

# python™



Sjsoft, <http://westmarch.sjsoft.com/2012/11/zen-of-python-poster/>

## PROGRAMAÇÃO E PYTHON

# Porquê Programar?



- Com ferramentas resolvem-se problemas
  - ▣ Aplicando soluções existentes
- Programando resolvem-se **novos** problemas
  - ▣ Ou velhos problemas de novas maneiras
- Tudo são *bits* e algoritmos
  - ▣ Som, imagem, documentos, música, etc...

# Linguagens



- Linguagens são ferramentas
  - ▣ Um mecânico tem várias chaves
  
- Existem diferentes necessidades:
  - ▣ Aplicações
  - ▣ Páginas Web
  - ▣ Aplicações Móveis
  - ▣ Desenvolvimento rápido
  - ▣ Velocidade de execução
  - ▣ Compreensão
  - ▣ Etc...

# Porquê Python



- Java: aplicações, serviços, web, mobile
  - ▣ Desenvolvimento rápido
  - ▣ Linguagem compilada
  - ▣ Execução universal (sobre VMs)
- Javascript: páginas e serviços web
  - ▣ Linguagem interpretada

# Python



- Python: aplicações, serviços, web, mobile
- Linguagem interpretada
  - ▣ Execução universal (com interpretadores)
- Desenvolvimento muito rápido
  - ▣ Prototipagem
- Linguagem obriga a formatação rígida
  - ▣ “Hacks” são sempre formatados corretamente

# Python



- Nome: Monty **Python**'s Flying Circus
- Combina funcionalidades modernas
  - ▣ Encontradas no Java, C#, Ruby, C++, etc...
- Com um estilo conciso e simples

# Zen of Python



- Python possui um código de princípios
- Guiam a linguagem e os programas que a utilizam

```
$> python3
```

```
>>> import this
```

# *Simple is better than complex*

## □ Só existem 33 palavras reservadas

- ▣ Java: ~50
- ▣ JavaScript: ~60 + ~111 (DOM)
- ▣ C++: ~50
- ▣ C#: ~80

<b>False</b>	<b>None</b>	<b>True</b>	<b>and</b>	<b>as</b>
<b>assert</b>	<b>break</b>	<b>class</b>	<b>continue</b>	<b>def</b>
<b>del</b>	<b>elif</b>	<b>else</b>	<b>except</b>	<b>finally</b>
<b>for</b>	<b>from</b>	<b>global</b>	<b>if</b>	<b>import</b>
<b>in</b>	<b>is</b>	<b>lambda</b>	<b>nonlocal</b>	<b>not</b>
<b>or</b>	<b>pass</b>	<b>raise</b>	<b>return</b>	<b>try</b>
<b>while</b>	<b>with</b>	<b>yield</b>		

Obtido com:  
import keyword  
print(keyword.kwlist)



# *Beautiful is better than ugly*



- Indentação define um bloco
  - ▣ Sempre com espaço ou tabulação (nunca ambos!)
  - ▣ 4 espaços
- ENTER delimita fim de linha
- Nomes usam separador "\_"
  - ▣ Ex: processa\_ficheiro

# Python: Hello World! (mínimo)



Ficheiro hello.py

```
# File: hello.py  
  
print("hello world")
```

Consola

```
$> python3 hello.py  
  
hello world
```

# Variáveis

- Declaram-se sem tipo
  - Tipo dinâmico

```
# File: vars.py
```

```
a = 3
```

```
b = 5.2
```

```
print(a * b)
```

```
a = "var"
```

# Variáveis String

- Podem ser tratadas como os *arrays* em Java
- Não existe *char* (é uma *string* com 1 carácter)
- Tamanho dado por função *len*

```
a = "hello"
```

```
b = "world"
```

```
print(a+" "+b)
```

```
print(a[1])
```

```
print(a[1:4])
```

```
print(len(a))
```

```
hello world
```

```
e
```

```
ell
```

```
5
```

# Variáveis String

- ❑ Concatenação com inteiros NÃO funciona
  - ❑ Necessário converter inteiros em String

```
r = 42
```

```
s = "A resposta para a vida, o Universo e tudo mais é: "
```

```
print(s + r)
```

```
print(s + str(r))
```

**TypeError: must be str, not int**

**A resposta para a vida, o Universo e  
tudo mais é: 42**

# Variáveis String

- Não existe *printf*
- Mas é possível formatar *strings*

```
r = 42
```

```
s = "A resposta para a vida, o Universo e tudo mais é: "
```

```
print("%s %d" % (s, r))
```

A resposta para a vida, o Universo e tudo mais é: 42

# Condições

- Usam-se operadores “and”, “or”, “not” explícitos

```
ano = 2000
if (ano % 4==0 and ano % 100 != 0) or ano % 400== 0:
    bissexto = True
else:
    bissexto = False

if bissexto:
    ndias = 29
else:
    ndias = 28
```

# *Beautiful is better than ugly*



## **ERRADO**

```
if a == 3 and b == False: print("3")
```

## **CORRETO**

```
if a == 3 and not b:  
  
    print("3")
```



# Ciclos: For



```
for i in range(1,10):  
    print(i)
```

```
1  
2  
3  
...  
9
```

# Ciclos: Range

- Cria uma lista entre 2 valores *start* e *stop* excluindo o valor final ( **[start .. stop[** ) com incremento constante.

**range(start, stop [, step])**

- Se o incremento *step* for omitido assume incremento unitário.
- Podemos ter listas crescentes e decrescentes (*step* negativo).

# Ciclos: While



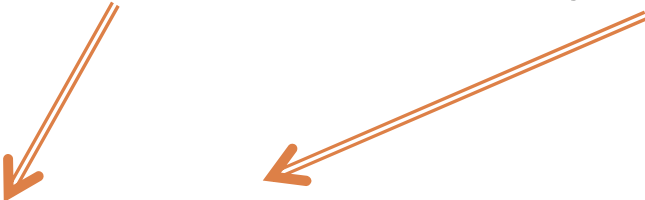
```
a = 3  
while a > 0:  
    print(a)  
    a = a - 1
```

```
3  
2  
1
```

# Funções

Declaração de função

Argumentos



```
def foo(name):
```

```
    print("Olá: " + name)
```

```
foo("Pedro")
```

**Indentação define bloco**

# Funções

Declaração de função

Ciclo while

```
def factorial(x):  
    a = 1  
    while x > 0:  
        a = a * x  
        x = x - 1  
    return a
```

Declaração de variável e atribuição

**Indentação define bloco**

# Listas

- Python não possui *arrays* como o Java
- Lista é o mais semelhante

```
a = [1, 2, 3]
```

```
print(a[1])
```

```
print(len(a))
```

```
for v in a:
```

```
    print(v)
```

2

3

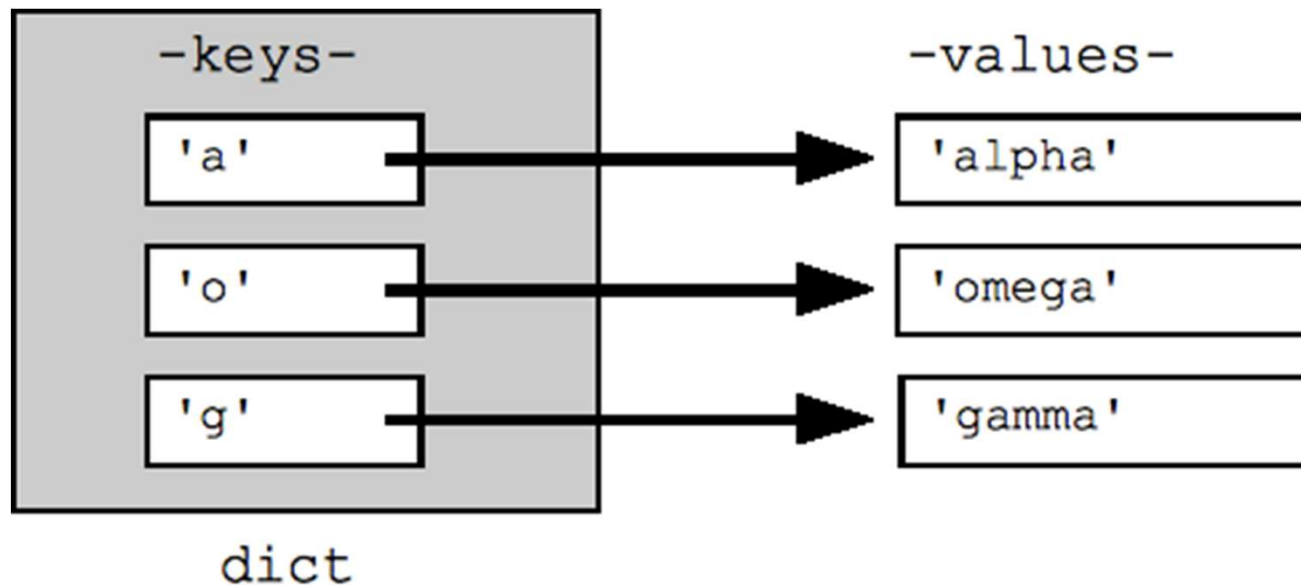
1

2

3

# Dicionários

- Estrutura que mapeia **chave** a **valor**
- Elementos não possuem ordem



# Dicionários

```
d = {"nome": "Pedro", "mec": 123, "turma": 0}
```

```
d["turma"] = "TP5"
```

```
print(d["nome"])
```

```
print(d)
```

**Pedro**

```
{'nome': 'Pedro', 'mec': 123, 'turma': 'TP5'}
```



# Módulos



- Funcionalidades adicionais são fornecidas em módulos
- Adicionados ao programa com “*import*”
  - ▣ Semelhante ao Java
- Cada programa usa módulos conforme necessário

# Módulos

- Programa imprime o número e conteúdo dos argumentos passados
  - ▣ Argumentos presentes numa lista `sys.argv`
  - ▣ `sys.argv[0]` contém o nome do programa

```
import sys
```

```
print("Número: %d" % len(sys.argv) )
```

```
print("Valores: %s" % (sys.argv) )
```

**Número: 4**

**Valores: ['modules.py', 'a', 'b', 'c']**

# Para Referência



- ❑ Python Docs: <https://docs.python.org/3>
- ❑ Code Like a Pythonist:  
<http://python.net/~goodger/projects/pycon/2007/idiomatic/handout.html>
- ❑ Learn Python: <http://www.learnpython.org/>
- ❑ Think Python: <http://greenteapress.com/wp/think-python-2e/>