + 1
    y = y + "World"         #Concatenação de String
print x
print y
```

... Usando o Shell

- Para iniciar o shell basta digitar o comando
  - #> python
- Quando o shell é iniciado aparecerão três '>' (">>>") indicando que ele está ativo e pode receber comandos
  - Exemplo
    - #> python
    - >>> print "HelloWorld!!!"
    - HelloWorld!!!
    - >>>

... Usando o Shell

- Para obter informações como métodos e atributos de um objeto basta executar o comando “dir”. Obs.: Tudo em Python é objeto!
  - >>>dir(“string de teste”)
  - <tudo sobre strings!>
  - >>>
- Para visualizar a documentação de um Objeto basta executar o comando “help”
  - >>>help(1000)
  - <Documentação do objeto Inteiro>

... Usando o Shell

- Para repetir o comando anterior pode-se
  - Usar a seta para cima
  - Digitar '_'
- Para navegar entre os comando já executados basta usar as setas para cima e para baixo
- Para obter ajuda geral executa-se o comando "help()"
  - Para sair do help "quit"
  - Para obter a lista dos módulos "modules"

Whitespace

- Importante para identação e novas linhas
  - Use \ para quando for para uma próxima linha prematuramente.
- Em Python não há { } !! Isso é para definição de dicionários (*dict*)
- Blocos de código definidos por identação!

Exemplo1

Comentários

- Comentários começam com #
  - Convenção: Você pode definir uma "documentação" em string como primeira linha de qualquer nova função que você definir.
  - Muito importante para o desenvolvedor, crítico para o usuário!

```
def my_function(x, y):
    """This is the docstring. This
    function does blah blah blah."""
    # The code would go here...
```

Conhecendo a linguagem...

- Dinamicamente tipada
  - Exemplo
    - >>>a = 10
    - >>>a = “teste”
    - >>>
- Fortemente tipada, não existe cast.
  - Se quiser mudar o tipo, use uma função
    - Exemplo
      - >>>a = (int) 1.0 # ERRO!!!
      - >>>a = int(1.0)

Conhecendo a linguagem...

- Não possui declaração de tipos
  - Java
    - int a = 0;
  - Python
    - a = 0
- Não possui comandos declarativos (“óbvios”)
  - Java
    - Algo n = new Algo();
  - Python
    - n = Algo()

Tipos Básicos

- Inteiros (padrão para números)
  - Divisão entre inteiros, resposta um inteiro!
- Inteiros Longos
  - L ou l no final. (Convertido automaticamente com precisão de inteiros > 32 bits)
- Floats (ponto flutuante)
  - 1.23, 3.4e-10
- Complexas
  - >> 2 + 3j
- Operações válidas: +, *, >> , ** , pow, abs, etc.

Exemplo1

Tipos Básicos

- Representação numérica
  - Representação de dígitos com/sem formatação de string
- Divisão clássica / base
  - Uso dos operadores // e /
- Operações em nível de bit
  - 1 << 2 , 1| 2 , 1&2
- Notações hexadecimal / octal
  - 2 , 0x10 , 0100 , oct(64), hex(255) , int('200'), int('0100',8) , int('0x40',16)
- Operações válidas: +, *, >> , ** , pow, abs, round, etc

Exemplo1

Tipos Básicos

Operação	Resultado
x + y	Soma dos valores x e y
x - y	Subtração de x por y
x * y	Multiplicação de x por y
x / y	Divisão de x por y
x // y	Divisão de x por y, obs.: Pegando o piso.
x % y	Resto da divisão de x por y
+x	Não altera nada
-x	Inverte o sinal de x
abs(x)	Valor absoluto de x
int(x)	x convertido em inteiro
long(x)	x convertido em long
float(x)	x convertido em float
complex(re, im)	Um número complexo com parte real re e imaginária im
x ** y	x elevado a y
pow(x, y)	x elevado a y

Obs.: as operações citadas acima valem para todos os tipos numéricos exceto números complexos

Operação	Resultado
x | y	Bit a bit **ou** de x e y
x ^ y	Bit a bit **ou exclusivo** de x e y
x & y	Bit a bit **e** de x e y
x << n	*x deslocado à esquerda n bits*
x >> n	*x deslocado à direito n bits*
~x	os bits de x invertidos

Exemplo1.py

Tipos Básicos

- Strings
  - “abc” ou ‘abc’
- Operadores de expressão de Python e sua precedência
- http://docs.python.org/reference/expressions.html#summary

Símbolo	ação comparativa
“<”	Menor que
“<=”	menor ou igual
“>”	maior que
“>=”	maior ou igual
“==”	igual (objeto ->referência)
“!=”	diferente
“<>”	diferente
“is”	igualdade de objetos
“is not”	diferença de objetos

```
>>>True > False
<Qual seria o resultado???>
```

Exemplo1.py

Comandos básicos

- Alguns comandos básicos que podem ajudar no ínicio!
  - dir(element) - todos os atributos e métodos que estão associados a elemento.
  - type(element) - Descobrir o tipo do objeto!
  - import - importe módulos para uso no seu código!

```
import modulo
import modulo.algo
import modulo.algo as malg
from modulo import *
from modulo import algo
from modulo import algo.item as ait
```

Exercício 01

- Faça um programa que peça 2 números e um real.
  - Calcule e mostre:
    - O produto do dobro do primeiro com a metade do segundo
    - A soma do triplo do primeiro com o terceiro
    - O terceiro elevado ao cubo