

Tópicos avançados em Informática I

Programação Python



Universidade Franciscana

Sumário

- ▶ Histórico
- ▶ Características
- ▶ Interpretar e interagir
- ▶ Documentação
- ▶ Conceitos básicos do Python
- ▶ Comandos de entrada e saída de dados
- ▶ Operadores Aritméticos
- ▶ Comentários
- ▶ PyCharm

Histórico

- ▶ Foi criada pelo holandês Guido Van Rossum, matemático e cientista da computação.
- ▶ Em 1989, trabalhava em um Instituto de Computação na Holanda, no projeto do sistema operacional Amoeba.
- ▶ Desenvolvimento de aplicativos que permitissem a administração do sistema operacional, a linguagem que estava sendo utilizada era C e o trabalho estava complicado, principalmente no tratamento de erros.
- ▶ Este mesmo laboratório havia desenvolvido uma linguagem de alto nível chamada ABC que tinha o propósito de facilitar o trabalho do programador.

Histórico

- ▶ Guido trabalhou entre o Natal e o ano novo.
- ▶ Experiência que tinha da linguagem ABC e desenvolveu a linguagem Python.
 - ▶ Pensando em uma linguagem que fosse fácil de aprender, de prototipar e também permitisse a extensão quando necessário.
- ▶ O nome “Python” é originário de um grupo humorístico britânico chamado *Monty Python*, e que apresentava o programa *Monty Python Flying Circus*.

Histórico

- ▶ Em 1991 Guido Van Rossum, tornou público o código da linguagem.
- ▶ Disponibilizou o código fonte e convidou outras pessoas interessadas
 - ▶ É construído colaborativamente por pessoas espalhadas pelo mundo inteiro.
- ▶ Cada programador ajuda a desenvolver a ferramenta que ele próprio gostaria de usar.
- ▶ Não existem empresas envolvidas com o desenvolvimento da linguagem.
- ▶ A interação entre as pessoas é o ponto chave no desenvolvimento da linguagem.
- ▶ A especificação da linguagem é mantida pela *Python Software Foundation* (PSF) e escrita em C.

<https://www.python.org/psf/>

Características

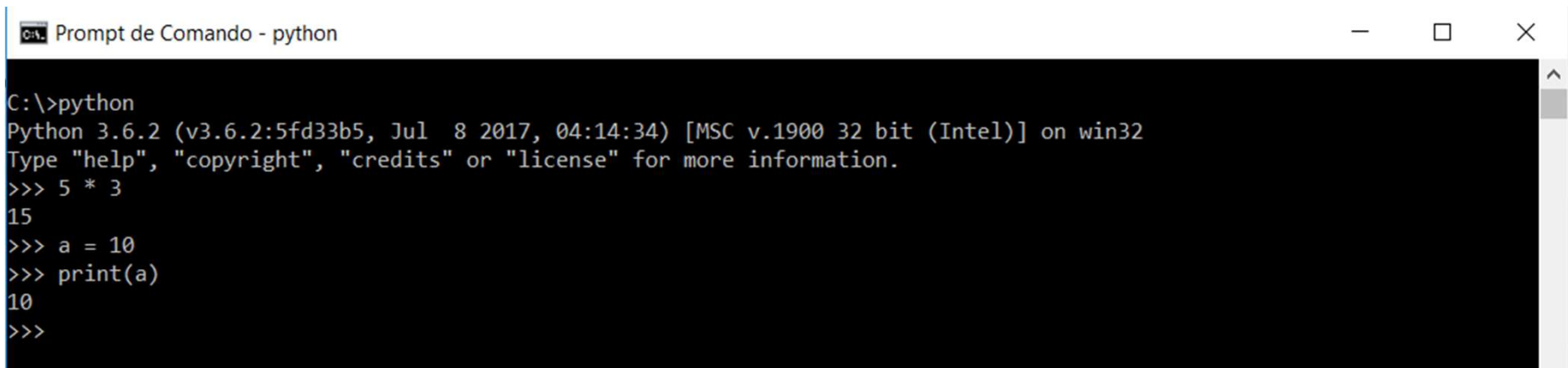
- ▶ Linguagem livre e multiplataforma
- ▶ Sintaxe simples
- ▶ Legibilidade do código
 - ▶ Indentação para marcar blocos
- ▶ Coletor de lixo para gerenciar automaticamente a memória
- ▶ Múltiplos paradigmas de programação
 - ▶ Imperativo
 - ▶ Orientado a objetos
 - ▶ Funcional

Características

- ▶ Baterias inclusas
 - ▶ Tudo o que é necessário para executar um programa está presente na instalação básica
 - ▶ Biblioteca padrão abrangente
- ▶ PyPI (*Python Package Index*), é um repositório indexado de softwares para a linguagem de programação Python
- ▶ Linguagem interpretada

Interpretar e interagir

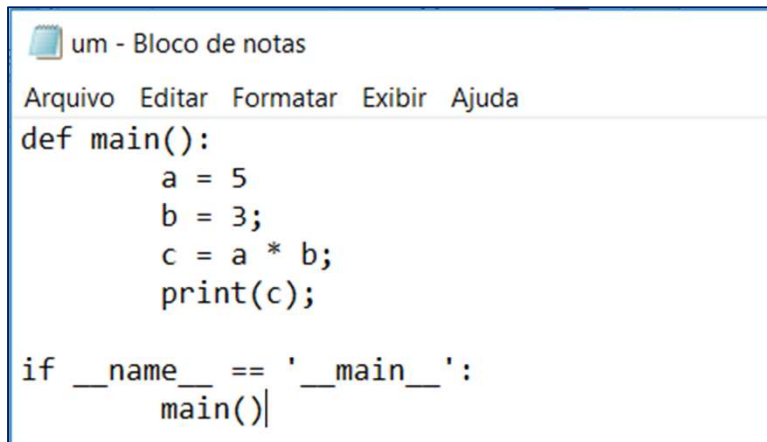
- ▶ Interpretação pode ser realizada por linha de comando ou por *script*
- ▶ Linha de comando: interpretador de forma interativa



```
C:\>python
Python 3.6.2 (v3.6.2:5fd33b5, Jul  8 2017, 04:14:34) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 5 * 3
15
>>> a = 10
>>> print(a)
10
>>>
```

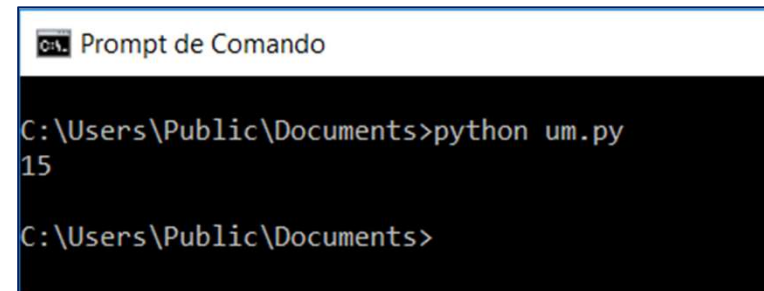

Interpretar e interagir

- ▶ Interpretação pode ser realizada por linha de comando ou por *script*
- ▶ Script
 - ▶ escrever e salvar o código Python em script com a extensão .py
 - ▶ Executar com o comando
 - ▶ Python <nome do arquivo>



```
um - Bloco de notas
Arquivo  Editar  Formatar  Exibir  Ajuda
def main():
    a = 5
    b = 3;
    c = a * b;
    print(c);

if __name__ == '__main__':
    main()
```



```
C:\> Prompt de Comando
C:\Users\Public\Documents>python um.py
15
C:\Users\Public\Documents>
```

Interpretar e interagir

- ▶ O interpretador interativo é considerado um diferencial da linguagem Python
- ▶ Permite o teste de comandos e conseqüentemente facilita o aprendizado
- ▶ O programador pode testar os comandos e verificar os resultados instantaneamente
- ▶ O Python converte o código fonte para um *bytecode*, que é um formato binário com instruções para o interpretador, não sendo necessário repetir esse processo para uma nova execução:
 - ▶ análise do código
 - ▶ conversão para linguagem de máquina (*bytecodes*)
 - ▶ execução na máquina virtual

Documentação

- ▶ Importante para todo o programador

<https://docs.python.org/3/>

- ▶ Na opção *Download*, é possível baixar toda a documentação.

Conceitos básicos de programação no Python

► Constantes

- Informações que permanecem sempre com valores fixos, (não são alteradas)
- São escritas em caixa alta

```
Python 3.6.2 (v3.6.2:5fd33b5, Jul 8 2017, 04:14:34) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> PI = 3.14159
>>> MAXIMO = 1000
>>> N = 200000
>>>
```

Conceitos básicos de programação no Python

► Variáveis

- O valor pode ser consultado, modificado e apagado
- São identificadas por um nome
- Recomendações para a nomenclatura das variáveis:
 - Ser composto apenas por letras, números e *underline*
 - Iniciar com letras
 - Escolher nomes que tenham relação com o conteúdo que será armazenado na variável
 - Quando o nome for composto por mais de uma palavra, usar o *underline* para separar as palavras
 - Não pode ser igual às palavras reservadas da linguagem.

Conceitos básicos de programação no Python

► Palavras reservadas da linguagem

```
Python 3.6.2 (v3.6.2:5fd33b5, Jul 8 2017, 04:14:34) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import keyword
>>> print(keyword.kwlist)
['False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']
>>>
```

► Variáveis

- O Python é *case-sensitive*
- Letras maiúsculas não são utilizadas na nomenclatura de variáveis.

Conceitos básicos de programação no Python

```
>>> letra$ = "Abcdefg"
      File "<stdin>", line 1
        letra$ = "Abcdefg"
              ^
SyntaxError: invalid syntax
>>> 1numero = 100
      File "<stdin>", line 1
        1numero = 100
          ^
SyntaxError: invalid syntax
>>> return = 50
      File "<stdin>", line 1
        return = 50
              ^
SyntaxError: invalid syntax
>>>
```

```
>>> letras = "Abcdefg"
>>> numero1 = 100
>>> retorno = 50
>>> print(letras)
Abcdefg
>>> print(numero1)
100
>>> print(retorno)
50
>>>
```

Conceitos básicos de programação no Python

► Atribuição

- O armazenamento de um determinado valor em uma variável se dá por intermédio da operação de atribuição
- O símbolo utilizado para representar a atribuição é o sinal de igualdade (=)
- Uma variável é criada por meio de uma atribuição
- Quando uma variável não for mais referenciada ela é desalocada pelo *garbage collector* do Python.

Tipos de dados

- ▶ O Python utiliza tipagem dinâmica, ou seja, o tipo da variável é definido pelo interpretador em tempo de execução
- ▶ Todo e qualquer valor pertence a um tipo específico de dado
 - ▶ 45 é do tipo inteiro
 - ▶ "Python é uma linguagem de programação" é uma string
 - ▶ 3.14159 é um número real
- ▶ Os tipos no Python podem ser mutáveis (permitem a alteração dos valores das variáveis) ou imutáveis (não permitem a alteração dos valores das variáveis).

Tipos de dados

► **Inteiros (*int*)**

- Números que não possuem a parte fracionária
- Podem ser negativos, positivos e zero
- Qual o maior valor inteiro que é possível armazenar? Isso vai depender da quantidade de memória da máquina.

```
>>> type(numero1)
<class 'int'>
>>> type(retorno)
<class 'int'>
>>>
```

Tipos de dados

► Ponto flutuante (real)

- Números com ponto flutuante
- São números formados pelas partes inteira e fracionária, que são separadas pelo uso do ponto (.)
- Podem ser negativos, positivos e zero
- São armazenados em 64 bits de informação

```
>>> temperatura = 12.3
>>> valor = 1234.56
>>> cientifico = 2.5e15
>>> type(temperatura)
<class 'float'>
>>> type(valor)
<class 'float'>
>>> type(cientifico)
<class 'float'>
>>>
```

Tipos de dados

► Conversão entre tipos numéricos

Conversão	Descrição
<code>int(<i>x</i>)</code>	O valor <i>x</i> é convertido para inteiro
<code>float(<i>x</i>)</code>	O valor <i>x</i> é convertido para ponto flutuante

```
>>> int(temperatura)
12
>>> int(cientifico)
25000000000000000
>>> int(valor)
1234
```

```
>>> nota = 8
>>> float(nota)
8.0
>>>
```

Tipos de dados

► Lógico ou Booleano (*bool*)

- Aceitam somente os valores verdadeiro ou falso
- Na linguagem Python a representação destes valores se dá pelas palavras-chave *True* e *False* ou então pelos valores inteiros 0 (*False*) ou 1 (*True*)

```
>>> a = True
>>> x = False
>>> type(x)
<class 'bool'>
>>> int(x)
0
>>> int(a)
1
>>> int(True)
1
>>> int(False)
0
>>> print(a)
True
>>> print(x)
False
>>>
```

Tipos de Dados

▶ **Caracteres ou Strings (*str*)**

- ▶ Armazenam textos ou sequências de caracteres
- ▶ São classificadas dentro dos tipos imutáveis, isto é, uma vez atribuído um determinado valor para esta variável este valor não poderá ser alterado
- ▶ Para modificar o conteúdo de uma string será necessário criar uma nova variável
- ▶ Seu conteúdo aparece entre aspas ("Isto é uma string") ou apóstrofes ('Isto também é uma string')
- ▶ Se uma string for inicializada com três aspas ou três apóstrofes, ela poderá ser escrita em múltiplas linhas.

Tipos de Dados

► Caracteres ou Strings (*str*)

```
>>> texto1 = "Texto definido com aspas"
>>> texto2 = 'Texto definido com apóstrofe'
>>> texto3 = """String que pode ser
... quebrada em várias linhas"""
>>> texto4 = '''String em várias linhas
... com três
... apóstrofes'''
>>> print(texto1)
Texto definido com aspas
>>> print(texto2)
Texto definido com apóstrofe
>>> print(texto3)
String que pode ser
quebrada em várias linhas
>>> print(texto4)
String em várias linhas
com três
apóstrofes
>>>
```

Comandos de Entrada e Saída

- ▶ A comunicação entre usuário e a máquina se dá por intermédio da interatividade
- ▶ Comandos de leitura e escrita são responsáveis pelas operações de entrada e saída de dados
- ▶ A saída de dados exibe alguma mensagem na tela do computador.
- ▶ Na linguagem Python o comando responsável por essa operação é o **print**

Caractere especial	Tipo de dado que representa
%d ou %i	Valores inteiros
%f	Valores de ponto flutuante
%c	Um caractere
%s	Strings

Comandos de Entrada e Saída

```
>>> PI = 3.14159
>>> coord_x = 4
>>> coord_y = 10
>>> area = 23.45
>>> print("As coordenadas x e y são %d %d" %(coord_x, coord_y))
As coordenadas x e y são 4 10
>>> print("O valor aproximado de PI é %f" %PI)
O valor aproximado de PI é 3.141590
```

Comandos de Entrada e Saída

- ▶ As operações de leitura representam a entrada de dados
- ▶ Geralmente esses dados são informados por intermédio do teclado
- ▶ Na linguagem Python a função padrão **input("Mensagem")** ao ser inserida no código solicita uma informação ao usuário e fica aguardando a digitação desta informação
- ▶ A mensagem corresponde ao texto que será exibido ao usuário
- ▶ As informações precisam ficar armazenadas em variáveis, assim, na maioria das vezes, será feita uma atribuição de variável à função **input("Mensagem")**.

Comandos de Entrada e Saída

```
>>> nome = input("Qual o seu nome?")
Qual o seu nome?Francisco
>>> idade = input("Qual sua idade?")
Qual sua idade?6
>>>
```

```
>>> print("Seu nome é %s e você tem %d anos" %(nome, idade))
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: %d format: a number is required, not str
>>> print("Seu nome é %s e você tem %d anos" %(nome, int(idade)))
Seu nome é Francisco e você tem 6 anos
```

Operadores Aritméticos

Símbolo	Operação	Prioridade
**	Exponenciação	1
*	Multiplicação	2
/	Divisão	2
//	Divisão inteira	2
%	Resto da divisão	2
+	Adição	3
-	Subtração	3

```
>>> 8 + 2 * 3
14
>>> 4 ** 2
16
>>> 8 / 3
2.6666666666666665
>>> 8 // 3
2
>>> 8 % 3
2
>>> 8 / 5
1.6
>>> 8 % 5
3
>>>
```

Comentários

- ▶ É uma prática que deve ser transformada em hábito
- ▶ Os comentários fazem parte da documentação do software
- ▶ Auxiliam no entendimento da lógica empregada na solução problema em questão
- ▶ Explicam o que os comandos fazem e como foram empregados
- ▶ Cabe ao programador decidir quando e onde fazer os comentários. Geralmente comentam-se as partes mais críticas, aquelas que exigiram mais tempo e dedicação
- ▶ # comentário de uma única linha
- ▶ """ comentários com mais de uma linha
- ▶ Os comentários só são vistos por quem analisar o código fonte.

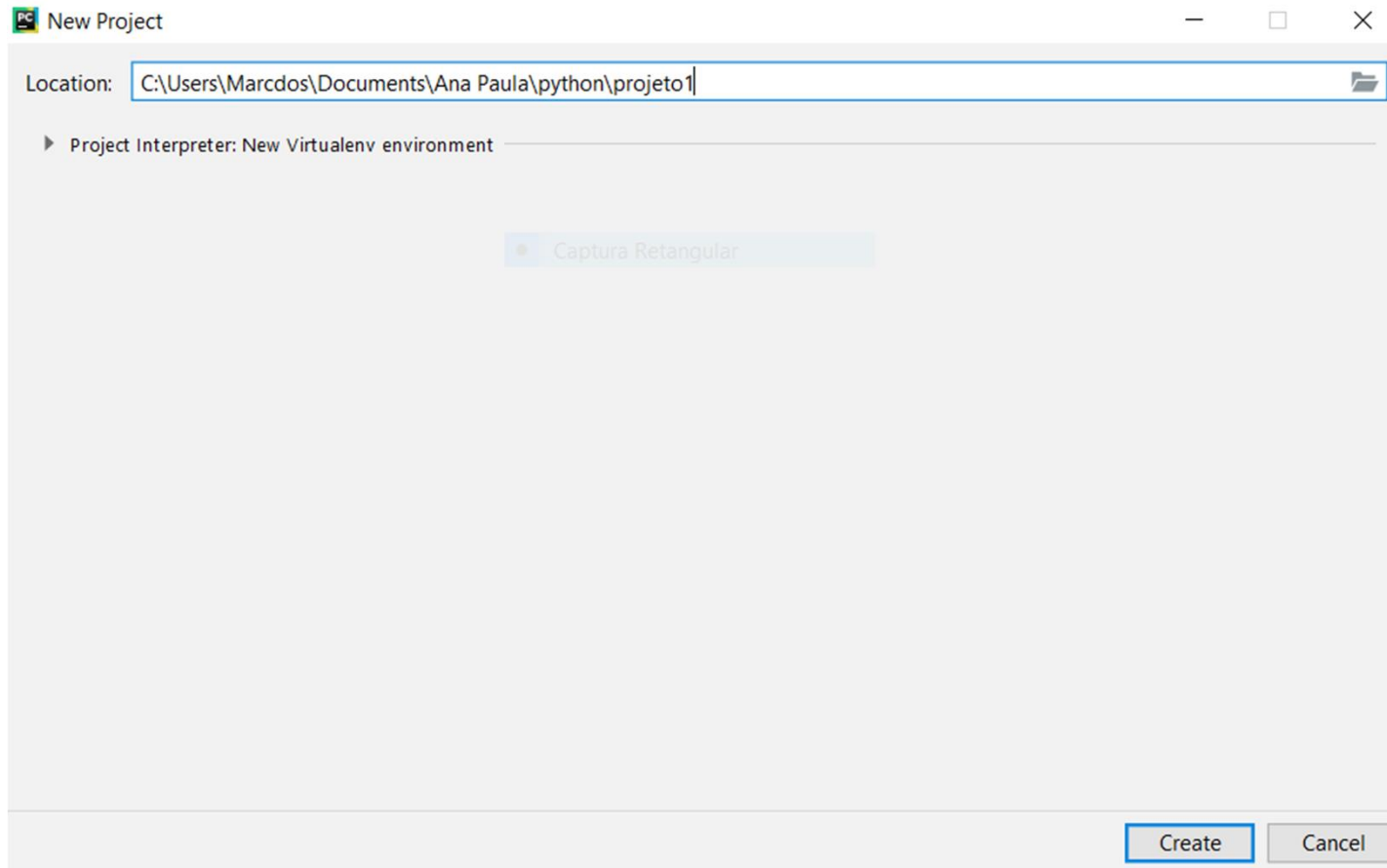
PyCharm

- ▶ É uma IDE (*Integrated Development Environment*), um editor específico para a escrita de programas de computador
- ▶ São ferramentas que auxiliam e agilizam o desenvolvimento de software, pois possuem incorporadas no mesmo ambiente vários recursos
 - ▶ Editor específico para a linguagem com recurso de auto completar o código
 - ▶ Compilador e/ou interpretador da linguagem
 - ▶ Depurador para auxiliar na correção de erros.

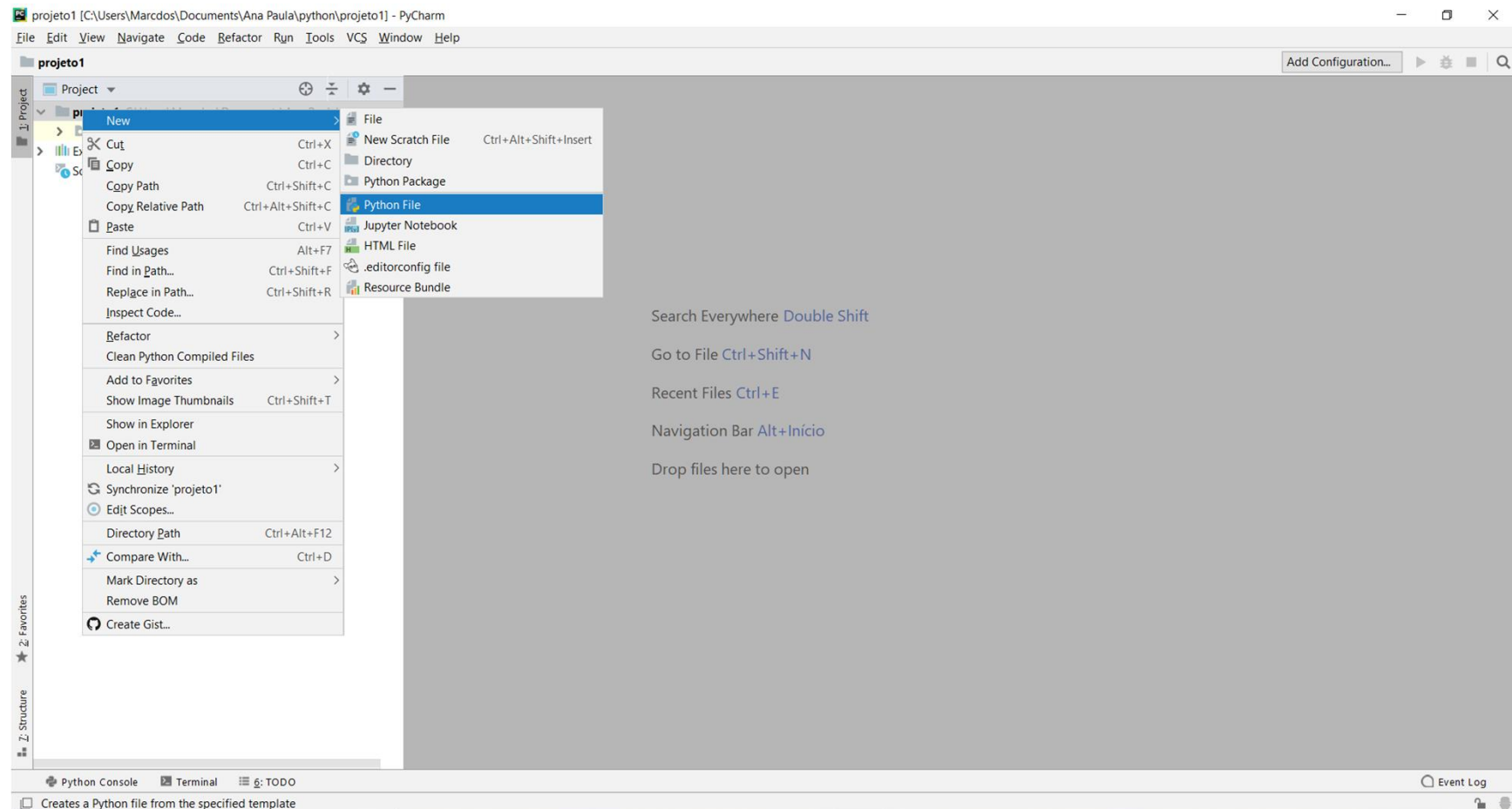
PyCharm

- ▶ PyCharm – JetBrains
- ▶ Três tipos de licenças disponíveis:
 - ▶ *Professional Edition* (versão paga e com o maior número de recursos)
 - ▶ *Community Edition*
 - ▶ *Educational Edition* (estudantes e professores)

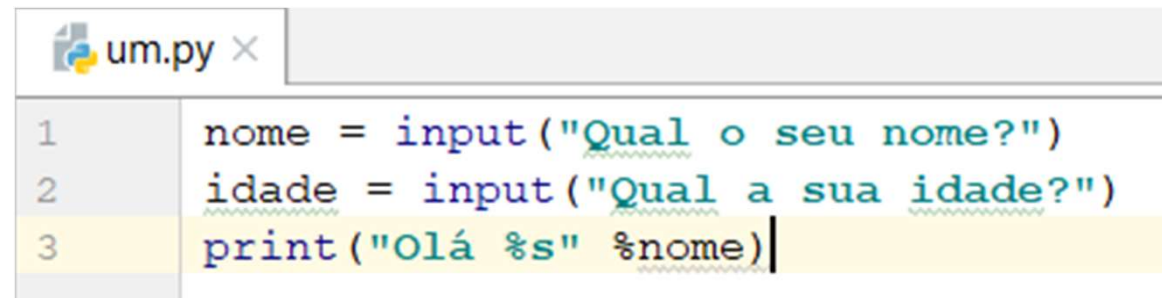
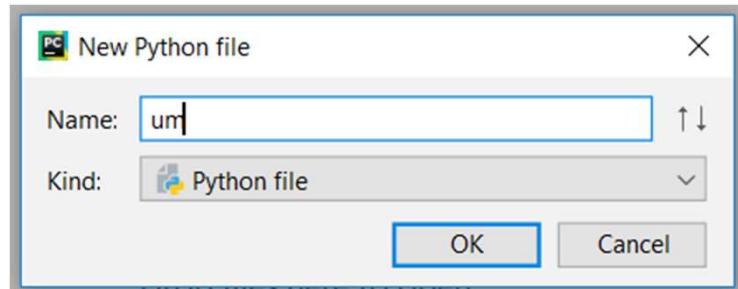
PyCharm



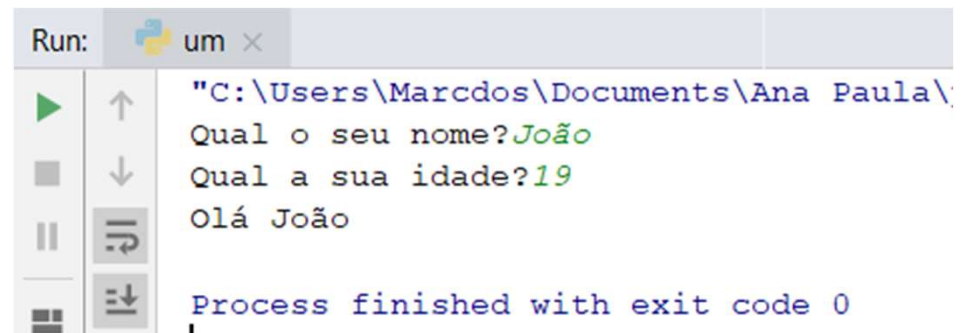
PyCharm



PyCharm



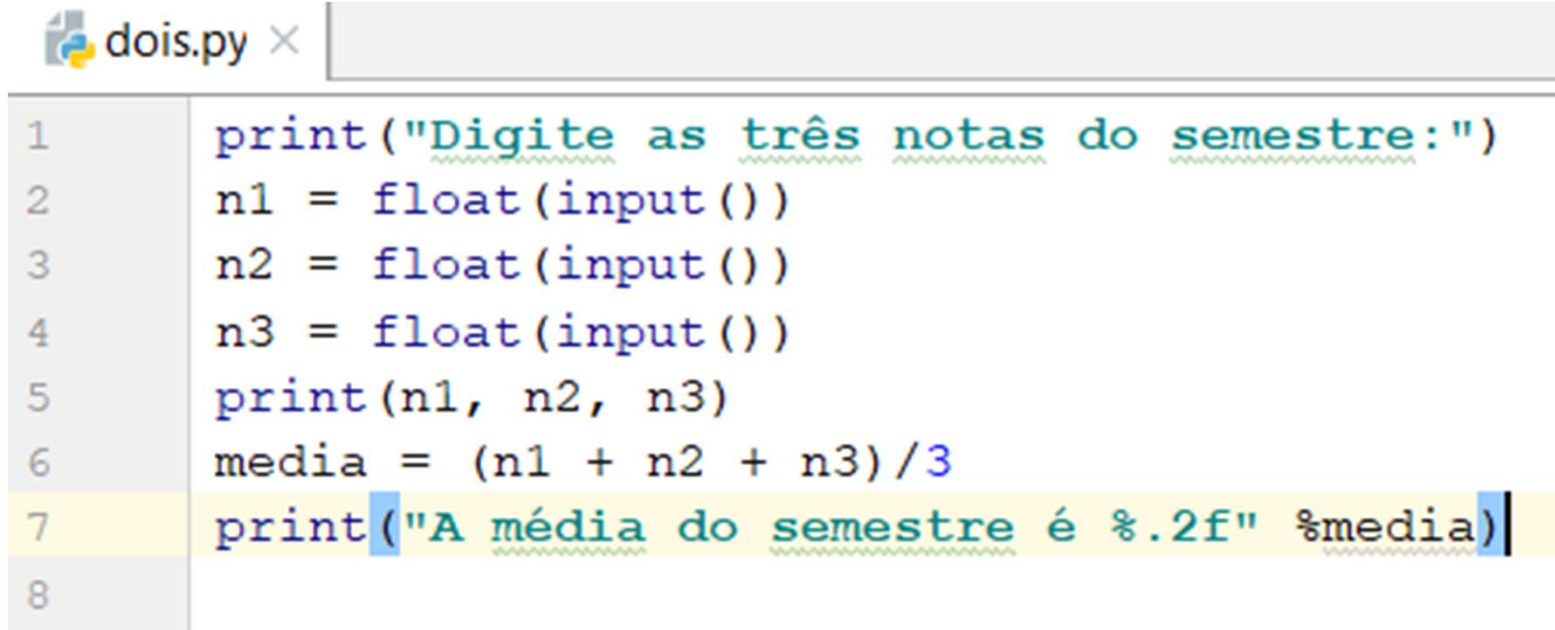
```
1 nome = input("Qual o seu nome?")
2 idade = input("Qual a sua idade?")
3 print("Olá %s" % nome)
```



```
Run: um x
"C:\Users\Marcos\Documents\Ana Paula\
Qual o seu nome?João
Qual a sua idade?19
Olá João

Process finished with exit code 0
```

Exemplo



The image shows a screenshot of a code editor window titled "dois.py". The editor contains a Python script with 8 lines of code. Line 7 is highlighted in yellow. The code calculates the average of three input grades and prints the result with two decimal places.

```
1 print("Digite as três notas do semestre:")
2 n1 = float(input())
3 n2 = float(input())
4 n3 = float(input())
5 print(n1, n2, n3)
6 media = (n1 + n2 + n3)/3
7 print("A média do semestre é %.2f" %media)
8
```

Alguns *links*

- ▶ Site oficial da linguagem:

<http://www.python.org/>.

- ▶ Site oficial da comunidade brasileira:

<http://www.pythonbrasil.com.br/>

- ▶ Perguntas frequentes:

<http://www.pythonbrasil.com.br/moin.cgi/PerguntasFrequentes/SobrePython>

- ▶ PyPI (*Python Package Index*):

<https://pypi.org/>