Ciências ULisboa
Faculdade de Ciências da Universidade de Lisboa
DEPARTAMENTO·DE·INFORMÁTICA

Folha 0 – Fazer funcionar

Python 3.*

Interagir com Python

Uma das razões pelas quais Python é uma linguagem fácil de usar deve-se do facto de vir com ferramentas que ajudam a desenhar, escrever e depurar (fazer *debug*) os programas. Algumas destas ferramentas são:

A. Modo interativo – Onde se escreve instruções para Python uma linha de cada vez, de uma forma semelhante às instruções que um sistema operativo recebe da linha de comandos. É também possível escrever pequenos programas de múltiplas linhas ou importar código de ficheiros de textos ou de módulos *built-in* de Python.

B. Scripts e módulos – Os comandos feitos no modo interativo não são guardados quando se sai deste modo, pelo que quando queremos guardar o trabalho devemos criar ficheiros de texto onde colocamos o código a executar. Estes ficheiros são chamados de scripts ou módulos e podem ser executados a partir da linha de comandos.

C. IDLE – É um ambiente de desenvolvimento integrado que inclui o modo interativo do Python e mais ferramentas para escrever e correr programas. O IDLE é escrito em Python e mostra as capacidades da linguagem.
Durante este guião iremos interagir com o Python das três formas descritas acima. Para nos ambientarmos com as ferramentas e à linguagem propomos um conjunto de exemplos clássicos.

As operações que devem ser executadas e experimentadas encontram-se escritas a **azul**.

A. Modo interativo

No modo interativo é possível fazer quase tudo o que se pode fazer num programa de Python, até escrever programas com múltiplas linhas. Podemos pensar no modo interativo como:

- Uma sandbox para experimentar Python de forma segura;
- Um tutor;
- Uma ferramenta para encontrar e resolver problemas (bugs) nos programas.

Aviso Não é possível guardar o que se escreve no modo interativo. Para manter uma cópia é preciso criar um registo à parte com o código que escrevemos e os resultados obtidos.
É possível usar o modo interativo para manipular texto, fazer atribuições e usar como uma calculadora. Podemos também importar módulos, funções, partes de um programa mais longo e testá-los. Estas características ajudam a:

- Fazer experiências com objetos Python sem ter de escrever programas longos;
- Depurar programas importando certas partes do programa de cada vez.

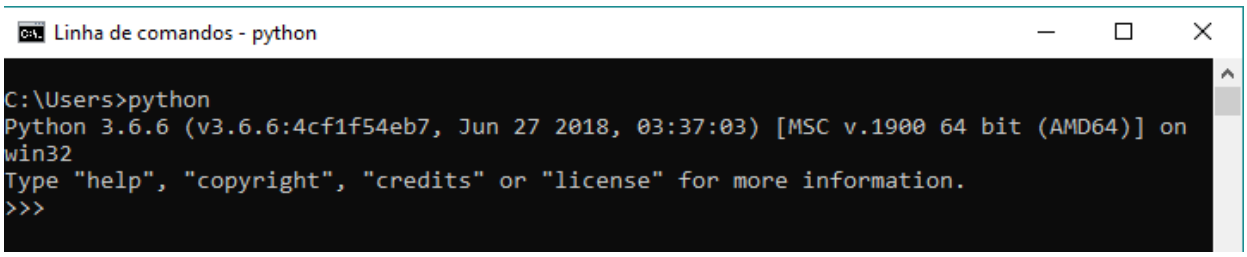
A1. Iniciar o modo interativo

Passos para iniciar o modo interativo de Python:

1. Abrir uma janela de linha de comandos.
    - Em Windows, procurar por *Linha de Comandos* no menu de pesquisa.
    - Em Linux, procurar por *Terminal* na barra de procura.
2. Escrever: `python <enter>`

Quando o Python abre, aparece o texto da Figura abaixo.

O modo interativo inicia escrevendo a versão que está a correr, a data em que a versão foi lançada e algumas dicas do que se segue. Depois mostra o *prompt* do Python: >>>



2. *"Hello, World!"*

Os programas "Hello, World!" são uma tradição de programação, normalmente usados para introduzir um programador a uma nova linguagem de programação. Estes programas têm o simples objetivo de escrever no ecrã a mensagem "Hello, World!". De acordo com a Wikipedia (http://www.wikipedia.org), a primeira instância de um programa que imprimiu "Hello, World!" ocorreu em 1973 e desde aí este programa é escrito em quase todas as linguagens de programação.
Uma razão pela qual os programas "Hello, World!" são populares deve-se ao programa ter de imprimir uma só afirmação que é normalmente o programa mais pequeno que se pode escrever numa linguagem.

- Em Python, o programa mais pequeno que funciona tem apenas uma linha.
- Em Java, o programa com mesmo intuito tem no mínimo 5 linhas.

O método mais básico de trabalhar com o modo interativo é o seguinte:

1. Escrever uma declaração ou expressão.
2. Premir a tecla de *Return* ou *Enter*.

Quando se executa o segundo passo, o Python interpreta o input e responde se as instruções introduzidas estiverem à espera de resposta ou se o interpretador não entende as instruções dadas.
No próximo exemplo, a declaração diz ao Python para imprimir (*print*) a string.

1. Escrever `print("Hello, World!")` `<enter>`

Como a declaração não especifica onde se deve escrever a mensagem, o Python imprime-a na janela (o comportamento padrão do modo interativo).

```
>>> print ("Hello, World!")
Hello, World!
```

Esta declaração é um programa completo de Python. Usando o modo interativo, o Python processa cada linha de código que se escreve logo que se prime a tecla *Enter* (a não ser que veja que se está a escrever um programa de múltiplas linhas), e assim o resultado aparece depois.

A3. Ver informação sobre um objeto Python

No modo interativo há duas formas de ver a informação de um objeto:

- Escrever o objeto ou o seu nome e premir Enter.
- Escrever o comando `print` e o objeto ou o seu nome e premir Enter.

O que vemos depende do que é o objeto.

- Com alguns tipos de dados (por exemplo inteiros e listas), os dois métodos de visualizar o valor têm o mesmo resultado. No exemplo seguinte guardamos o valor inteiro 3 na variável com nome *x* e escrever o objeto ou imprimi-lo mostra o mesmo resultado.

1. Escrever `x = 3 <enter>`
2. Escrever `x <enter>`
3. Escrever `print(x) <enter>`

4. O mesmo acontece com o objeto com nome *y.*

1. Escrever `y = [1,2] <enter>`
2. Escrever `y <enter>`
3. Escrever `print(y) <enter>`

```
>>> x = 3
>>> x
3
>>> print(x)
3
>>> y = [1,2]
>>> y
[1, 2]
>>> print(y)
[1, 2]
```

4. Com strings, o resultado de escrever `print name` e premir Enter é um pouco diferente do resultado de escrever `name` e premir Enter. No segundo caso, o valor que é impresso aparece entre aspas enquanto no primeiro caso, o resultado impresso não contém as aspas.
   No próximo exemplo podemos ver a diferença entre usar apenas o nome ou a declaração de print com o nome.

   1. Escrever `x = "mystring" <enter>`
   2. Escrever `x <enter>`
   3. Escrever `print(x) <enter>`

```
>>> x = "mystring"
>>> x
'mystring'
>>> print (x)
mystring
```

   Em Python as aspas " " ou plicas ' ' têm o mesmo significado, o que não acontece noutras linguagens.

A4. Ver o resultado da última expressão

Quando escrevemos uma expressão no modo interativo ou quanto o Python retorna uma expressão como resultado de algo que escrevemos, o Python guarda o valor da expressão num nome especial: _ (o caracter *underscore*).

1. Escrever `"Hello, World!" <enter>`
2. Escrever `_ <enter>`

```
>>> "Hello, World!"
'Hello, World!'
>>> _
'Hello, World!'
```

Contudo o _não guarda resultados de declarações (ex: x = 24 ou comandos como print). Não devemos confiar no uso do _ em longos segmentos de código uma vez que o resultado guardado em _ pode mudar se não estivermos com atenção à diferença entre as declarações e expressões:

1. Escrever `"Hello, Nurse!" <enter>`
2. Escrever `x = 35 <enter>`
3. Escrever `_ <enter>`

```
>>> "Hello, Nurse!"
'Hello, Nurse!'
>>> x = 35
>>> _
'Hello, Nurse!'
```

A5. Manipular strings

- Vírgula como espaço branco

Podemos usar o modo interativo do Python para ver alguns ações interessantes que o Python consegue fazer com strings.
Quando queremos imprimir várias strings ou outros objetos, podemos usar uma vírgula para simbolizar um espaço em branco no output produzido, como podemos ver no exemplo.

1. Escrever `a = "The meaning of Life, the Universe, and Everything is" <enter>`
2. Escrever `b = 42 <enter>`
3. Escrever `print(a,b) <enter>`

```
>>> a = "The meaning of Life, the Universe, and Everything is"
>>> b = 42
>>> print (a, b)
The meaning of Life, the Universe, and Everything is 42
```

- Medir e separar strings

Podemos obter o tamanho de uma string usando a função `len()`, que é uma abreviação de *length*. Esta função pode ser aplicada em outros tipos de dados com sequências como por exemplo listas.

1. Escrever `x = "abc" <enter>`
2. Escrever `len(x) <enter>`
3. Escrever `y = "raxacoricofallapatorius" <enter>`
4. Escrever `len(y) <enter>`

```
>>> x = "abc"
>>> len(x)
3
>>> y = "raxacoricofallapatorius"
>>> len(y)
23
```

Para separar strings em pedaços mais pequenas usamos o método `split()` que retorna uma lista com as várias partes obtidas da divisão. Usando este método sem argumentos (nada dentro do parêntesis) iremos dividir a string pelos espaços presentes.

1. Escrever `x = "This is an ex-parrot!" <enter>`
2. Escrever `x.split() <enter>`
3. Escrever `y = "one and/or two" <enter>`
4. Escrever `y.split() <enter>`

```
>>> x = "This is an ex-parrot!"
>>> x.split()
['This', 'is', 'an', 'ex-parrot!']
>>> y = "one and/or two"
>>> y.split()
['one', 'and/or', 'two']
```

A6. Modo interativo como calculadora

O interpretador de Python pode ser usado como uma calculadora. Se quisermos fazer cálculos simples podemos escrever os números e os operadores e premir Enter para ver o resultado.

1. Escrever `(1 + 3) * (2 + 2) <enter>`
2. Escrever `1 + 3 * 2 +2 <enter>`

```
>>> (1 + 3) * (2 + 2)
16
>>> 1 + 3 * 2 + 2
9
```

Aviso Não podemos usar o sinal = para fazer cálculos deste género. Em Python o este sinal é usado para dar um valor a uma variável.

Também é possível usar nomes para fazer os cálculos no interpretador de Python, o que torna mais fácil resolver cálculos com vários passos.

```
>>> x = 1 + 3
>>> y = 2 + 2
>>> x * y
16
```

A7. Sair do modo interativo

Para sair do modo interativo do Python podemos chamar a função `quit()` ou `exit()` que funcionam para os vários sistemas operativos. Ao sair do modo interativo voltamos à janela da Linha de Comandos.

1. Escrever `quit() <enter>`

Aviso Ao sair do interpretador, todos os valores dados a variáveis desaparecem. Para manter um registo do trabalho podemos copiar os comandos executados para um ficheiro de texto à parte.

A8. *Help*

Existem vários recursos disponíveis que auxiliam a programação em Python, um dos mais completos é o sistema de ajuda integrado no Python.
Este sistema assume que o utilizador já tem algum conhecimento de programação pelo que a terminologia usada é um pouco complexa no início.

Help no modo interativo

Abrindo novamente o modo interativo:

1. Abrir uma janela de linha de comandos.
   - Em Windows, procurar por *Linha de Comandos* no menu de pesquisa.
   - Em Linux, procurar por *Terminal* na barra de procura.

2 Escrever: `python <enter>`

podemos aceder ao sistema de ajuda de duas formas.

- Entrar no programa de ajuda

1. Escrever `help()` `<enter>`

O programa *help* abre-se e aparece uma mensagem de boas-vindas e alguns tópicos sugeridos. Depois mostra que está pronto para receber comandos.

```
>>> help()

Welcome to Python 3.6's help utility!

If this is your first time using Python, you should definitely check out
the tutorial on the Internet at https://docs.python.org/3.6/tutorial/.

Enter the name of any module, keyword, or topic to get help on writing
Python programs and using Python modules.  To quit this help utility and
return to the interpreter, just type "quit".

To get a list of available modules, keywords, symbols, or topics, type
"modules", "keywords", "symbols", or "topics".  Each module also comes
with a one-line summary of what it does; to list the modules whose name
or summary contain a given string such as "spam", type "modules spam".

help>
```

- Escrever o nome do item sobre o qual se quer saber mais.
  Podemos por exemplo procurar pelo tipo de dado lista e os seus métodos.

  1. Escrever `list` `<enter>`

```
help> list
```

- Para sair do modo *help* e retornar ao modo interativo basta escrever `quit`.

  1. Escrever `quit` `<enter>`

```
help> quit
```

- Usar o *help* sem sair do modo interativo
  É possível usar o modo *help* para obter uma dica sobre um item em particular sem sair do modo interativo.
  Na linha de comando escrever `help` seguindo do nome do item sobre o qual se quer saber mais entre parêntesis.
  Por exemplo para saber mais sobre o tipo de dados *list* escrevemos:

1. Escrever `help(list) <enter>`

```
>>> help(list)
```

Se a informação estiver toda em apenas uma janela, o terminal volta ao modo interativo. Caso contrário, para visualizar a restante informação podemos usar a barra de espaço ou o Enter até não haver mais informação.

B. Scripts e Módulos

Uma vez que o modo interativo do Python não guarda o trabalho feito quando se sai do modo, teremos de guardar o trabalho em ficheiros de texto. Os ficheiros de texto que contêm código Python chamam-se *scripts* (se forem o programa completo) ou módulos (se contiverem partes de código que podem ser importadas para outros programas). Estes ficheiros têm de terminar com o sufixo **`.py`**.

Executar um *script* na linha de comandos

Abrindo a linha de comandos podemos executar um *script* escrevendo `python` e o nome do ficheiro.

```
C:\Users>python script.py
```

Atenção Para executar o ficheiro, a linha de comandos tem de estar no mesmo diretório do ficheiro, caso contrário não será possível encontrar o ficheiro.
Quando termina a execução do ficheiro na linha de comandos, voltamos ao terminal normal, se quisermos ir para o modo interativo depois de executar o ficheiro podemos usar o modificador `-i` ao executar o comando.

```
C:\Users>python -i script.py
```

Para criar um script e o correr seguimos os próximos passos.

1. Usar o editor de texto para escrever o script.

   Podemos usar qualquer editor disponível, se estivermos em Linux podemos usar por exemplo o *gedit*, em Windows podemos usar o *Visual Studio Code*. No novo ficheiro podemos colocar o seguinte código:

```
print ("teste de como os scripts funcionam")
x = 500
print ("o valor de x e", x)
```

e

Atenção Se copiarmos diretamente o texto de um ficheiro word ou pdf, temos de verificar se as aspas são reconhecidas como indicador de string (podemos apaga-las e voltar a escrever no editor de texto).

2. Guardar o script numa pasta e dar-lhe um nome com o sufixo `.py`

   O script criado em cima poderá ter o nome de `tinyscript.py`

3. Executar o script na linha de comandos.

   Na linha de comandos, sabemos em que diretório estamos a partir do texto que aparece quando abrimos o terminal.

```
C:\Users>
```

   Neste caso estamos na pasta *Users*, mas dependendo do computador, podemos estar noutra pasta. Para vermos quais os ficheiros e diretórios aos quais podemos aceder a partir daqui usamos o comando `ls` em Linux e `dir` em Windows. Sabendo quais os ficheiros e diretórios a que podemos aceder, podemos movimentar-nos pelas pastas até chegar ao nosso *script* para o executar. Para trocar de diretório usamos o comando `cd` (que significa *change directory*) seguido do nome do diretório (podemos usar a tecla do *tab* para auto-completar o nome do diretório).
   A maneira mais direta de acedermos à pasta onde guardámos o nosso *script* (por exemplo, a pasta *Desktop*) é usando o comando seguinte, substituindo xxxxx pelo número de aluno respetivo:

   `1. cd \Users\fcxxxxx\Desktop <enter>`

```
C:\Users>cd MyUser/Desktop

C:\Users\MyUser\Desktop>dir
 Volume in drive C is OS
 Volume Serial Number is 6C7E-DE9F

 Directory of C:\Users\MyUser\Desktop

13/09/2019  18:13    <DIR>          .
13/09/2019  18:13    <DIR>          ..
13/09/2019  18:13                93 tinyscript.py
               1 File(s)             93 bytes
               2 Dir(s)  794 664 861 696 bytes free
```

   Uma vez que já estamos no diretório onde se encontra o nosso ficheiro podemos executar o *script*.

2. `python tinyscript.py <enter>`

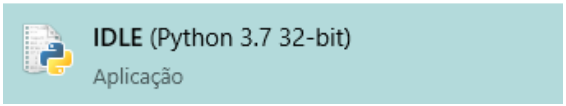
```
>python tinyscript.py
teste de como os scripts funcionam
o valor de x e 500
```

C. IDLE

IDLE é um acrónimo para Interactive DeveLopment Environment e é um programa de edição escrito completamente em Python.

C1. Abrir o IDLE

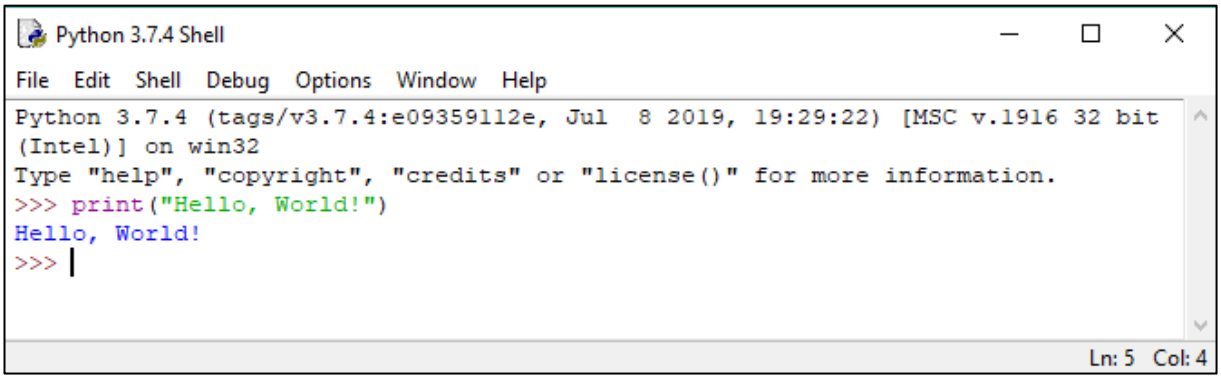
Para abrir o programa, podemos procurar pelo acrónimo na barra de pesquisa e escolher o programa com símbolo semelhante ao do lado.



Abrindo o IDLE, vemos uma janela chamada Python Shell que se assemelha ao modo interativo com algumas diferenças:

- O código que escrevemos fica colorido de acordo com o tipo de declarações que fazemos o que os torna mais fácil distinguir;
- Quando se escreve várias linhas o IDLE não mostra os … de continuação como o modo interativo faz;
- O IDLE indenta automaticamente as várias linhas.

É também possível usar o menu de ajuda escrevendo `help()`.



C2. Escrever e editar código com o editor de texto do IDLE

O IDLE inclui um editor de texto para abrir, editar e criar módulos e scripts. Algumas das características principais são:

- Criar um novo ficheiro: File -> New File
- Abrir script ou módulo já existente: File -> Open
- Num ficheiro temos o menu Format com opções para ajudar na indentação
- Para correr um programa ou módulo usamos a opção Run -> Run Module com um ficheiro aberto.

Vamos experimentar o mesmo código que usamos no *script* em cima com o IDLE.

1. Criar um novo ficheiro no IDLE e colocar o código

```
print ("teste de como os scripts funcionam")
x = 500
print ("o valor de x e", x)
```

2. Guardar o ficheiro num diretório do computador, com o sufixo `.py`.
3. Executar o programa através do menu Run->Run Module, ou usando o atalho F5.

O resultado dos *prints* do programa será escrito na *Python Shell* que aparece quando se abre o IDLE.

C3. Alguns comandos no IDLE

O IDLE tem comandos de procura (*Find*) que se encontram no menu *Edit* do editor de ficheiros. Para procurar um certo texto no ficheiro onde estamos podemos usar o *Find…* que é o equivalente a usar o atalho Ctrl+F. Também é possível procurar um certo texto num conjunto de ficheiros que estejam no caminho pré-definido do Python usando a opção *Find in Files.*
Com o *Path Browser* que se pode encontrar no menu *File* podemos analisar e abrir qualquer código Python (ficheiro com sufixo `.py`) que esteja no caminho pré-definido do Python (*Python Path*). Para navegar entre as várias pastas, basta clicar nos botões com o símbolo + e para abrir um ficheiro da lista basta clicar duas vezes no mesmo.