



INTRODUÇÃO AO PYTHON

PYTHON



INTRODUÇÃO A PYTHON

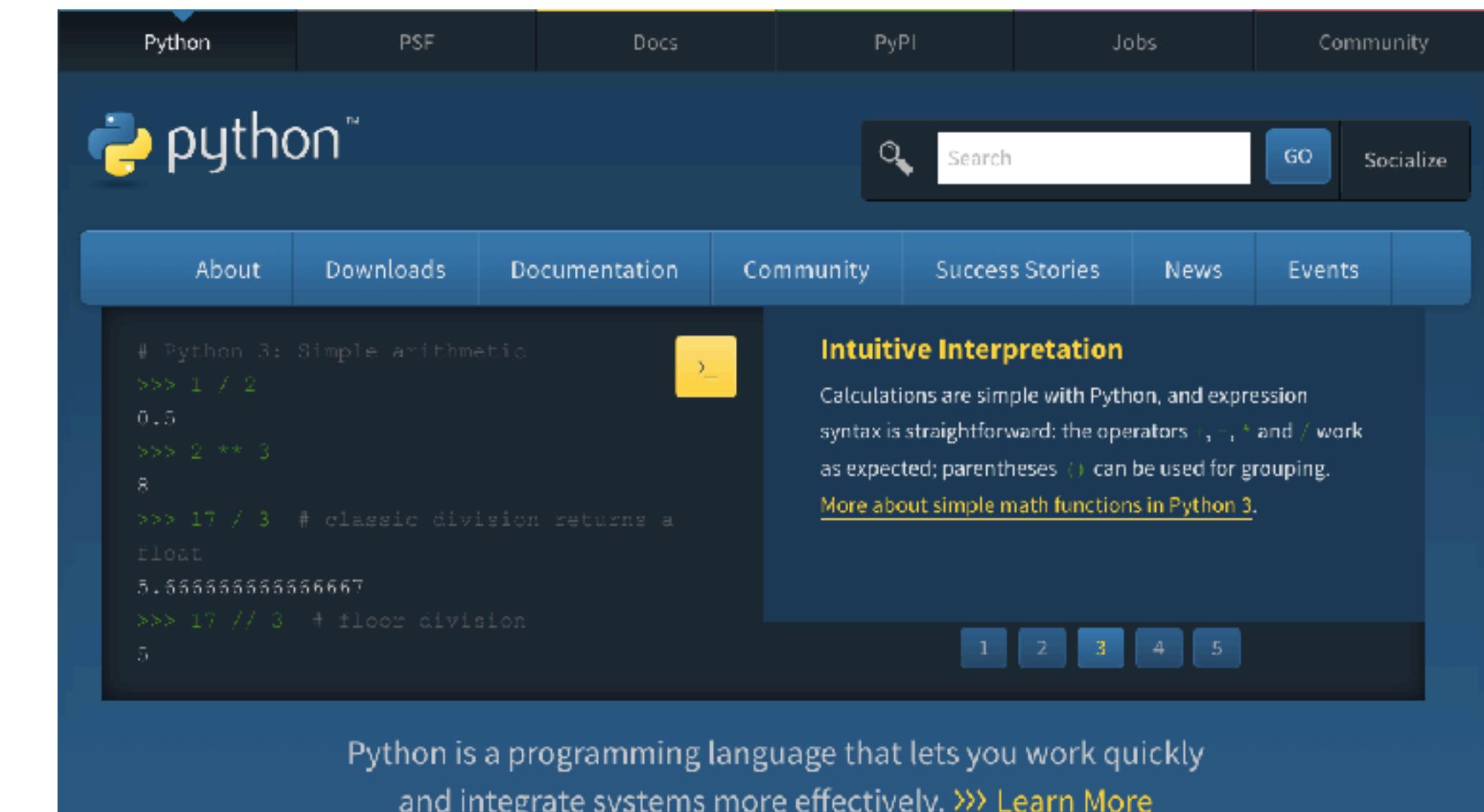
Desmistificando Python

CARACTERÍSTICAS

Python: linguagem de programação multiparadigma:
script, modular / funcional e ordenado a objetos:

- Sintaxe clara
- Estruturas de dados
- Quantidade elevada de bibliotecas para diversas finalidades
- Desenvolvimento para script, desktop, web e mobile
- Permite adicionar frameworks de terceiros
- Linguagem fortemente ligada à infraestrutura

www.python.org



MERCADO PYTHON

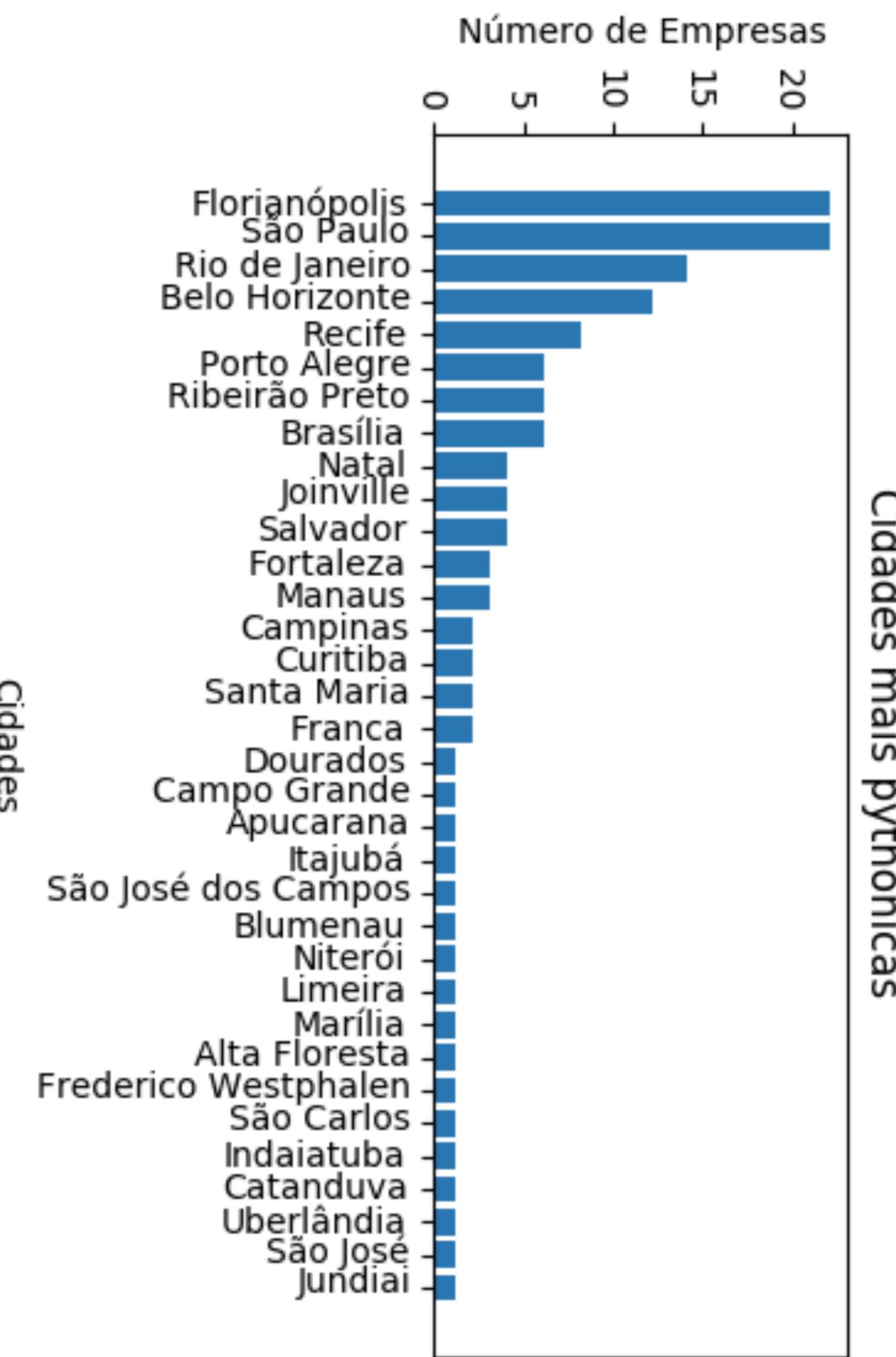
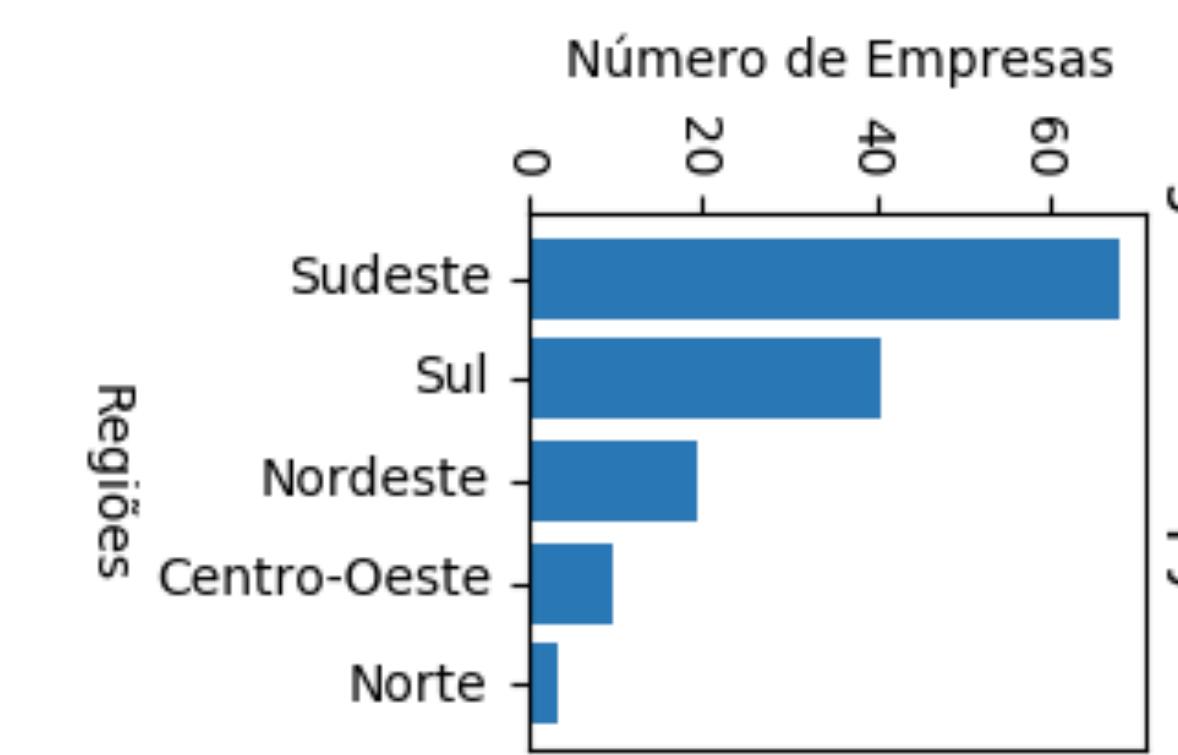
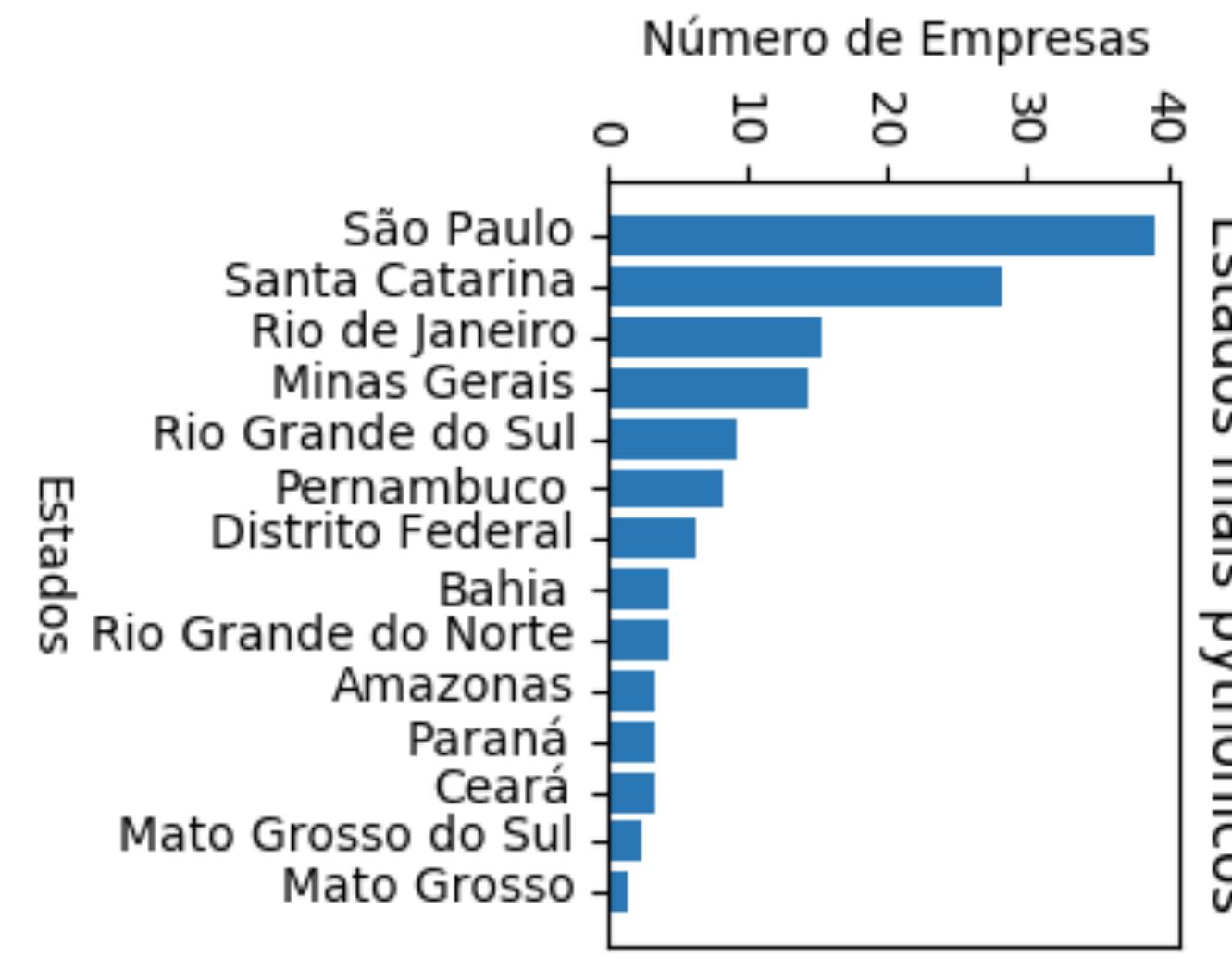


Google



YAHOO!

Disney



IDE E FRAMEWORKS DE DESENVOLVIMENTO PARA PYTHON



IDE DE DESENVOLVIMENTO



PyDev on
eclipse



SPYDER

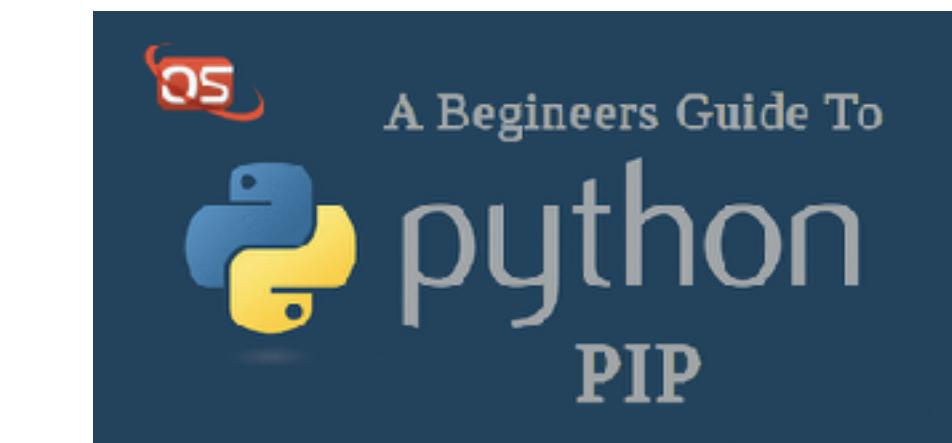


Visual Studio Code



IDLE

BIBLIOTECAS E FRAMEWORK



IP[y]: IPython
Interactive Computing



django



CONDA



PYTHON'S INTEGRATED DEVELOPMENT AND LEARNING ENVIRONMENT - IDLE



Python 3.6.3 (v3.6.3:2c5fed86e0, Oct 3 2017, 00:32:08)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>> WARNING: The version of Tcl/Tk (8.5.9) in use may be unstable.
Visit <http://www.python.org/download/mac/tcltk/> for current information.

```
>>> |
```

Teclas de Atalho	Descrição
Ctr+a	Inicio da linha
Ctr + e	Fim da linha
Ctr + k	Matar linha
Ctr + l	Limpa a tela
Ctr + b	Retrocede um caracter sem excluir
Ctr + f	Avança um caracter sem excluir
Ctr + p	Acessar comandos executados anteriormente
Ctr + d	Excluir próximo caracter



TIPOS E ESTRUTURAS DE DADOS

Armazenando dados com Python

TIPOS DE DADOS: BOOLEANO, NUMÉRICO, TEXTO E COLEÇÕES



Variáveis Python:

- Dinamicamente Tipada
- Sensitive Case

Tipos de dados:

- **bool**: valores booleanos **True** ou **False**
- **None**: valor especial **null**
- **int**: número inteiro de precisão ilimitada
- **complex**: número complexo
- **float**: número de ponto-flutuante IEEE 754
- **str**: sequência imutável de caracteres Unicode
- **bytes**: sequência imutável de bytes
- **list**: sequência mutável de objetos
- **tuple**: sequência imutável de objetos
- **set**: coleção mutável de objetos únicos e mutáveis
- **dict**: dicionário mutável mapeia key-objetos imutáveis

Tipo	Descrição	Exemplos
Boolean	<code>bool</code>	b = True ou b = False
Numérico	<code>int</code>	n = 1
	<code>float</code>	pi = 3.14
	<code>complex</code>	c = 3 + 4j
Texto	<code>str</code>	s = 'simples' ou s = "dupla"
	<code>list</code>	l = [1, 2, 'AbC', [2,3,4]]
Coleções	<code>tuple</code>	t = ('Nome', 20,'M')
	<code>dict</code>	d = {'nome': " Nome", 'idade':20,}
	<code>set</code>	s = {1,2,3,4,5,}

TIPOS NUMÉRICOS - INT, FLOAT E COMPLEX



- No modo interativo da IDLE do Python, pode-se visualizar os resultados de modo instantâneo:

```
>>> Copy  
2 + 2  
4  
>>> 6 * 7  
42
```

- No modo interativo da IDLE do Python, pode-se declarar varáveis:

```
>>> Copy  
x = 5
```

- No modo interativo da IDLE do Python, pode-se imprimir o conteúdo de uma variável:

```
>>> Copy  
x  
5
```

TIPOS NUMÉRICOS - INT, FLOAT E COMPLEX



- No modo interativo da IDLE do Python, referenciar expressões:

```
>>> 3 * x  
15
```

[Copy](#)

- No modo interativo da IDLE do Python, utilizar o sublinhado para imprimir o ultimo valor impresso:

```
>>> _  
15
```

[Copy](#)

```
>>> _ * 2  
30
```

[Copy](#)

```
>>> print('Hello, Python')  
Hello, Python
```

[Copy](#)

TIPOS NUMÉRICOS - INT, FLOAT E COMPLEX



- No modo interativo da IDLE do Python, permite-se utilizar estruturas de controle como if, while, for dentre outros:

```
[>>> for i in range(5):
[...     print(i)
[...
[...
0
1
2
3
4
>>>
```

- ▶ TIPOS NUMÉRICOS
 - ▶ OPERADORES ARITMÉTICOS
 - ▶ OPERADORES LÓGICOS
 - ▶ MÓDULO MATH
-

TIPOS E ESTRUTURAS DE DADOS

TIPOS NUMÉRICOS - INT, FLOAT E COMPLEX



NUMEROS INTEIROS - int

- Número inteiro:

```
>>> 10  
10
```

[Copy](#)

- Número binário:

```
>>> 0b10  
2
```

[Copy](#)

- Número octal:

```
>>> 0o10  
8
```

[Copy](#)

- Número hexadecimal:

```
>>> 0x10  
16
```

[Copy](#)

TIPOS NUMÉRICOS - INT, FLOAT E COMPLEX



NUMEROS REAIS - float

- Número float:

```
3.125
```

```
>>> 3e8
300000000.0
```

```
>>> 1.616e-35
1.616e-35
```

```
>>> float(7)
7.0
```

SÍMBOLOS ESPECIAIS

Funções	Descrição
NaN	Not a Number - Número inválido
inf / -inf	inf ou -inf - mais ou menos infinito

```
>>> float("nan")
nan
>>> float("inf")
inf
>>> float("-inf")
-inf
```

TIPOS NUMÉRICOS - INT, FLOAT E COMPLEX



VALOR NULL - None

- Valor nulo :

Copy

```
>>> None  
>>>
```

Copy

```
>>> a = None
```

Copy

```
>>> a is None  
True
```

TIPOS NUMÉRICOS - INT, FLOAT E COMPLEX



- Descobrindo as funções que pertencem à um módulo Python:

```
>>> help
Type help() for interactive help, or help(object) for help about object.
```

Copy

- Importar módulos

```
>>> import math
```

Copy

```
>>> math.sqrt(81)
9.0
```

Copy

TIPOS NUMÉRICOS - INT, FLOAT E COMPLEX



- Descobrindo as funções que pertencem à um módulo Python:

Copy

```
>>> help(math)
Help on module math:

NAME
    math

MODULE REFERENCE
    http://docs.python.org/3.3/library/math
```

The following documentation is automatically generated from the Python source files. It may be incomplete, incorrect or include features that are considered implementation detail and may vary between Python implementations. When in doubt, consult the module reference at the location listed above.

DESCRIPTION

This module is always available. It provides access to the mathematical functions defined by the C standard.

TIPOS NUMÉRICOS - INT, FLOAT E COMPLEX



- Descobrindo as funções que pertencem à um módulo Python:

Copy

```
>>> help(math.factorial)
Help on built-in function factorial in module math:

factorial(...)
    factorial(x) -> Integral

    Find x!. Raise a ValueError if x is negative or non-integral.
```

TIPOS NUMÉRICOS - INT, FLOAT E COMPLEX



- Utilizando a função de um módulo:

```
>>> n = 5  
>>> k = 3  
>>> math.factorial(n) / (math.factorial(k) * math.factorial(n - k))  
10.0
```

```
>>> from math import factorial  
>>> factorial(n) / (factorial(k) * factorial(n - k))  
10.0
```

Problema:



Vamos usar os fatoriais para calcular quantas maneiras existem para extrair três frutas de um conjunto de cinco frutas.

OPERADORES ARITMÉTICOS



- Operadores Aritméticos:

```

1 ⊕ '''
2 LAB03 - Operacoes Aritmeticas
3
4 @author: daniel
5 ...
6 x = 10
7 y = 2
8
9 print(x+y)
10 print(x-y)
11 print(x*y)
12 print(x/y)
13 print(x//y)
14 print(x%y)
15 print(x**y)
16 print(-x)
17 print(abs(-x))
18 print(divmod(x,y))
19 print(round(1.3245678,4))

```

Operadores	Descrição
x + y	Soma x e y
x - y	Subtrai x e y
x * y	Multiplica x e y
x / y	Divide x por y
x // y	Trunca a divisão de x por y
x % y	Retorna o resto da divisão de x por y
x ** y ou pow(x,y)	Eleva x por y
-x	Nega ou muda o sinal de x
abs(x)	Retorna o valor absoluto de x
divmod(x,y)	Retorna uma dupla com o quociente e resto da divisão de x por y
round(x)	Retorna o número float do inteiro x com n casas decimais

CONVERSÕES NUMÉRICAS



- Conversões numéricas:

```

1 ⊕ """
2 LAB01 - Conversao de Bases Numericas
3 @autor: Daniel Correa da Silva
4 """
5
6 s = '12345'
7 nDecimal = 255;
8 nBinary = 0b101010
9 nOctal = 0o074356
10 nHex = 0xFFFF
11
12 print('string -> inteiro:', int(s))
13 print('inteiro -> binario:', bin(nDecimal))
14 print('inteiro -> hexadecimal:', hex(nDecimal))
15 print('inteiro -> octal',oct(28) )
16 print('string -> base(1 -36):', int('255',16) )

```

Funções	Descrição
bin(x)	Retorna uma string de dígitos binários equivalente ao decimal x
hex(x)	Retorna uma string de dígitos hexadecimais equivalente ao decimal x
int(x)	Converte o objeto x em um número inteiro
int(s,base)	Converte a string s em um inteiro
oct(x)	Retorna uma string de dígitos cotais equivalentes ao decimal x

MÓDULO - MATH



```

6 import math
7
8 rad = math.pi/4;          #MEDIDA EM RADIANOS
9 graus = math.degrees(rad) #MEDIDA EM GRAUS
10 f = math.e
11 x = 9
12 y = 3
13
14 print( rad,graus )
15 print( math.cos(rad) )
16 print( math.sin(rad) )
17 print( math.tan(rad) )
18 print( math.acos(rad) )
19 print( math.asin(rad) )
20 print( math.atan(rad) )
21 print( math.cosh(rad) )
22 print( math.sinh(rad) )
23 print( math.tanh(rad) )
24 print( f )
25 print( math.ceil(f) )
26 print( math.floor(f) )
27 print( math.exp(x) )
28 print( math.log(x,y) )
29 print( math.log10(y) )
30 print( math.modf(f) )
31 print( math.fmod(x,y) )

```

FUNÇÕES	Descrição
math.acos(x)	Arco coseno de x em radianos
math.acosh(x)	Arco coseno hiperbólico de x em radianos
math.asin(x)	Arco seno de x em radianos
math.asinh(x)	Arco seno hiperbólico de x em radianos
math.atan(x)	Arco tangente de x em radianos
math.atanh(x)	Arco tangente hiperbólico de x em radianos
math.sin(x)	Retorna seno de x em radianos
math.cos(x)	Retorna coseno de x em radianos
math.tan(x)	Retorna tangente de x em radianos
math.degrees(r)	Conver o float r em radianos para graus
math.ceil(x)	Retorna o menor inteiro maior ou igual a x
math.floor(x)	Retorna o maior inteiro menor ou igual a x

MÓDULO - MATH



```

6 import math
7
8 rad = math.pi/4;          #MEDIDA EM RADIANOS
9 graus = math.degrees(rad) #MEDIDA EM GRAUS
10 f = math.e
11 x = 9
12 y = 3
13
14 print( rad,graus )
15 print( math.cos(rad) )
16 print( math.sin(rad) )
17 print( math.tan(rad) )
18 print( math.acos(rad) )
19 print( math.asin(rad) )
20 print( math.atan(rad) )
21 print( math.cosh(rad) )
22 print( math.sinh(rad) )
23 print( math.tanh(rad) )
24 print( f )
25 print( math.ceil(f) )
26 print( math.floor(f) )
27 print( math.exp(x) )
28 print( math.log(x,y) )
29 print( math.log10(y) )
30 print( math.modf(f) )
31 print( math.fmod(x,y) )

```

FUNÇÕES	Descrição
math.exp(x)	Retorna e^x
math.log(x,b)	Retorna $\log_b x$
math.log10(x)	Retorna $\log_{10} x$
math.modf(x)	Retorna o fracional de x e parte inteira como dois floats
math.fmod(x,y)	Produz o resto da divisão de x por y
math.trunc(x)	Retorna a parte inteira de x como um int
math.fsum(i)	Retorna a soma dos valores num iterável i como um float
math.sum(i)	Retorna a soma de pontos flutuantes do itens iteráveis do i
math.pi	Retorna o valor da constante pi
math.e	Retorna o valor da constante de Euler e
math.pow(x,y)	Retorna a potência x^y
math.sqrt(x)	Retorna a raiz quadrada de x

TIPOS NUMÉRICOS - INT, FLOAT E COMPLEX



BOOLEAN - bool

```

1 ⊕ """
2 LAB03 - OPERADOR BINARIO
3
4 @author: DANIEL CORREIA DA SILVA
5 """
6 x = 0b111000
7 y = 0b111111
8
9 print( bin(x|y) )
10 print( bin(x^y) )
11 print( bin(x&y) )
12 print( bin(~x) )
13 print( bin(x<<3) )
14 print( bin(x>>4) )

```

OPERADORES LÓGICOS

Operadores	Descrição
x y	Operador lógico OR
x ^ y	Operador lógico XOR
x & y	Operador lógico AND
~ x	Operador lógico NOT
x >> y	Deslocamento à direita dos bits de x por y vezes
x << y	Deslocamento à esquerda dos bits de x por y vezes

```

>>> True
True
>>> False
False

```

```

>>> bool(0)
False
>>> bool(42)
True
>>> bool(-1)
True

```

TIPOS NUMÉRICOS - INT, FLOAT E COMPLEX



BOOLEAN - bool

```
>>> g = 20
```

```
>>> g == 20
True
>>> g == 13
False
```

```
>>> g != 20
False
>>> g != 13
True
```

```
>>> g < 30
True
```

```
>>> g > 30
False
```

```
>>> g <= 20
True
```

```
>>> g >= 20
True
```

OPERADORES RELACIONAIS

Operadores	Descrição
<code>x == y</code>	Igual
<code>x != y</code>	Diferente
<code>x > y</code>	Maior que
<code>x < Y</code>	Menor que
<code>x >= y</code>	Maior ou igual
<code>x <= y</code>	Menor ou igual

- ▶ TIPO TEXTO
 - ▶ CARACTERES ESPECIAIS
 - ▶ MÉTODOS PARA MANIPULAÇÃO DE STRINGS
-

TIPOS E ESTRUTURAS DE DADOS

TIPO TEXTO - STR



STRINGS - " " ou ''

- String de aspas simples:

```
>>> 'This is a string'
```

- String de aspas dupla:

```
>>> "This is also a string"
```

- Combinando às aspas simples e duplas:

```
>>> "It's a good thing."  
"It's a good thing."  
>>> '"Yes!"', he said, "I agree!"'  
'"Yes!", he said, "I agree!"'
```

- Concatenação strings adjacentes

```
>>> "first" "second"  
'firstsecond'
```

- Múltiplas strings e novas linhas

```
>>> """This is  
... a multiline  
... string"""  
'This is\na multiline\nstring'
```

```
>>> '''So  
... is  
... this.''  
'So\nis\nthis.'
```

TIPO TEXTO – STR



- Caracter escape \n:

```
>>> '''So
... is
... this.'''
'So\nis\nthis.'
```

- Saltar de linhas com **nspam** e **nlines**:

```
>>> m = 'This string\nspans multiple\nlines'
>>> m
'This string\nspans multiple\nlines'
```

```
>>> print(m)
This string
spans multiple
lines
```

- Strings brutas:

```
>>> path = r'C:\Users\Merlin\Documents\Spells'
>>>
>>> path
'C:\\Users\\Merlin\\Documents\\Spells'
>>> print(path)
C:\Users\Merlin\Documents\Spells
```

TIPO BYTE - BYTE



METODOS PARA MANIPULAÇÃO DE STRING

- Função **capitalize**:

```
>>> c = "oslo"
```

```
>>> c.capitalize()
'Oslo'
```

- String com Unicode:

```
>>> '\xe5'
'â'
```

```
>>> '\345'
'â'
```

TIPO BYTE - BYTE



METODOS PARA MANIPULAÇÃO DE STRING

Sintaxe	Descrição
<code>s.capitalize()</code>	Retorna uma cópia de str s com a primeira letra maiúscula; ver também o método <code>str.title()</code> .
<code>s.center(width, char)</code>	Retorna uma cópia de um s centralizado numa string de comprimento width preenchida com espaços, ou, opcionalmente, com char (uma string de comprimento 1); ver <code>str.ljust()</code> , <code>str.rjust()</code> e <code>str.format()</code> .
<code>s.count(t, start, end)</code>	Retorna o número de ocorrências de str t em str s (ou na fatia <code>start:end</code> de s).
<code>s.encode(encoding, err)</code>	Retorna um objeto bytes, que representa a string usando a codificação padrão ou usando a codificação (<code>encoding</code>) especificada e manipulação de erros, de acordo com os argumento opcional <code>err</code> .
<code>s.endswith(x, start, end)</code>	Retorna True se s (ou a fatia <code>start:end</code> de s) termina com str x, ou com qualquer uma das string na tuple x; de outra maneira, retorna False. Ver também <code>str.startswith()</code> .
<code>s.expandtabs(size)</code>	Retorna uma cópia de s com tabulação substituída por espaços em múltiplos de 8 ou, então, o <code>size</code> se especificado.
<code>s.find(t, start, end)</code>	Retorna a posição esquerda de t em s (ou na fatia <code>start:end</code> de s) ou -1 se não é encontrado. Use <code>str.rfind()</code> para achar a posição mais à direita. Ver também <code>str.index()</code> .

<code>s.format(...)</code>	Retorna uma cópia de s formatada de acordo com o argumento dado. Este método e seus argumentos são abordados na próxima seção.
<code>s.index(t, start, end)</code>	Retorna a posição mais à esquerda de t em s (ou na fatia <code>start:end</code> de s) ou lança <code>ValueError</code> se nada for encontrado. Use <code>str.rindex()</code> para procurar a partir da direita. Ver <code>str.find()</code> .
<code>s.isalnum()</code>	Retorna True se s não for vazia e cada caractere de s é alfanumérico.
<code>s.isalpha()</code>	Retorna True se s não for vazia e cada caractere de s é alfabetico.
<code>s.isdecimal()</code>	Retorna True se s não for vazia e cada caractere de s é um dígito Unicode de base 10.
<code>s.isdigit()</code>	Retorna True se s não for vazia e cada caractere de s é um dígito ASCII.
<code>s.isidentifier()</code>	Retorna True se s não é vazio e um identificador válido.
<code>s.islower()</code>	Retorna True se s possuir pelo menos um caractere em letra minúscula, e se todos os seus caracteres letra minúscula estiverem em minúscula; ver também <code>str.isupper()</code> .



METODOS PARA MANIPULAÇÃO DE STRING

Sintaxe	Descrição
<code>s.isnumeric()</code>	Retorna True se s não é vazia e cada caractere em s for um caractere numérico do Unicode, assim como um dígito ou fração.
<code>s.isprintable()</code>	Retorna True se s é vazia ou se todo caractere em s é considerado imprimível, incluindo espaços; exceto nova linha.
<code>s.isspace()</code>	Retorna True se s não é vazia e se cada caractere em s é um caractere espaço em branco.
<code>s.istitle()</code>	Retorna True se s não é string vazia e tem as primeiras letras de cada palavra maiúscula (título); ver também str.title().
<code>s.isupper()</code>	Retorna True se str s possui ao menos um caractere passível de estar em maiúsculo e se todos os seus caracteres passíveis de serem maiúsculos estejam em maiúsculo; ver também str.islower().
<code>s.join(seq)</code>	Retorna a concatenação de cada item na sequência seq, com str s (o qual pode estar vazio) entre cada uma.

<code>s.ljust(width, char)</code>	Retorna uma cópia do s alinhado à esquerda em uma string de comprimento width preenchida com espaços ou, opcionalmente, com char (uma string de comprimento 1). Use str.rjust() para alinhar à direita e str.center() para centralizar. Ver também str.format().
<code>s.lower()</code>	Retorna uma cópia em minúscula de s; ver também str.upper().
<code>s.maketrans()</code>	Companheiro de str.translate(); ver o texto para mais detalhes.
<code>s.partition(t)</code>	Retorna uma tupla de três string – parte de str s antes e mais à esquerda de str t, t e a parte de s depois de t; ou se t não está em s retorna s e duas strings vazias. Use str.rpartition() para distribuir ocorrência mais à direita de t.
<code>s.replace(t, u, n)</code>	Retorna uma cópia de s com cada (ou o máximo de n, se este for dado) ocorrência de str t substituído com str u.
<code>s.split(t, n)</code>	Retorna uma lista de strings fragmentadas, no máximo, n vezes no str t; se n não for dado, ele irá fragmentar quantas vezes for possível; se t não for dado, ele fragmenta os espaços em branco. Use str.rsplit() para fragmentação iniciando da direita – isto faz diferença apenas se n for dado e se for menor do que o número máximo de fragmentações possíveis.
<code>s.splitlines(f)</code>	Retorna a lista de linhas produzidas pela fragmentação de s nos finalizadores de linhas, tirando os finalizadores, a menos que f seja True.
<code>s.startswith(x, start, end)</code>	Retorna True se s (ou a fatia start:end de s) iniciar com str x ou qualquer outra das strings na tupla x; de outra maneira, retorna False. Ver também str.endswith().

► TIPO BYTE

TIPOS E ESTRUTURAS DE DADOS

TIPO BYTE - BYTE



Byte: diferente de string, um string de bytes é uma sequência de caracteres Unicode.

- Bytes:

```
>>> b'data'  
b'data'  
>>> b"data"  
b'data'
```

```
>>> d = b'some bytes'  
>>> d.split()  
[b'some', b'bytes']
```

- Convertendo **byte** em **str** e vice-versa:

```
>>> norsk = "Jeg begynte å fortære en sandwich mens jeg kjørte taxi på vei til quiz"  
  
>>> data = norsk.encode('utf-8')  
>>> data  
b'Jeg begynte \xc3\xaa fort\xc3\xabre en sandwich mens jeg kj\xc3\xb8rte taxi p\xc3\xaa vei til quiz'
```

```
>>> norwegian = data.decode('utf-8')
```

► TIPO LISTA

TIPOS E ESTRUTURAS DE DADOS

TIPO LISTA - LIST



List: uma sequência de objetos.

- Lista de objetos inteiros e strings:

```
>>> [1, 9, 8]  
[1, 9, 8]
```

```
>>> a = ["apple", "orange", "pear"]
```

- Acessando e alterando os elementos da lista

```
>>> a[1]  
"orange"
```

```
>>> a[1] = 7  
>>> a  
['apple', 7, 'pear']
```

- Lista vazia:

```
>>> b = []
```

TIPO LISTA - LIST



List: uma sequência de objetos.

- **append():** adiciona ao final da lista:

```
>>> b.append(1.618)
>>> b
[1.618]
>>> b.append(1.414)
[1.618, 1.414]
```

- **list():**Transforma uma string em lista de caracteres:

```
>>> list("characters")
['c', 'h', 'a', 'r', 'a', 'c', 't', 'e', 'r', 's']
```

- Lista de strings:

```
>>> c = ['bear',
...       'giraffe',
...       'elephant',
...      'caterpillar',]
>>> c
['bear', 'giraffe', 'elephant', 'caterpillar']
```

► TIPO DICIONÁRIO

TIPOS E ESTRUTURAS DE DADOS

TIPO DICIONÁRIO - DICT



DICT: mapeia Key em valores.

- Dicionário:

```
>>> d = {'alice': '878-8728-922', 'bob': '256-5262-124',  
        'eve': '198-2321-787'}
```

- Localizando um elemento:

```
>>> d['alice']  
'878-8728-922'
```

- Alterando um elemento do dicionário:

```
>>> d['alice'] = '966-4532-6272'  
>>> d  
{'bob': '256-5262-124', 'eve': '198-2321-787',  
   'alice': '966-4532-6272'}
```

- Dicionário vazio:

```
>>> e = {}
```



ESTRUTURAS DE CONTROLE

Controlando o fluxo com Python

ESTRUTURAS DE CONTROLE



ESTRUTURA CONDICIONAL

- Estrutura condicional simples:

```
>>> if False:  
...     print("It's true!")  
...  
>>>
```

```
>>> if bool("eggs"):  
...     print("Yes please!")  
...  
Yes please!
```

- Estrutura condicional composta:

```
>>> if h > 50:  
...     print("Greater than 50")  
... else:  
...     if h < 20:  
...         print("Less than 20")  
...     else:  
...         print("Between 20 and 50")  
...  
Between 20 and 50
```

```
>>> if h > 50:  
...     print("Greater than 50")  
... elif h < 20:  
...     print("Less than 20")  
... else:  
...     print("Between 20 and 50")  
...  
Between 20 and 50
```

ESTRUTURAS DE CONTROLE



ESTRUTURA REPETIÇÃO - WHILE

- Estrutura de repetição **while**:

```
>>> c = 5
>>> while c != 0:
...     print(c)
...     c -= 1
...
5
4
3
2
1
```

```
>>> c = 5
>>> while c:
...     print(c)
...     c -= 1
...
5
4
3
2
1
```

- Loop infinito:

```
>>> while True:
...     print("Looping!")
...
Looping!
Looping!
```

- Interrompendo com **break**:

```
>>> while True:
...     response = input()
...     if int(response) % 7 == 0:
...         break
...
```

ESTRUTURAS DE CONTROLE



ESTRUTURA REPETIÇÃO - FOR

- Estrutura de repetição **for**: iterando lista de objetos:

```
>>> cities = ["London", "New York", "Paris", "Oslo", "Helsinki"]
>>> for city in cities:
...     print(city)
...
London
New York
Paris
Oslo
Helsinki
```

- Iterando um dicionário:

```
>>> colors = {'crimson': 0xdc143c, 'coral': 0xff7f50,
...             'teal': 0x008080}
>>> for color in colors:
...     print(color, colors[color])
...
coral 16744272
crimson 14423100
teal 32896
```

- Interrompendo com **break**:



MODULARIDADE

Modularizando com Python

MODULARIDADE



FUNÇÕES- def

- Definindo uma função:

```
>>> def square(x):
...     return x * x
...
...
```

- Função para informar se o número é ímpar/par:

```
>>> def even_or_odd(n):
...     if n % 2 == 0:
...         print("even")
...         return
...     print("odd")
...
>>> even_or_odd(4)
even
>>> even_or_odd(5)
odd
```

```
>>> w = even_or_odd(31)
odd
>>> w is None
True
```

MODULARIDADE



Módulos: são arquivos-fontes que podem ser importados para um programa

Os módulos são localizados pelo interpretador por meio da lista de pastas PYTHONPATH (sys.path) que normalmente inclui a pasta corrente em primeiro lugar.

Pacotes: os pacotes são como coleções para organizar módulos de forma hirárquica. Essas pastas são identificadas pela interpretador pela presença de um arquivo com o nome `__init__.py`



MÓDULO

- Módulo: arquivo Python com o nome **words.py**

```
from urllib.request import urlopen

def fetch_words():
    with urlopen('http://sixty-north.com/c/t.txt') as story:
        story_words = []
        for line in story:
            line_words = line.decode('utf-8').split()
            for word in line_words:
                story_words.append(word)

    for word in story_words:
        print(word)
```

MODULARIDADE



MÓDULO

- Importando o módulo **words.py**

```
$ python3
Python 3.5.0 (default, Nov  3 2015, 13:17:02)
[GCC 4.2.1 Compatible Apple LLVM 6.1.0 (clang-602.0.53)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import words
```

- Acessando a função **fetch_words()** do módulo **words.py**

```
>>> words.fetch_words()
It
was
the
best
of
times
```