

Introdução a Python

Algumas coisas para aprender
antes de Machine Learning

Giana de Almeida

Workshop básico adaptado para dar tempo



Talita | ADS



Giana | SI

[Para treinar em casa, clique aqui e acesse o original](#)

O que é algoritmo

Um algoritmo é uma sequência de passos.

Por exemplo:

- abrir a geladeira
- pegar o ônibus
- levantar da cama
- ler esse slide até o fim

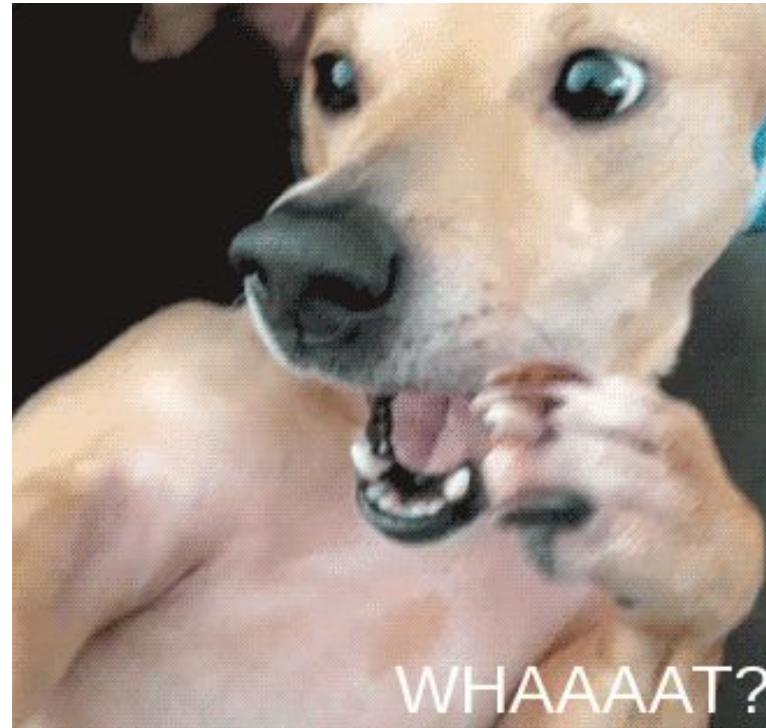
Explique os passos que eu devo realizar para ir até o fim da sala pegar um copo d'água

Obs: estou aproveitando esse momento para beber água :)

Mas o que isso tem a ver com programação?

Se prepare para a polêmica....

Computadores são burros



Mas como assim??

Os computadores não sabem o que devem fazer.

Precisamos **pensar pelo computador** e dizer os passos que queremos que ele dê para fazer o que queremos que ele faça.

Basicamente, temos que ensinar ele.

Mas como ensinamos os computadores?

Obviamente, nós e o computador, não falamos a mesma linguagem, então precisamos de um tradutor: uma linguagem de programação.

E para nos comunicarmos com a linguagem, utilizamos algoritmos.

Estes algoritmos podem ser executados por um humano ou um computador. Logo, programação é a arte de fazer com que o computador execute uma **sequência de instruções**.

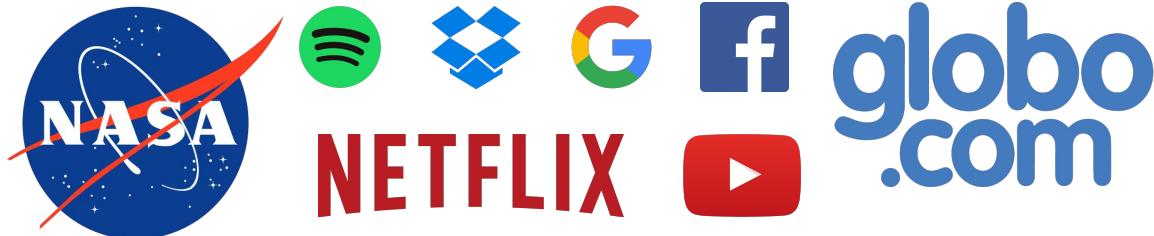
Mas por que Python?

Senta que lá vem textão...



Python é uma linguagem de programação com código aberto, de alto nível (mais fácil para a pessoa que está desenvolvendo entender).

Utilizada para fazer - quase - tudo o que vocês pensarem: aplicações web, scripts para administrar sistemas, buscar informações da internet automaticamente, analisar dados, entre outras coisas que vocês irão aprender com o tempo.



Além de ter uma comunidade linda <3

A COMUNIDADE



E acima de tudo...

A simplicidade comparado à outras linguagens

Java

```
1 public class Hello
2 {
3     public static void main(String args[]) {
4         java.util.Scanner s = new java.util.Scanner(System.in);
5         System.out.print("Digite seu nome:");
6         String nome = s.nextLine();
7         System.out.println("Olá, " + nome);
8     }
9 }
```

A simplicidade comparado à outras linguagens

Java

```
1 public class Hello
2 {
3     public static void main(String args[]) {
4         java.util.Scanner s = new java.util.Scanner(System.in);
5         System.out.print("Digite seu nome:");
6         String nome = s.nextLine();
7         System.out.println("Olá, " + nome);
8     }
9 }
```

C

```
1 #include <stdio.h>
2 int main()
3 {
4     char nome[200];
5     printf("Digite seu nome:");
6     scanf("%s", nome);
7     printf("Olá, %s\n", nome);
8     return 0;
9 }
```

A simplicidade comparado à outras linguagens

Java

```
1 public class Hello
2 {
3     public static void main(String args[]) {
4         java.util.Scanner s = new java.util.Scanner(System.in);
5         System.out.print("Digite seu nome:");
6         String nome = s.nextLine();
7         System.out.println("Olá, " + nome);
8     }
9 }
```

C

```
1 #include <stdio.h>
2 int main()
3 {
4     char nome[200];
5     printf("Digite seu nome:");
6     scanf("%s", nome);
7     printf("Olá, %s\n", nome);
8 }
9 }
```

Pascal

```
1 program HelloWorld(output);
2 var
3     nome: string;
4 begin
5     writeln('Digite seu nome:');
6     read(nome);
7     writeln('Olá, ', nome);
8 end.
```

A simplicidade comparado à outras linguagens

Java

```
1 public class Hello
2 {
3     public static void main(String args[]) {
4         java.util.Scanner s = new java.util.Scanner(System.in);
5         System.out.print("Digite seu nome:");
6         String nome = s.nextLine();
7         System.out.println("Olá, " + nome);
8     }
9 }
```

C

```
1 #include <stdio.h>
2 int main()
3 {
4     char nome[200];
5     printf("Digite seu nome:");
6     scanf("%s", nome);
7     printf("Olá, %s\n", nome);
8     return 0;
9 }
```

Pascal

```
1 program HelloWorld(output);
2 var
3     nome: string;
4 begin
5     writeln('Digite seu nome:');
6     read(nome);
7     writeln('Olá, ', nome);
8 end.
```

Python

```
1 nome = input('Digite seu nome:')
2 print ('Olá,', nome)
```



Ok... Esse tal Python parece
ótimo, bora instalar?

A instalação...

Os slides terão passo a passo via internet, porém sabemos que a internet nem sempre colabora, então passaremos pen drives com o instalador



Verifique se já está instalado:

Instalando o Python no Linux

Verifique se já tem o Python instalado, se você usa GNU/Linux, provavelmente já possui alguma versão do Python instalada por padrão. Para conferir, digite em um terminal:

```
$ which python
```

ou

```
$ which python3
```

que deve retornar algo como `/usr/bin/python`. Isso significa que o Python está instalado nesse endereço.

Caso contrário, se retornar algo como `which: no python in (/usr/local/sbin:/usr/local/bin:/usr/bin:/usr....)` você precisa instalar pelos repositórios ou gerenciador de pacotes de sua distribuição.



Instalando...

Instalação por Gerenciadores de Pacotes

Os gerenciadores de pacotes mais comuns são apt-get (Debian, Ubuntu) e yum (RedHat, CentOS). Caso sua distribuição utilize um gerenciador de pacotes diferente, acesse a página de downloads do Python.

Apt-get

Para instalar o Python 2.7, digite em um terminal:

```
$ sudo apt-get install python2.7
```

Para instalar o Python 3.5, digite em um terminal:

```
$ sudo apt-get install python3.5
```

(Opcional) Para instalar o gerenciador de pacotes pip, digite em um terminal:

```
$ sudo apt-get install python-pip
```



Caso esteja usando o Yum

Yum

Para instalar o Python 2.7, digite em um terminal:

```
$ sudo yum install python27
```

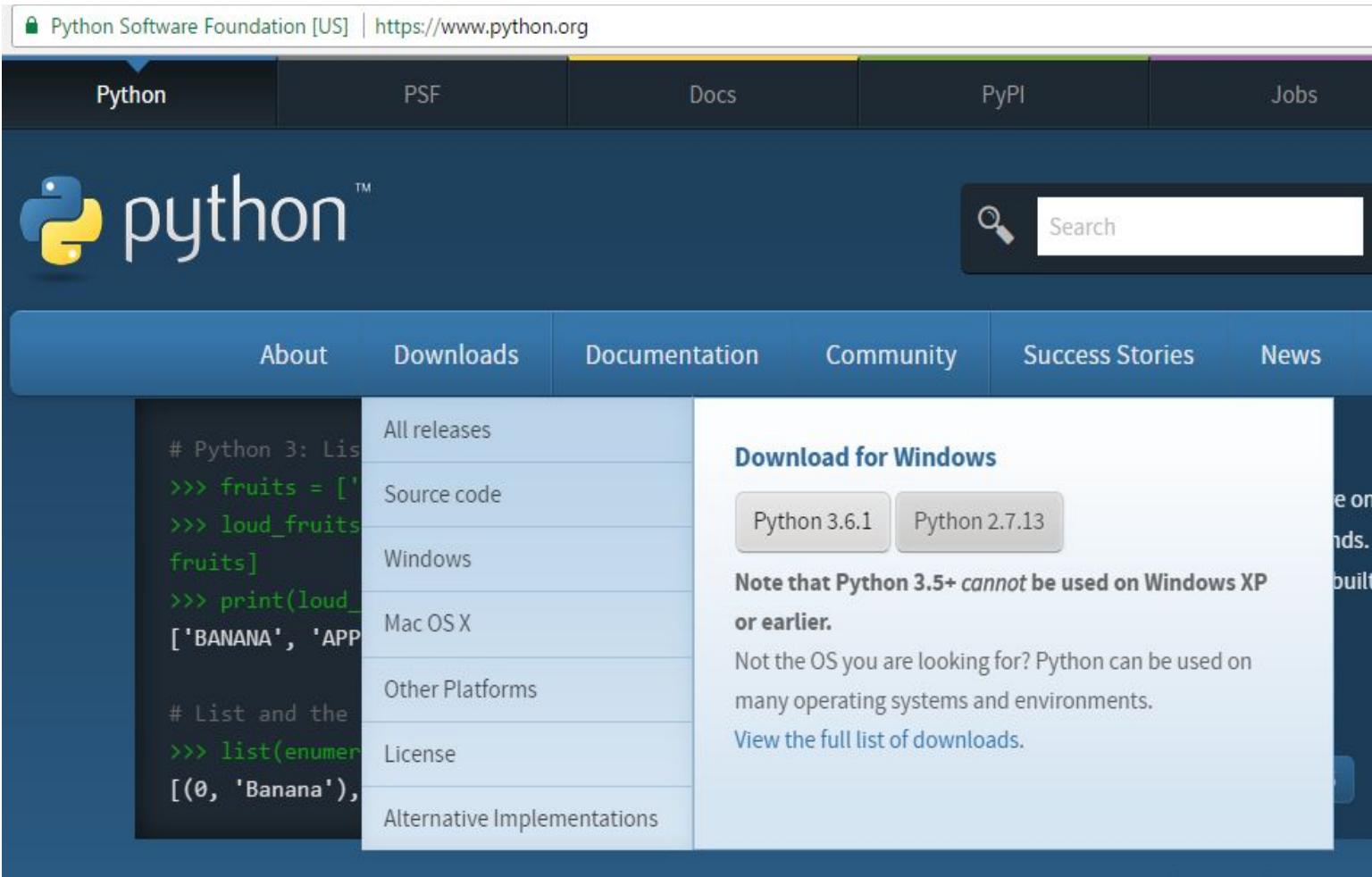
Para instalar o Python 3.5, digite em um terminal:

```
$ sudo yum install python35
```

(Opcional) Para instalar o gerenciador de pacotes pip, digite em um terminal:

```
$ yum -y install python-pip
```

Caso você use outros sistemas operacionais, primeiro baixe o instalador



The screenshot shows the Python Software Foundation website's 'Downloads' page. At the top, there is a navigation bar with links for Python, PSF, Docs, PyPI, and Jobs. Below the navigation bar is the Python logo and a search bar. The main content area has a blue header with tabs for About, Downloads, Documentation, Community, Success Stories, and News. The 'Downloads' tab is active. On the left, there is a sidebar with a code snippet:

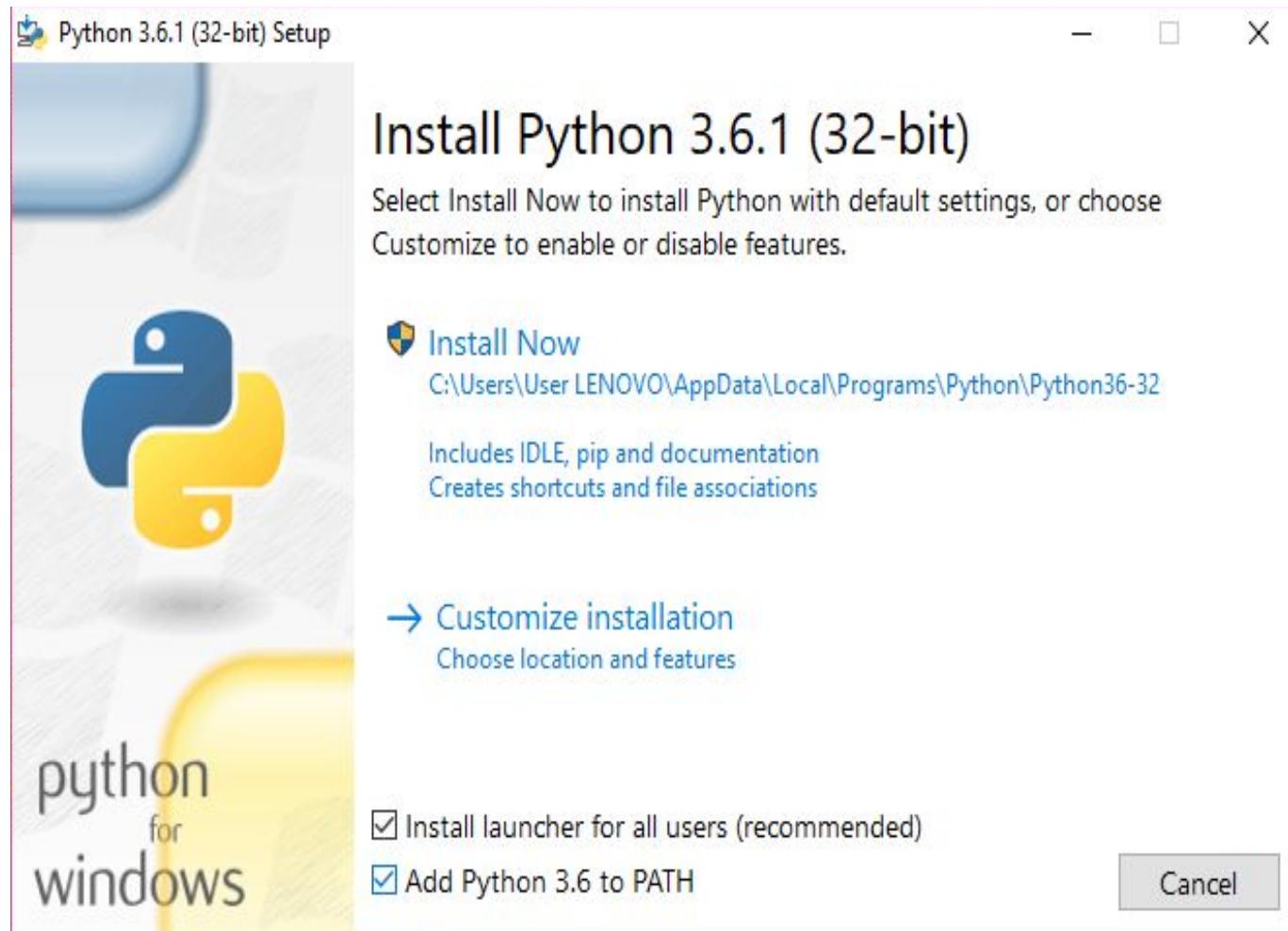
```
# Python 3: List comprehension
>>> fruits = [
    >>>     loud_fruit
    >>>     fruit
]
>>> print(loud_
    ['BANANA', 'APP'])

# List and the
>>> list(enumerate(
    [(0, 'Banana'),
```

The sidebar also lists links for All releases, Source code, Windows, Mac OS X, Other Platforms, License, and Alternative Implementations. To the right of the sidebar, under the 'Downloads' tab, is a section titled 'Download for Windows' with buttons for Python 3.6.1 and Python 2.7.13. A note states: 'Note that Python 3.5+ cannot be used on Windows XP or earlier.' It also mentions that Python can be used on many operating systems and environments, with a link to 'View the full list of downloads.'

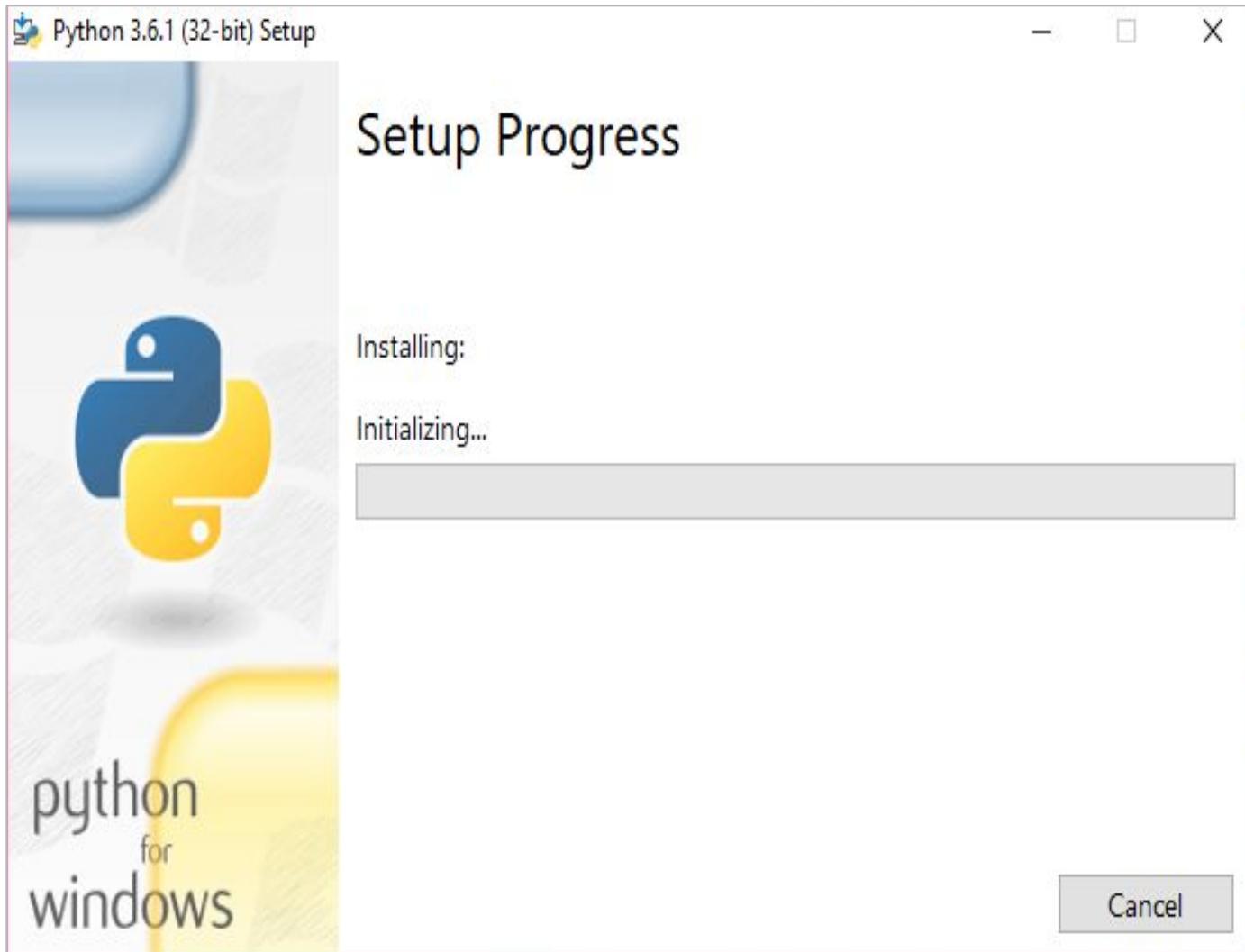


Segundo: Clique duas vezes no instalador, selecione "add Python 3.6 to PATH" e clique em "Instal Now"

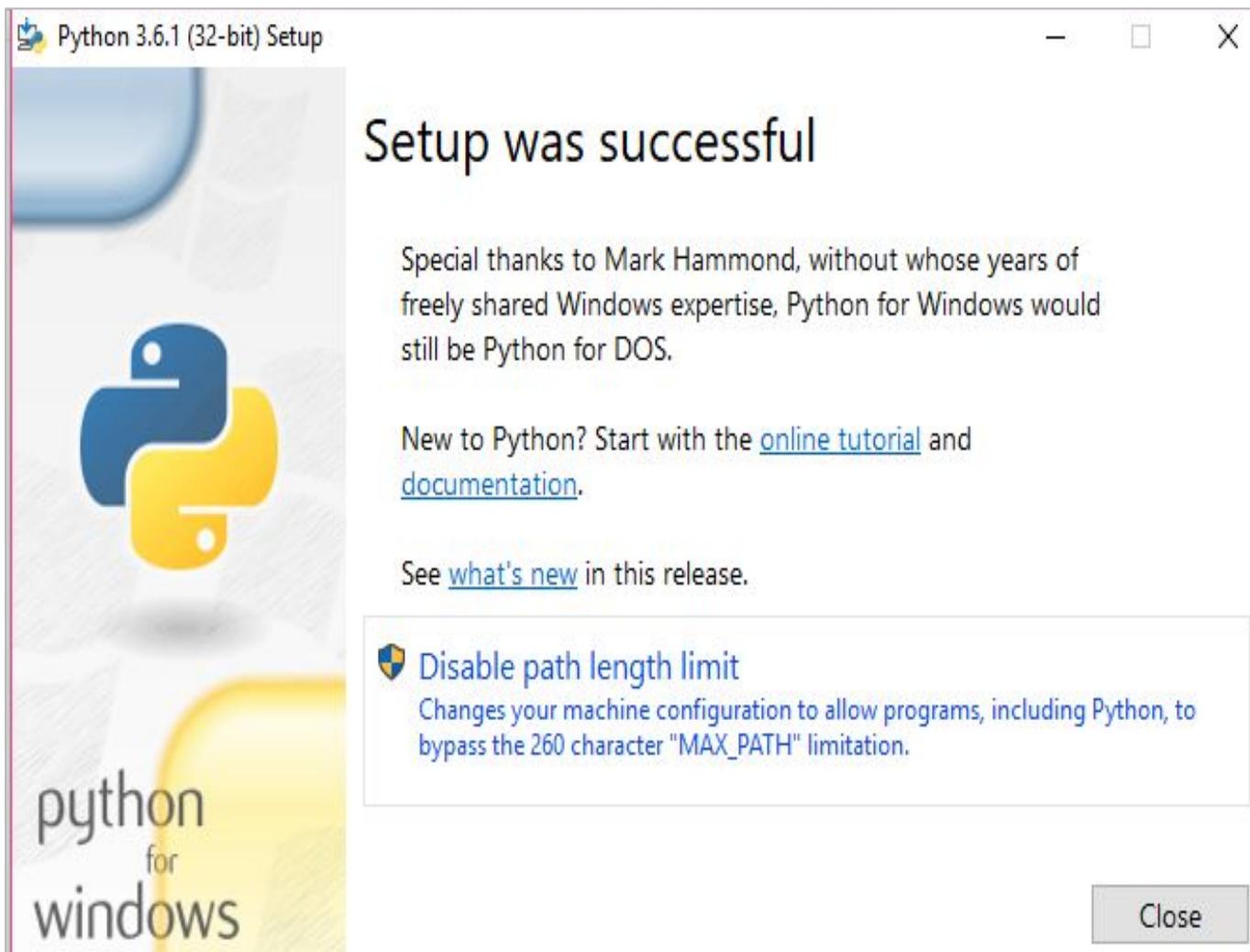




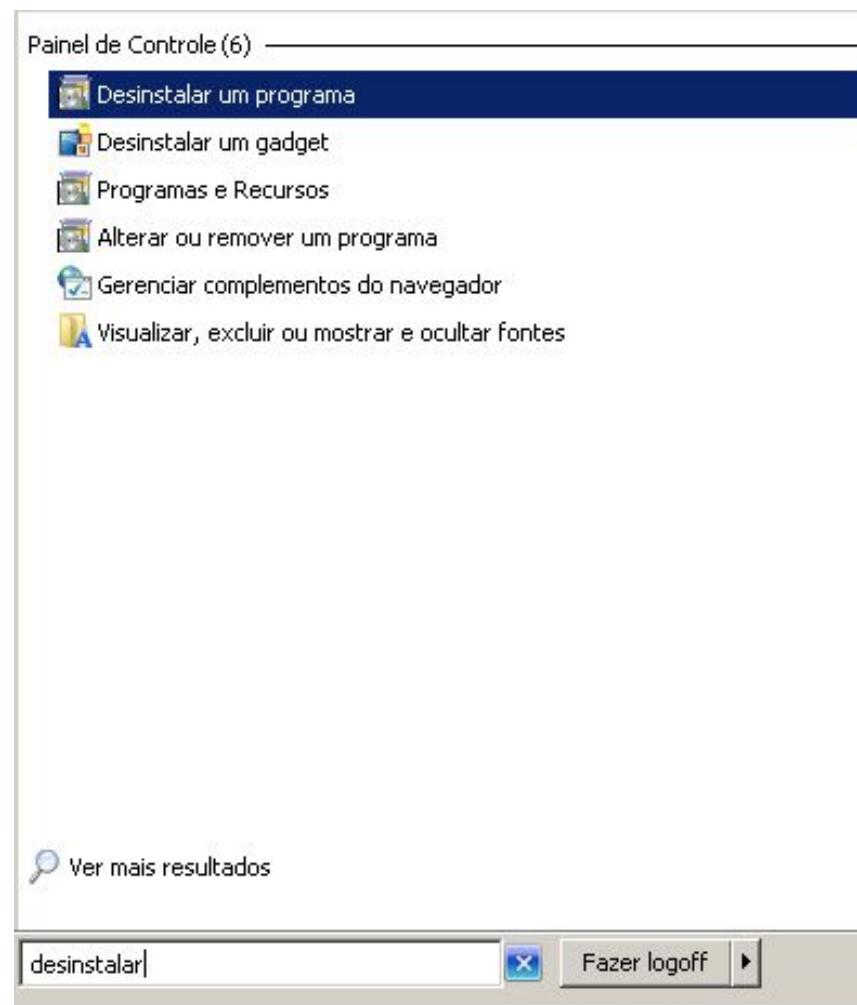
Agora, espere...



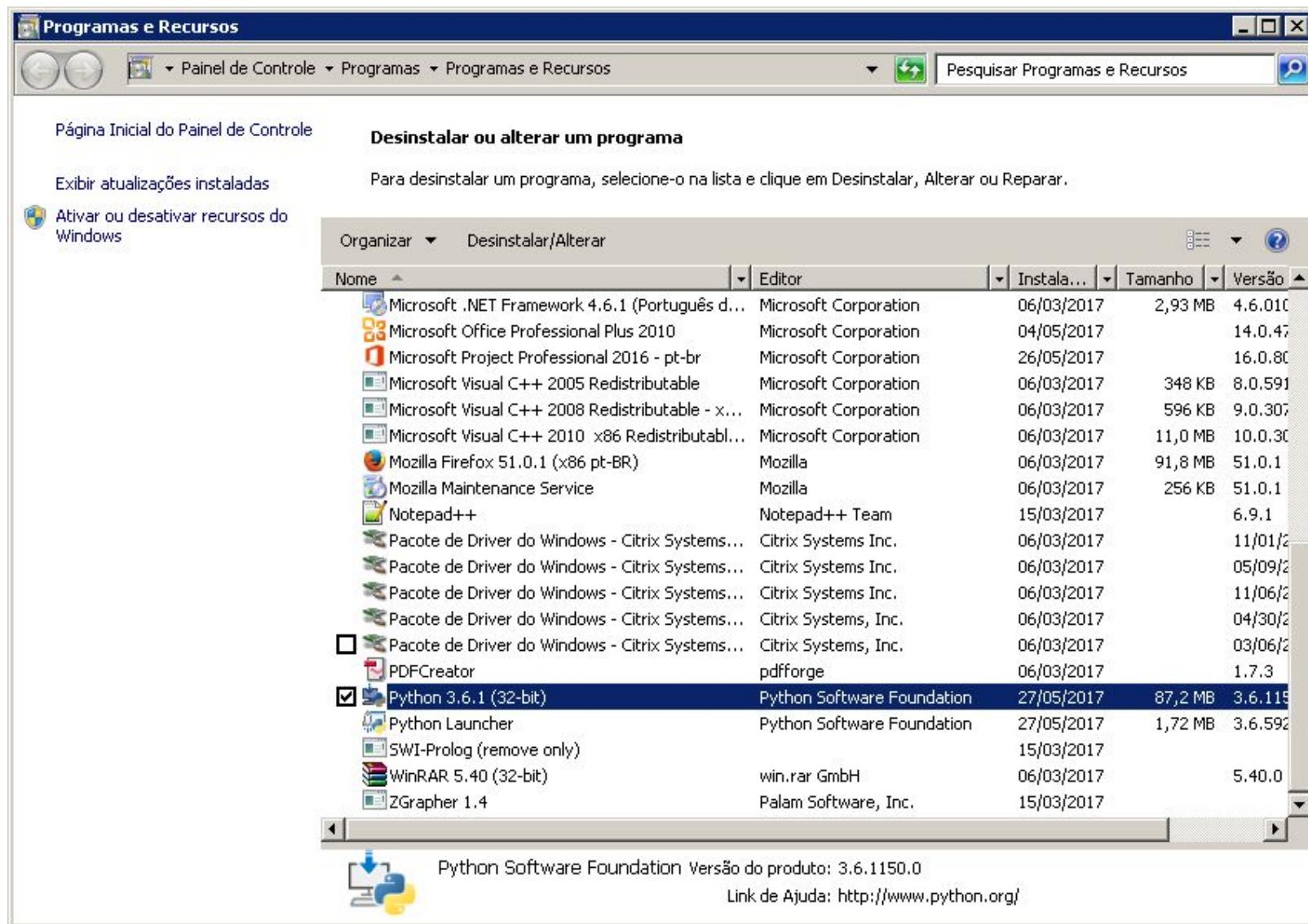
Pode rolar isso... Mas é só clicar no Disable path



PARA FUNCIONAR O PIP



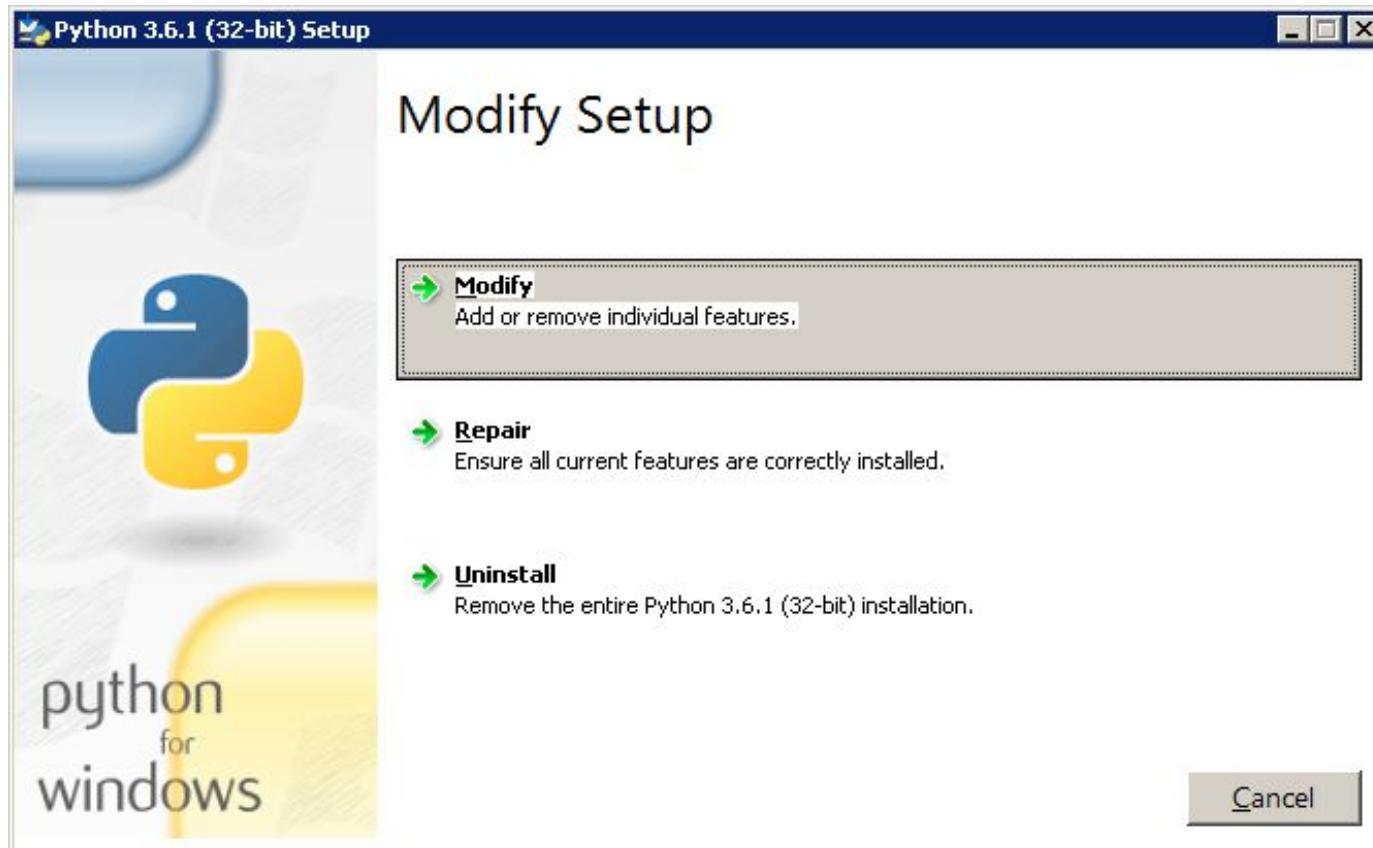
PARA FUNCIONAR O PIP



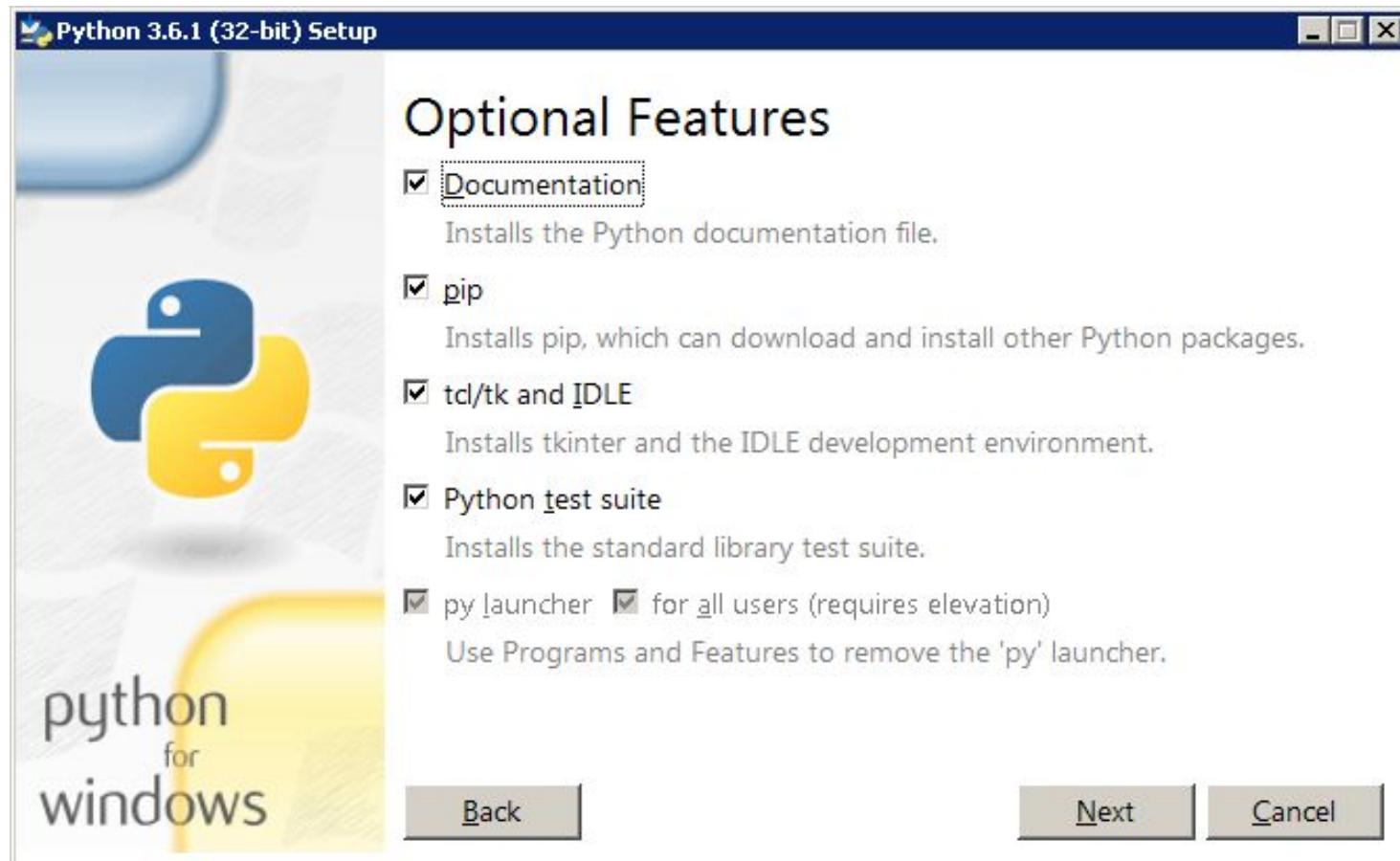
The screenshot shows the Windows Control Panel with the title "Programas e Recursos". The left sidebar has links for "Página Inicial do Painel de Controle", "Exibir atualizações instaladas", and "Ativar ou desativar recursos do Windows". The main area is titled "Desinstalar ou alterar um programa" with the sub-instruction "Para desinstalar um programa, selecione-o na lista e clique em Desinstalar, Alterar ou Reparar.". A table lists installed programs, with "Python 3.6.1 (32-bit)" selected and highlighted in blue. The Python logo is visible at the bottom left, along with the text "Python Software Foundation Versão do produto: 3.6.1150.0" and "Link de Ajuda: <http://www.python.org/>".

Nome	Editor	Instala...	Tamanho	Versão
Microsoft .NET Framework 4.6.1 (Português d...	Microsoft Corporation	06/03/2017	2,93 MB	4.6.010
Microsoft Office Professional Plus 2010	Microsoft Corporation	04/05/2017		14.0.47
Microsoft Project Professional 2016 - pt-br	Microsoft Corporation	26/05/2017		16.0.80
Microsoft Visual C++ 2005 Redistributable	Microsoft Corporation	06/03/2017	348 KB	8.0.591
Microsoft Visual C++ 2008 Redistributable - x...	Microsoft Corporation	06/03/2017	596 KB	9.0.307
Microsoft Visual C++ 2010 x86 Redistributabl...	Microsoft Corporation	06/03/2017	11,0 MB	10.0.30
Mozilla Firefox 51.0.1 (x86 pt-BR)	Mozilla	06/03/2017	91,8 MB	51.0.1
Mozilla Maintenance Service	Mozilla	06/03/2017	256 KB	51.0.1
Notepad++	Notepad++ Team	15/03/2017		6.9.1
Pacote de Driver do Windows - Citrix Systems...	Citrix Systems Inc.	06/03/2017		11/01/2
Pacote de Driver do Windows - Citrix Systems...	Citrix Systems Inc.	06/03/2017		05/09/2
Pacote de Driver do Windows - Citrix Systems...	Citrix Systems Inc.	06/03/2017		11/06/2
Pacote de Driver do Windows - Citrix Systems...	Citrix Systems, Inc.	06/03/2017		04/30/2
Pacote de Driver do Windows - Citrix Systems...	Citrix Systems, Inc.	06/03/2017		03/06/2
PDFCreator	pdfforge	06/03/2017		1.7.3
<input checked="" type="checkbox"/> Python 3.6.1 (32-bit)	Python Software Foundation	27/05/2017	87,2 MB	3.6.1150
Python Launcher	Python Software Foundation	27/05/2017	1,72 MB	3.6.592
SWI-Prolog (remove only)		15/03/2017		
WinRAR 5.40 (32-bit)	win.rar GmbH	06/03/2017		5.40.0
ZGrapher 1.4	Palam Software, Inc.	15/03/2017		

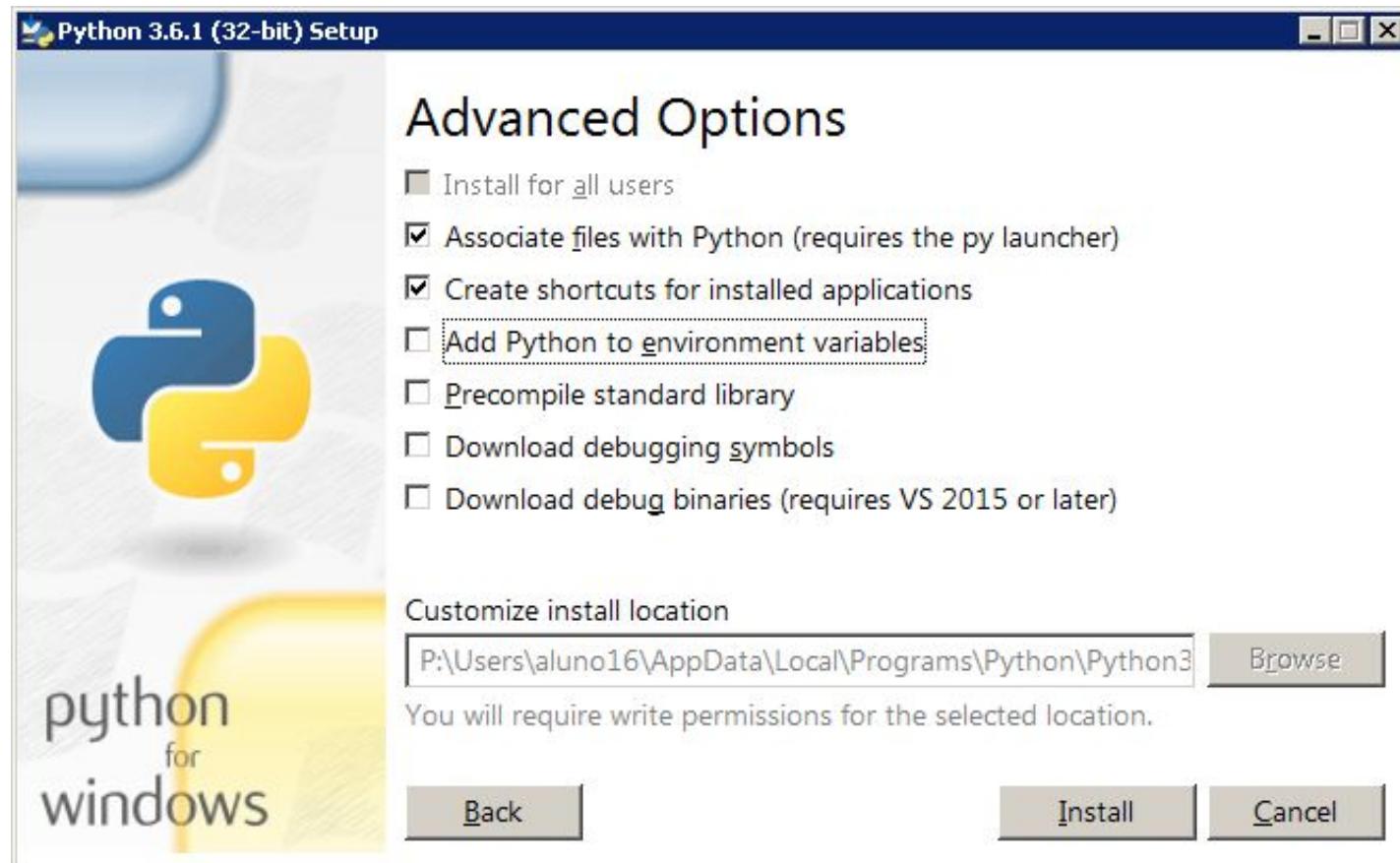
PARA FUNCIONAR O PIP



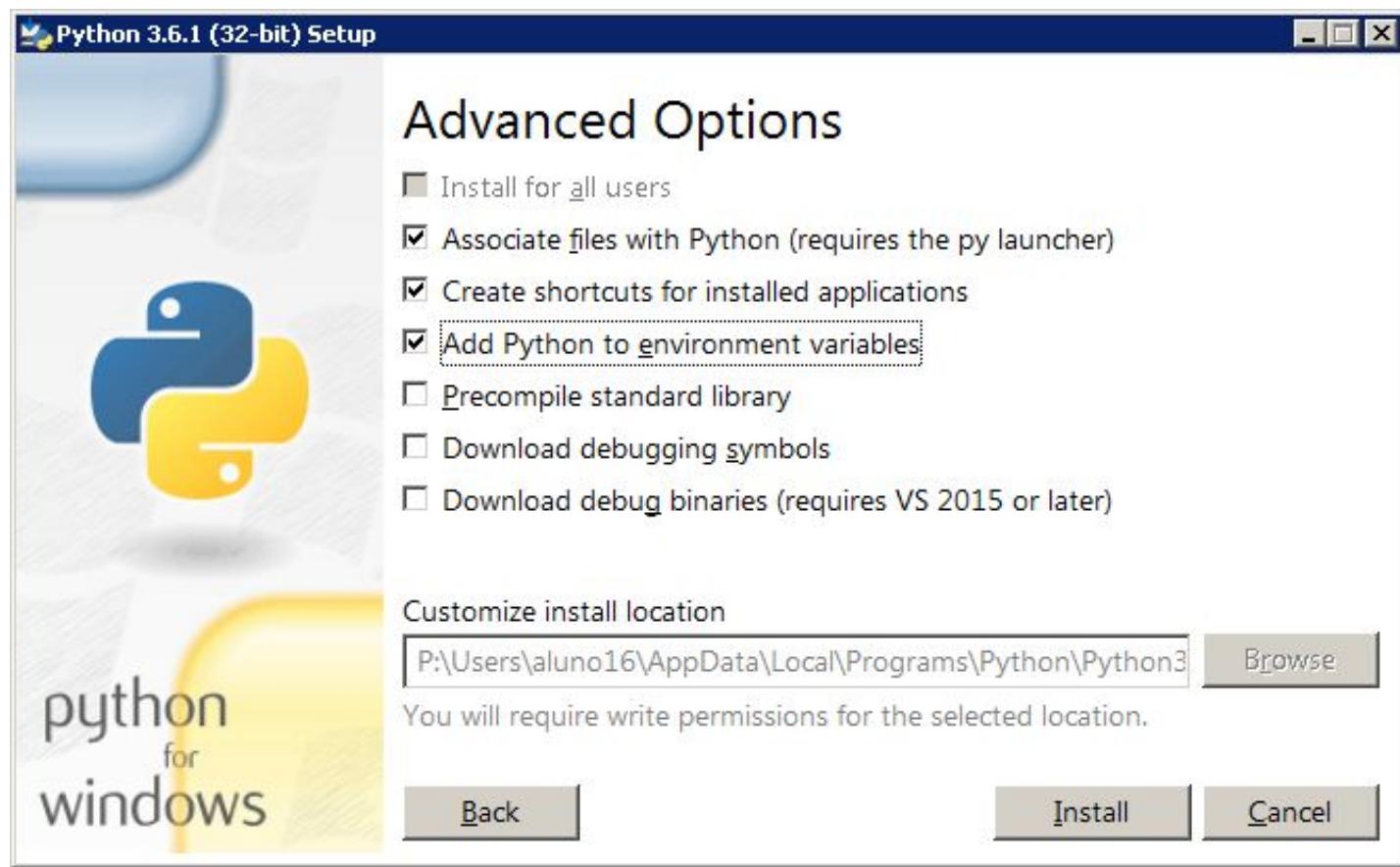
PARA FUNCIONAR O PIP



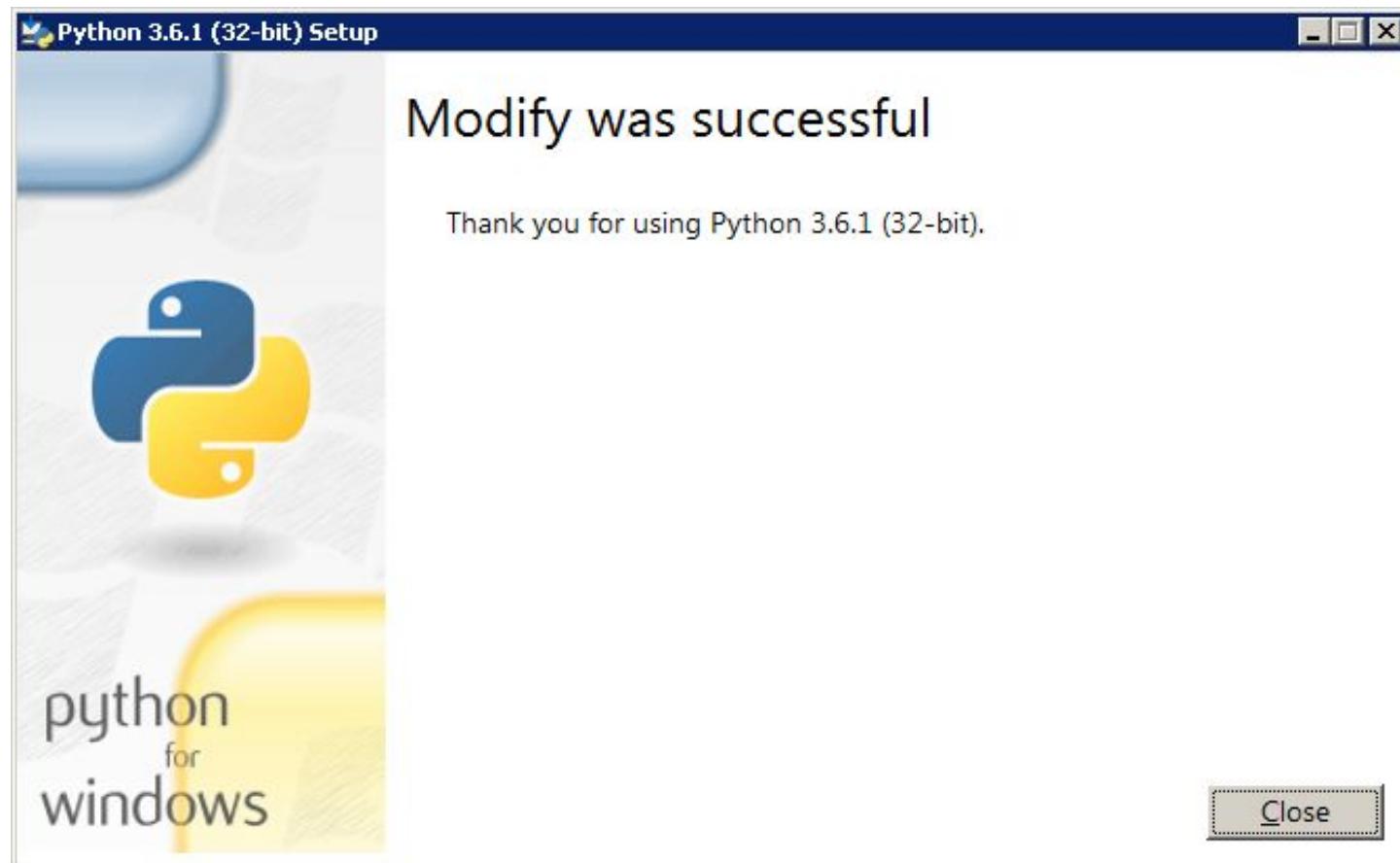
PARA FUNCIONAR O PIP



PARA FUNCIONAR O PIP



PARA FUNCIONAR O PIP





TODO MUNDO
CONSEGUIU INSTALAR?



BORA ABRIR A TELINHA
PRETA DE HACKER



Filtros ▾

Melhor correspondência



Python 3.6 (32-bit)

Aplicativo da área de trabalho



Python 3.6 Module Docs (32-bit)

Aplicativo da área de trabalho

Aplicativos



Python 3.6 Manuals (32-bit)



IDLE (Python 3.6 32-bit)

Comando



python 3.6

Sugestões de pesquisa



python 3.6 - Ver resultados da Web

Documentos

python 3.6 (32-bit)

Abra o python

Python 3.6 (32-bit)

```
Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 17:54:52) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```



UHUUL, VOCÊS ESTÃO NA
TELINHA PRETA HACKER



"Printando" na tela

Bom... Essas setas "```" é a indicação que ali vai ser digitado um "comando", que não é nada mais que a ação que você quer que o **Python realize.** (Desculpa essa frase ruim, não achamos jeito melhor de falar)

O primeiro comando que você irá aprender é o "**print**". Um comando que é responsável por mostrar/escrever/"printar" o que você escrever entre "``" na tela.

O resultado deste comando é mostrado logo abaixo.

```
>>> print("Essa vai ser a frase que será mostrada na tela")
```

Agora, nós, programadoras/es,
temos um rito de passagem:

Mostre na tela um "Olá mundo"



Saindo dessa telinha como
uma/um hacker

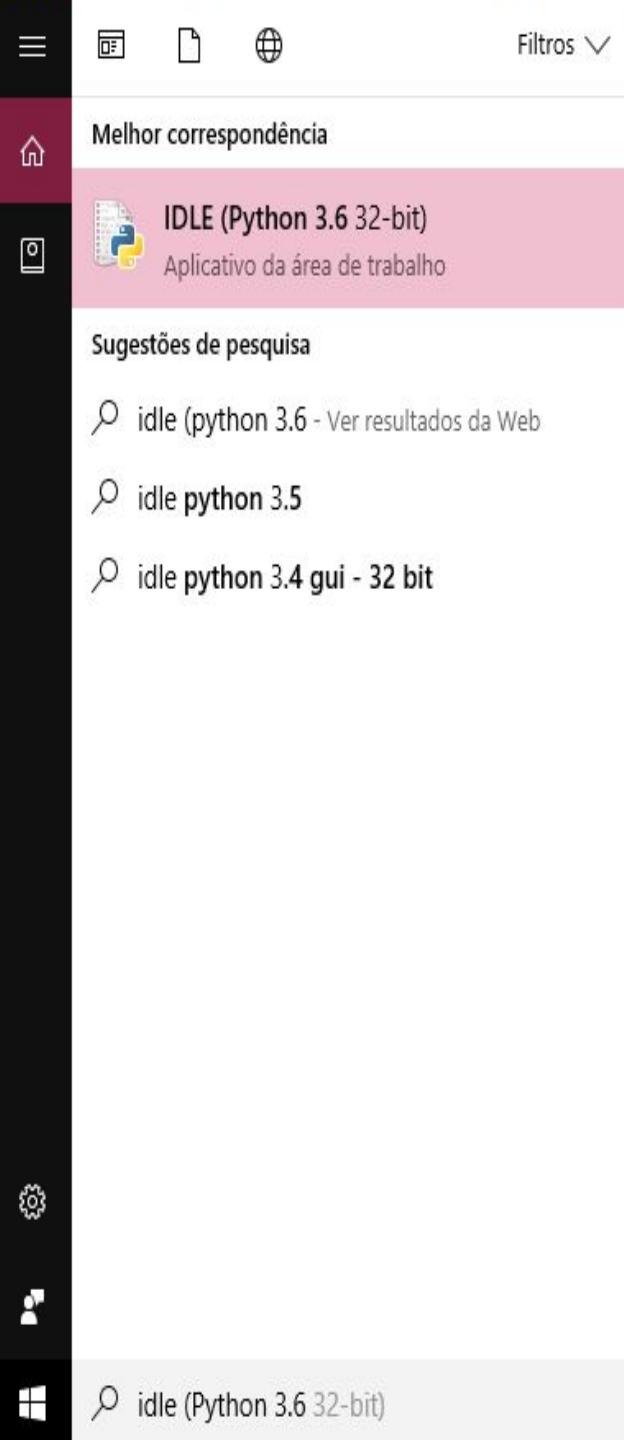
Digite "exit()" na linha de comando



Tá, mas não salvou, como eu
salvo para mostrar para a/o
crush?

Editores de código

Procure por "IDLE"



Python 3.6.1 Shell

File Edit Shell Debug Options Window Help

Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 17:54:52) [MSC v.1900 32 bit (AMD64)]
Type "copyright", "credits" or "license()" for more information.

```
>>> |
```



Filtros ▾

Melhor correspondência

 IDLE (Python 3.6 32-bit)
Aplicativo da área de trabalho

Sugestões de pesquisa

-  idle (python 3.6 - Ver resultados da Web)
-  idle python 3.5
-  idle python 3.4 gui - 32 bit



idle (Python 3.6 32-bit)

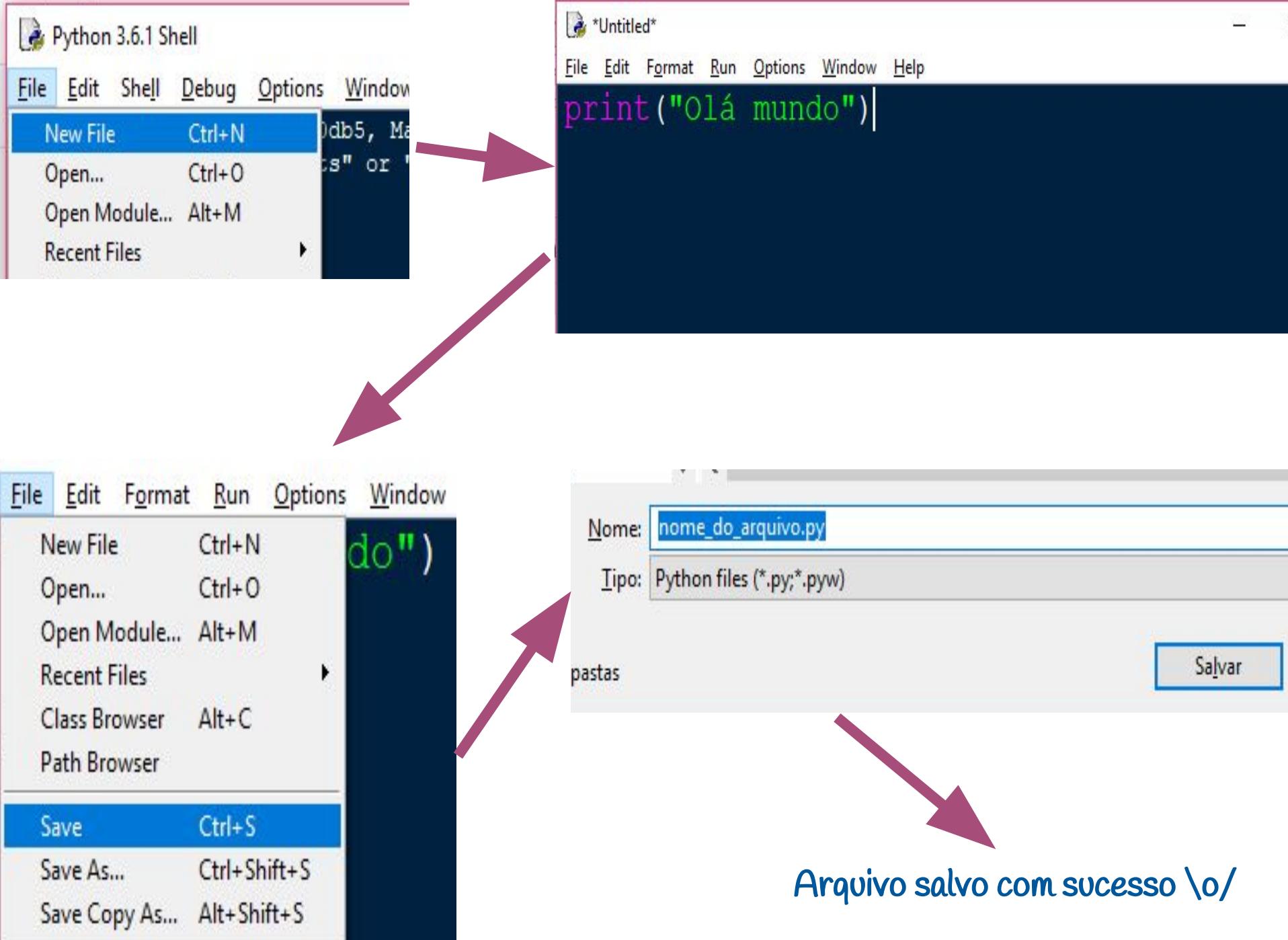


Editores de texto

Clique em:

- "File" > "New File" (ou ctrl + n)
- Coloque seu "print" no arquivo que você acabou de abrir
- "File" > "Save" (ou ctrl + s)
- Escolha o nome do arquivo

E... Pronto! Agora tudo o que você escreveu estará salvo!





Chega de enrolar... bora
aprender a codar o/

Variáveis

Variáveis são como **gavetas nomeadas** para o computador, onde ele **guarda as informações** que você dá a ele.

E como é "nomeada", você não precisa saber onde está a gaveta, só precisa saber o **nome** dela.

Quando você precisa buscar uma informação, você diz o nome da gaveta e o computador vai lá e "abre" esta gaveta.

Variáveis

Mas como tudo na vida... Tem regras!

- Podem ser usados **números, letras ou _**
- Nunca devem começar com um algarismo

E tem **palavras reservadas** que o Python usa para entender o que queremos que ele faça, como aquele "print" que demos.

Para conseguirmos nos comunicar com o Python, **não podemos dizer que uma dessas palavras reservas é uma gaveta**(variável), senão, ao invés do computador "printar" ele vai **abrir** uma gaveta (mostra o valor da variável).



Mas como isso fica no código?



>>> < nome > = < valor >

TUDO O QUE TIVER ENTRE "()" DEVE SER SUBSTITUÍDO PELO O QUE VOCÊS QUEREM

Tipos de variáveis simples

Para saber como organizar essas variáveis, o Python tem tipos de variáveis

- **Numéricos:** Armazena números. São: inteiros (`int`), de ponto flutuante (`float`)
- **Letras:** Armazena caracteres (qualquer um do teclado). São strings (`string`)
- **Lógicos:** Armazena verdadeiro ou falso. São: booleanos (`bool`)

Numéricos

- **Tipo inteiro (int):** números inteiros, sem ponto da casa decimal.

```
>>> a = 2  
>>> b = 4  
>>> c = 0
```

- **Tipo ponto flutuante (float):** números que aparece o ponto da casa decimal.

```
>>> d = 2.0  
>>> e = 4.5  
>>> f = 0.0
```

Letras e Lógicos

- **String (str)**: comprehende todos os caracteres dentro de aspas (simples ou duplas)

```
>>> j = "2"
```

```
>>> k = "2.0"
```

```
>>> l = "12@#nikukyu"
```

```
>>> m = "ççççççç"
```

- **Booleanos (bool)**: comprehende respostas do tipo True ou False (Verdadeiro ou Falso)

```
>>> n = True
```

```
>>> o = False
```



Tá, como eu sei se o Python
entendeu que a variável l é do
tipo lógico e não uma string?

Variáveis

Existe uma forma de pedir para o Python te dizer qual o tipo do valor:

Sintaxe:

```
type(<nome_da_variavel>)
```

Exemplo:

```
>>> type(m)
<type "bool">
```



Exercícios \o/

1) Teste algumas variáveis definidas anteriormente.

2) Tente definir `m = true`.

O que acontece? Por que dava certo quando era `True`?

Tente `m = "true"` e verifique o tipo de `m`. Qual sua teoria para explicar o que aconteceu?

Perdeu alguma variável pelo caminho?

`a = 2`

`b = 4`

`c = 0`

`d = 2.0`

`e = 4.5`

`f = 0.0`

`g = "2"`

`h = "2.0"`

`i = "12@#nikukyu"`

`j = "§§§§§"`

`k = True`

`l = False`



Mudando o tipo

Outra forma de transformar um tipo de variável em outro é usando os comandos:

- `int()`
- `float()`
- `bool()`
- `str()`

Exemplos:

```
>>> a = 2
>>> type(a)
<class "int">

>>> b = float(a)
>>> type(b)
<class "float">

>>> c =
bool(True)
>>> type(c)
<class "bool">

>>> d = str(a)
>>> d
"2"

>>> e = int(d)
>>> e
2
>>> type(e)
<class "int">
```



Mas o que podemos fazer com
essas variáveis?



Podemos fazer operações matemáticas:

Operadores numéricos básicos:

Adição: $+$

Subtração: $-$

Divisão: $/$

Multiplicação: $*$

Potenciação: $**$

Resto de uma divisão:
 $\%$



Exemplos:

```
>>> a = 2
>>> b = 3
>>> c = (a + b - 2 * a + a ** a) / 5
>>> c
1.0
>>> 10 % 8 # resto da divisão entre números inteiros
2
>>> 5 / 2 # quero que retorne um número
"quebrado"
2.5
>>> 5 // 2 # quero que retorne um inteiro
2
```

Operadores lógicos

Utilizamos quando queremos ter respostas lógicas, isto é, que retornam True ou False (verdadeiro/falso):

Maior que: >

Menor que: <

Maior ou igual a: >=

Menor ou igual a: <=

Idêntico: ==

Diferente de: !=

Não: not

E: and (exige que todas as condições sejam satisfeitas)

Ou: or (apenas uma das condições precisa ser satisfeita)



Exemplos

```
>>> a = 2
```

```
>>> b = 3
```

```
>>> a > b
```

False

```
>>> b > a
```

True

```
>>> a == b
```

False

```
>>> a != b
```

True

```
>>> a > b and b > a
```

False

```
>>> a > b or b > a
```

True

```
>>> not (a != b) == False
```

True

TABELA VERDADE

A	B	A and B	A or B	not (A)	not (B)
True	True	True	True	False	False
True	False	False	True	False	True
False	True	False	True	True	False
False	False	False	False	True	True



Exercícios \o/

Se $a = 2$, $b = 3$, $c = 2.0$ e $d = "2.0"$, faça as operações a seguir:

```
>>> a + b
```

```
>>> b ** a
```

```
>>> a + c
```

```
>>> a + d
```

E teste as condições:

```
>>> a == c
```

```
>>> a <= b
```

```
>>> a < b and b < c
```

```
>>> a < b or b < c
```

```
>>> a > c or a >= c
```

```
>>> not(a != b and b <= (a**2)-1)
```



*Mas só tem esses tipos de
variáveis?*

Tipos de variáveis compostas

Por que compostas?

R: Porque elas são compostas por mais de um valor "simples" (mais de um número, mais de uma string).

Pode ser valores diferentes?

R: No Python sim, mas algumas linguagens não permitem isso.

Quais são esses tipos?

R: Existem muitos tipos (lista, tupla, dict, etc...), porém vamos focar no principal:

Listas: permitem armazenar várias informações em uma mesma variável.

Listas

Como dissemos anteriormente, as listas permitem que adicionemos várias informações em uma variável só, independente do tipo dessa informação (números, letras e booleano). Inclusive, você pode colocar uma lista dentro de uma lista.

Sintaxe:

<variável> = [info1, info2, info3]

Exemplos:

```
>>> a = ["String", 9, True]  
>>> a  
["String", 9, True]  
>>> b = [[“Nome”, 99999], True]  
>>> b  
[[“Nome”, 99999], True]  
>>> type(b)  
<class "list">
```



*Mas como eu resgato uma dessas
informações?*



Como recuperar um valor de uma Lista

Índice em uma lista: número que indica a posição de cada caractere na string. O número será 0 (zero) para o primeiro caractere na string, 1 para o segundo, 2 para o terceiro, etc.

Sintaxe:

<lista>[número]

["a", 2, True, 1.0,
 "Ladies"]

"a"	2	True	1.0	"Pyladies"
0	1	2	3	4

```
>>> a = ["a", 2, True, 1.0, "Pyladies"]
>>> a[0]
"a"
>>> a[2]
True
>>> a[3]
[1]
```

Tamanho da sua Lista

Para saber o tamanho da sua lista, basta pedir para o Python a partir da comando "`len`".

Sintaxe:

```
len(<lista>)
```

Exemplos:

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

```
>>> len(a)
```

```
3
```



Beleza, eu criei minha lista, posso
colocar mais coisas nela?



Adicionando mais itens a minha lista

Existem duas formas de fazer isso: "concatenando" listas ou adicionando o item com a comando **append**.

Sintaxe:

<lista>.append(<item para adicionar>)

ou

<lista1> + <lista2>

Exemplos:

```
>>> a = [1,2]
>>> a.append(3)
>>> a
[1,2,3]
>>> b = [4, 5]
>>> a + b
[1,2,3,4,5]
```



Vamos aprender mais coisas \o/

Condicionais

Bom... Na nossa vida nem sempre andamos reto, certo? Pode ter lugares onde temos que tomar uma decisão: direita, esquerda, continuar em frente ou voltar.

Essas decisões são **condicionais** na programação, são ações que tomamos a partir de algum acontecimento.

MAS COMO ASSIM???



Bom, vamos tentar resolver um problema: a lâmpada queimou, o que devemos fazer?



Primeiro, precisamos saber se
temos uma escada. Certo?



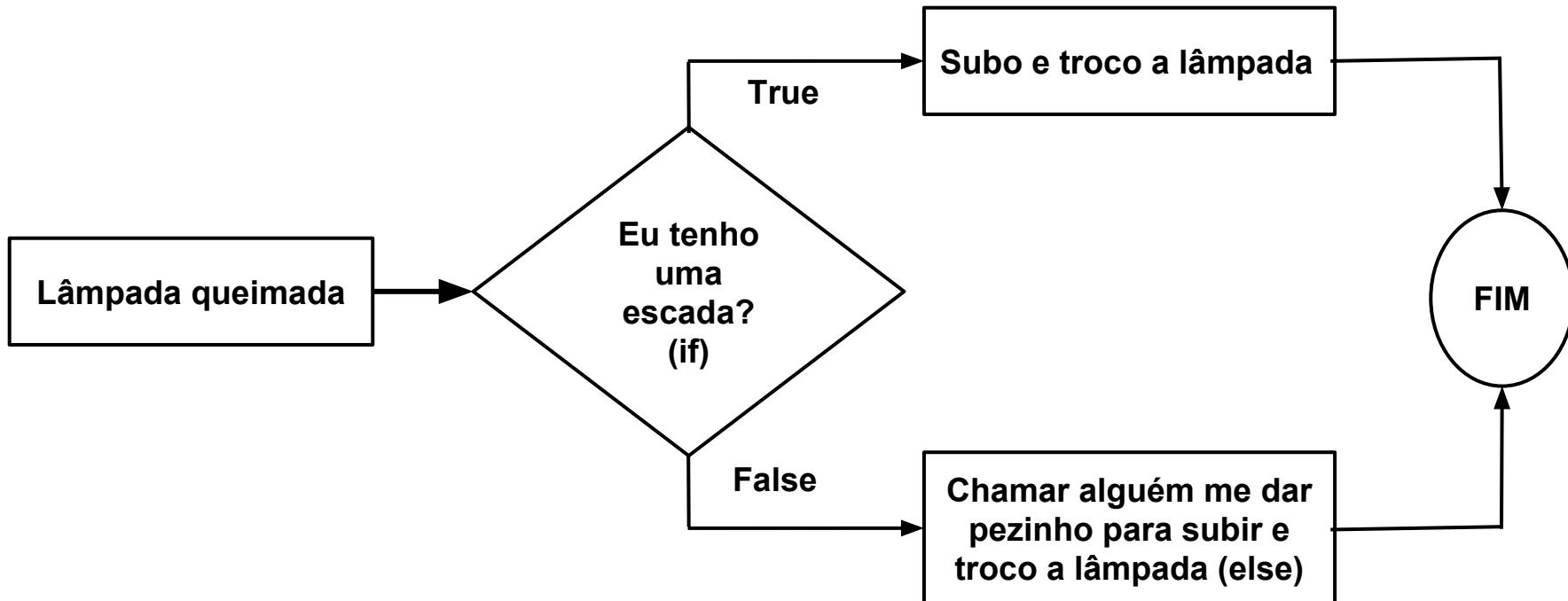
Se não temos uma escada, o que faremos?



E se pedirmos um pezinho? Pode ser?

Condicionais

Bom... Tudo o que fizemos, foi tomar decisões, isso é: você **condicionou**:
se (if) eu tiver uma escada, eu subo nela e troco a lâmpada
caso contrário (**else**), eu chamo alguém para me dar pezinho, subo e
continuo o processo de trocar a lâmpada.





Ok, entendemos isso. Agora,
como codamos isso?



Condicionais

Sintaxe:

if <condição dada por operador booleano>:

<o que tenho que fazer, caso a condição seja satisfeita>

else:

<o que tenho que fazer, caso a condição não seja satisfeita>



Condicionais

Exemplo:

```
tem_escada =input("Tem alguma escada por aí?")
```

```
if tem_escada == "sim":
```

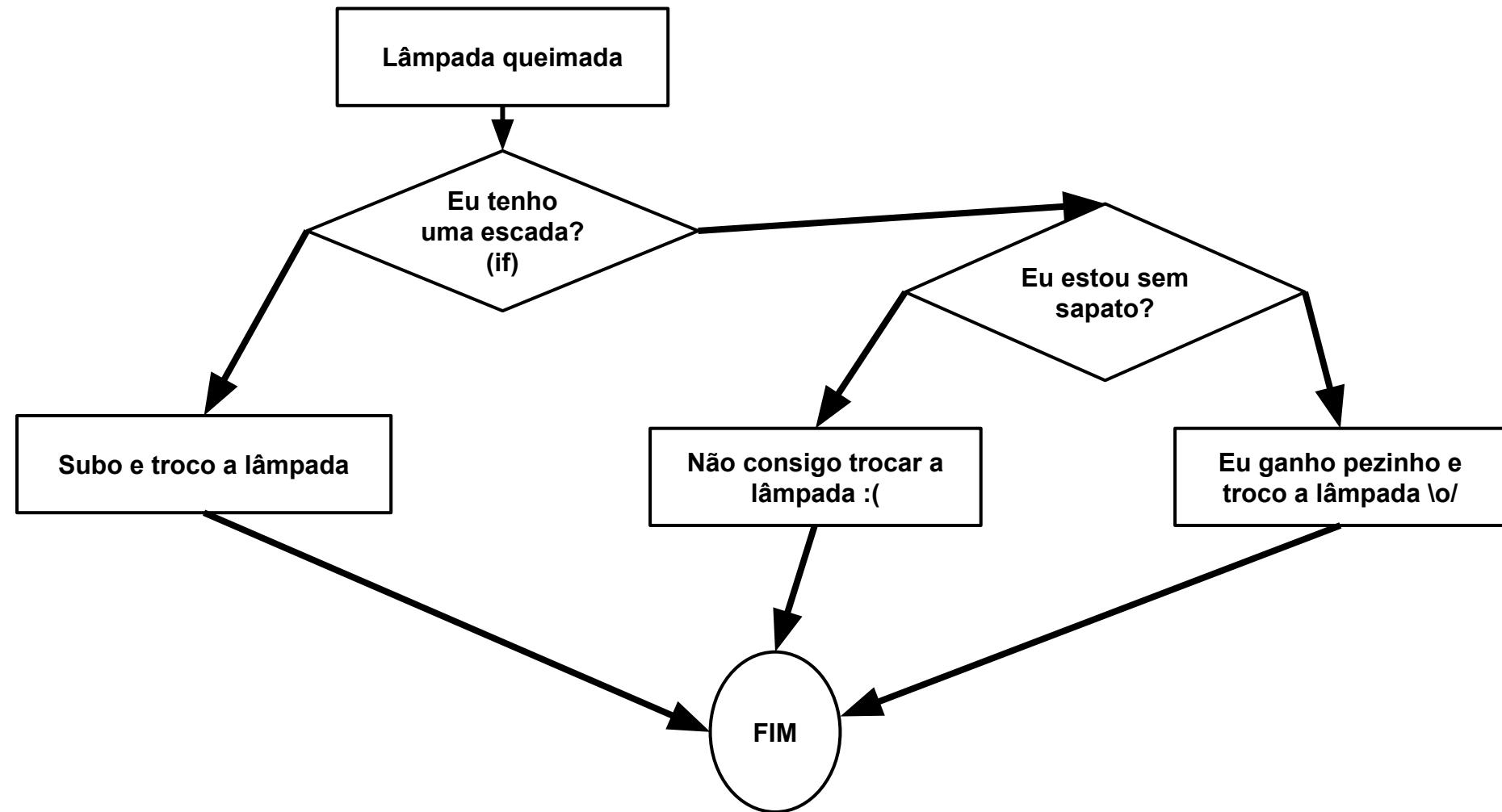
```
    print("Subir na escada e trocar a lâmpada")
```

```
else:
```

```
    print("Pedir para alguém fazer pezinho para eu subir e trocar a  
lâmpada")
```

Condicionais

Mas caso só me ajudam se eu tirar o sapato. Então, se eu tirar o sapato, eu uso o pezinho da pessoa e continuo o processo: **elif**



Condicionais

Sintaxe:

if <condição dada por operador booleano>:

<o que tenho que fazer, caso a condição seja satisfeita>

(caso a condição anterior não seja satisfeita, tenho mais uma condição para verificar)

elif <condição dada por operador booleano>:

<o que tenho que fazer se a segunda condição for satisfeita>

else:

<o que tenho que fazer, caso nenhuma das condições acima sejam satisfeitas>



Condicionais

Exemplo:

```
tem_escada =input("Tem alguma escada por aí?")
sem_sapato =input("Estou sem sapato?")
if tem_escada == "sim":
    print("Subir na escada e trocar a lâmpada")
elif sem_sapato == "sim":
    print("Pedir para alguém fazer pezinho para eu subir")
else:
    print("Não consigo trocar a lâmpada :(")
```

Então, vamos para um problema
de programadoras/es...

O PROBLEMA DE SER UMA PROGRAMADORA



Minha mãe disse:
"Filha, vá no mercado e
compre 1 garrafa de leite. Se
tiverem ovos, traga 6"

Eu voltei com 6 garrafas de
leite.

Ela disse:"Por que diabos você
comprou 6 garrafas de leite?"

"PORQUE TINHAM OVOS"



Exercício \o/

Crie um código para uma outra pessoa executar.
Pergunte seu nome e a convide para assistir uma
palestra. Deixe a opção para que ela recuse; caso
escolha esta opção escreva um smile triste.

Tempo: 7 minutos.

```
variavel = input("Pergunta?")
```

Contadores e Acumuladores

Contadores: variáveis auxiliares que usamos para incrementar valores fixos, como por exemplo somar o valor 1 em uma variável. O valor pode ser qualquer valor positivo ou negativo.

Acumuladores: Contadores que incrementam valores diferentes.

Exemplo

Soma de 3 números dados pelo usuário.

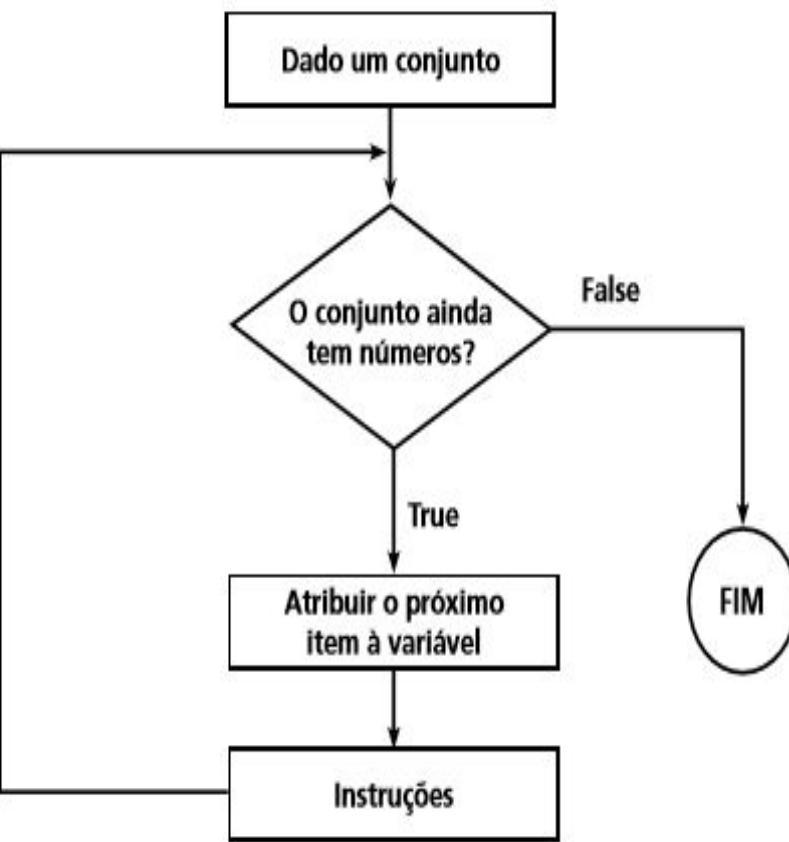
```
>>> total = 0
>>> num = int(input("entre com um número: "))
>>> total = total + num
>>> num = int(input("entre com um número: "))
>>> total = total + num
>>> num = int(input("entre com um número: "))
>>> total = total + num
>>> print("O total é: {}".format(total))
```



Tah, mas...
Como otimizar isso??

for (Repetições)

O comando **for** itera sobre os itens de qualquer tipo de sequência (lista ou string), na ordem em que eles aparecem na sequência. A variável que aparecem na linha do for se comporta como cada item da **lista**.



Sintaxe:

```
for <variável> in <lista>:  
<comando que quero executar>
```

Exemplos

Exemplo 1:

```
nomes = ["Giana", "Ana", "Caroline", "Debora",
          "Carlos", "Jussara", "Veronica"]
```

```
for nome in nomes:
```

```
    if nome.startswith("C") and nome.endswith("e"):
        print(nome)
```

Exemplos

Exemplo 2: Somar todos os números ímpares da lista.

```
print("Soma de números ímpares")
```

```
numeros = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

```
soma_impar= 0
```

```
for numero in numeros:
```

```
    if (numero % 2) != 0:
```

```
        soma_impar= soma_impar+ numero
```

```
        print("Somando o número ímpar: {}".format(numero))
```

```
        print("A soma dos ímpares é: {}".format(soma_impar))
```

```
print("A soma de números ímpares da lista é
```

```
{}.format(soma_impar)
```



Estão precisando de uma pausa?
Acordadas/os?

Funções

Funções são pedaços de código que servem para executar um procedimento muitas vezes, evitando que você tenha que reescrevê-lo mais de uma vez.

Uma funcionalidade importante é o fato que, caso precise realizar alguma alteração ou correção, ela vai ser feita neste pedaço de código e não em diversas partes do código



Funções

Sintaxe:

Quando a função não recebe parâmetros:

```
def <nome da função> ():  
    <pedaço do código que você quer executar>
```

OU

Quando a função recebe **parâmetros**:

```
def <nome da função> (<parâmetro(s)>):  
    <pedaço do código que você quer executar>  
    return (caso essa função retorne algum valor)
```



Funções

O comando `input()`, usado nos exemplos, é na verdade uma função nativa no Python.

Ela solicita uma informação do usuário e nos retornar o valor informado.

Agora vamos criar a nossa própria função :)



Exemplos

Exemplo1: Uma função que soma.

```
def soma(a, b):  
    return a + b
```

```
print(soma(1, 2))  
>>> 3  
#ou  
print(soma('PyLadies ', 'Floripa'))  
>>> PyLadies Floripa
```



Exemplos

Exemplo 2: Uma função que multiplica.

```
def multiplica(n1, n2):  
    return n1 * n2
```

```
n1 = float(input('Informe o primeiro número: '))  
n2 = float(input('Informe o segundo número: '))  
print(multiplica(n1, n2))
```

Resultado:

```
>>>  
Informe o primeiro número: 6  
Informe o segundo número: 7  
42.0  
>>>
```



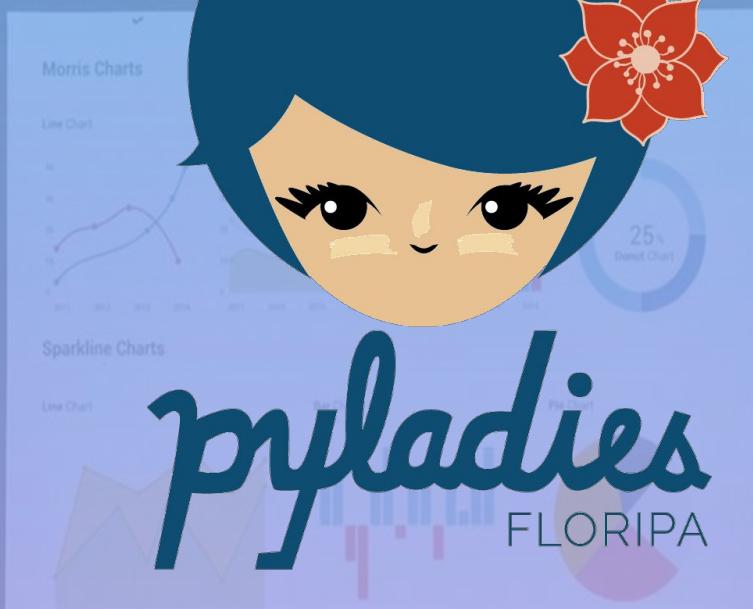
Exercício \o/

Faça uma função para pedir ao usuário dois números e calcular a subtração entre eles.



Gratidão pela participação!

Bora entender
Machine Learning?



Introdução a Python

Algumas coisas para aprender
antes de Machine Learning

Giana de Almeida



Referências



Referências

<https://github.com/PyLadiesSP/Cursos>

<http://wiki.python.org.br/PrincipiosFuncionais>

[Curso Python para Zumbis](#)

[Curso “An Introduction to Interactive Programming in Python” - Coursera](#)

<http://www.peachpit.com/articles/article.aspx?p=1312792&seqNum=6>

<https://docs.python.org/release/2.3.5/whatsnew/section-slices.html>

<http://www.bbc.co.uk/education/guides/zqh49j6/revision/3>

<https://www.youtube.com/watch?v=SYioCdLPmfw>

https://pt.wikibooks.org/wiki/Python/Conceitos_b%C3%A1sicos/Tipos_e_operadores

<http://www.dcc.ufrj.br/~fabiom/mab225/02tipos.pdf>

www.arquivodecodigos.net/principal/dicas_truques_categoria2.php?linguagem=12&categoria1=1&categoria2=59

https://docs.google.com/presentation/d/1C1T5RyHNFTkPy-fQ3CstZ_KgkWrcDo4e-Bryf5uk9Rk/edit?usp=sharing

<https://www.youtube.com/watch?v=pz1CnWJ8qbY&list=PLxNM4ef1Bpxin3gBz3RMqUvawK6xHjvYo&index=1>

Referências

www.arquivodecodigos.net/principal/dicas_truques_categoria2.php?linguagem=12&categoria1=1&categoria2=51

www.dotnetperls.com/lower-python

<https://pt.wikipedia.org/wiki/Algoritmo>

<http://wiki.python.org.br/SoftwarePython>

<http://wiki.python.org.br/EmpresasPython>

<https://powerpython.wordpress.com/2012/03/16/programas-e-jogos-feitos-em-python/>

http://tutorial.djangogirls.org/pt/python_installation/index.html

<https://powerpython.wordpress.com/2012/03/19/aula-python-17-estrutura-de-decisao/>

<https://under-linux.org/entry.php?b=1371>

<http://aprenda-python.blogspot.com.br/2009/10/nova-formatacao-de-strings.html>

<https://docs.python.org/3/library/string.html#format-spec>

<https://www.python.org/dev/peps/pep-3101/>