

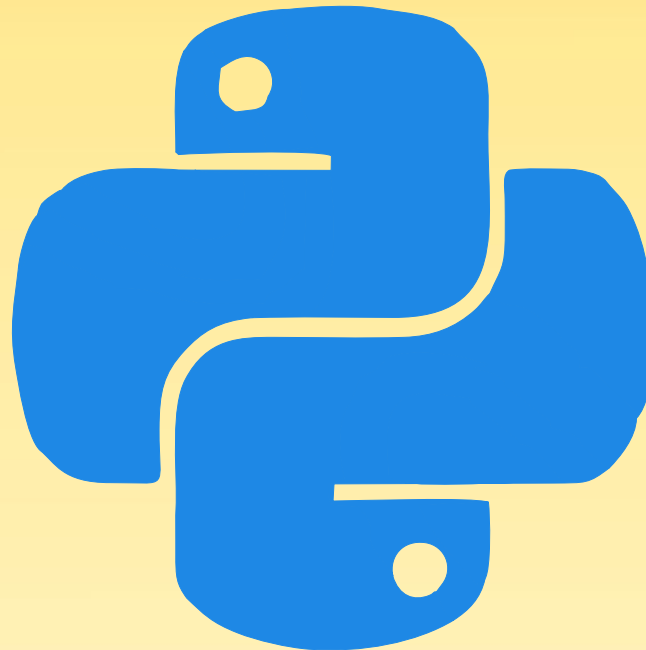


---

# MINICURSO PYTHON 1.0

---

Slide-aula minicurso 1.0



2020

PROGRAMA DE EDUCAÇÃO TUTORIAL – ENGENHARIA ELÉTRICA –  
UNIVERSIDADE FEDERAL DE MINAS GERAIS



---

# Introdução

---



# História

---

Monty Python's Flying Circus,

Guido van Rossum,

1991



# Motivação

---

Alto nível,  
tipagem dinâmica,  
propósito geral,  
várias aplicações,  
orientada a objetos,  
interpretada.



# Motivação

---

```
import this
"""The Zen of Python, by Tim Peters. (poster by Joachim Jablon)"""

1 Beautiful is better than ugly.
2 Explicit is better than impl..
3 Simple is better than complex.
4 Complex is better than cOmp1|c@ted.
5 Flat is better than nested.
6 Sparse is better than dense.
7 Readability counts.
8 Special cases aren't special enough to break the rules.
9 Although practicality beats purity.
10 raise PythonicError("Errors should never pass silently.")
11 # Unless explicitly silenced.
12 In the face of ambiguity, refuse the temptation to guess.
13 There should be one-- and preferably only one --obvious way to do it.
14 # Although that way may not be obvious at first unless you're Dutch.
15 Now is better than ... never.
16 Although never is often better than rightnow.
17 If the implementation is hard to explain, it's a bad idea.
18 If the implementation is easy to explain, it may be a good idea.
19 Namespaces are one honking great idea -- let's do more of those!
```





---

# Ambiente

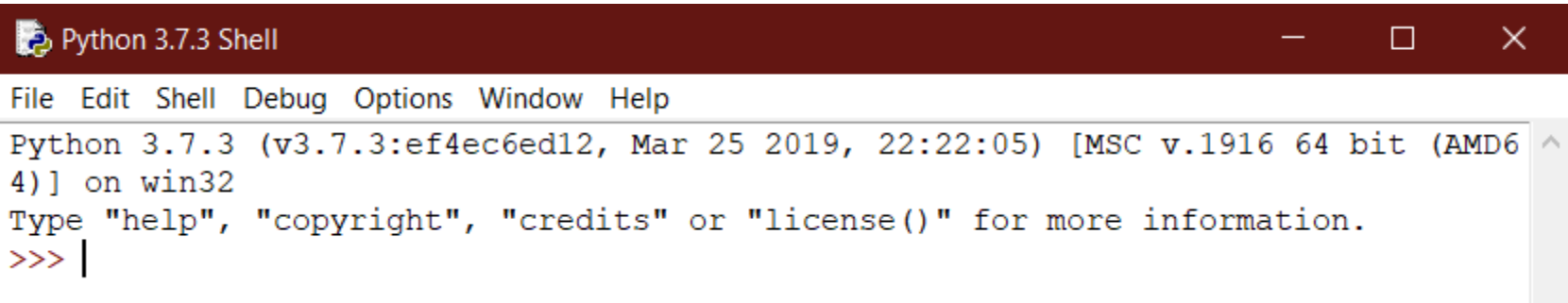
---



# Shell vs Script

---

## Python Shell

A screenshot of a Python 3.7.3 Shell window. The window title bar is dark red and contains the text "Python 3.7.3 Shell" and standard window control buttons (minimize, maximize, close). The main content area is white and displays the following text:

```
File Edit Shell Debug Options Window Help
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 22:22:05) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> |
```



# Shell vs Script

---

## Python Shell

```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 22:22:05) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> |
```

a = 3

b = 2

a + b

print('Ola Mundo')

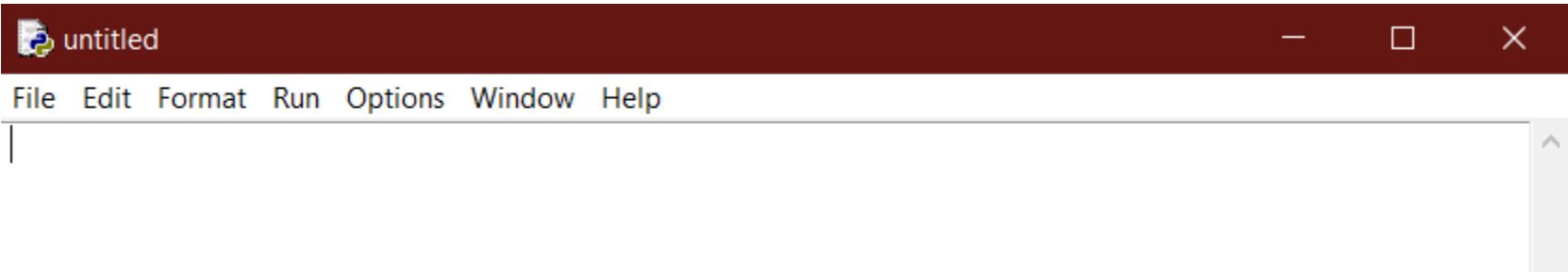




# Shell vs Script

---

## Python Script



# Shell vs Script

---

## Python Script



Salve e rode (F5)



# Sumário

---

## Parte II – Conceitos Básicos

3 - Estrutura de Dados

4 - Comandos e Funções

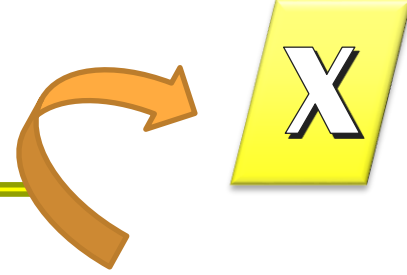
5 - Operadores

6 - Controle de Fluxo



# Dicas

---



Número do capítulo da apostila

Nomeie e salve os programas  
que você fará durante o curso





---

# Estrutura de Dados

---



Armazenamento

Nomenclatura

Tipos



# Tipos de Dados

---

3

Tipos de Dados → Variáveis

Tipos primitivos:

Ponto Flutuante

Inteiro

Lógico

Caractere



# Tipos de Dados

---

Tipagem dinâmica:

Não é necessário declarar variáveis

O tipo já é dinamicamente atribuído

Comando **type()** informa o tipo da variável.





# Tipos do Python

---

3

Ponto Flutuante

float

Inteiro

int

Lógico

bool

Caractere

str

OBS: Atenção para o tipo **str**. O veremos depois com mais detalhes



# Tipos do Python

---

3

Tipos → Classes.

Outros tipos:

complex

list

tuple

dict



# Tipos do Python

---

Tipagem dinâmica

```
>>> Frase = 'Esta e uma frase'
>>> type(Frase)
<class 'str'>
>>> a = 4
>>> type(a)
<class 'int'>
>>> x = 2.5
>>> type(x)
<class 'float'>
>>> Q = True
>>> type(Q)
<class 'bool'>
>>> s = 1 + 2j
>>> type(s)
<class 'complex'>
```



# Tipos do Python

---

3

```
>>> Frase = 'Esta e uma frase'
>>> type(Frase)
<class 'str'>
>>> a = 4
>>> type(a)
<class 'int'>
>>> x = 2.5
>>> type(x)
<class 'float'>
>>> Q = True
>>> type(Q)
<class 'bool'>
>>> s = 1 + 2j
>>> type(s)
<class 'complex'>
```

Criem uma variável de cada tipo para testar



# Tipos do Python

3

Conversão:

```
>>> x = float(3)
>>> a = str(x)
>>> x = x + 1
>>> a
'3.0'
>>> print('Valor = ' + a)
Valor = 3.0
>>> print('Valor = ', x)
Valor = 4.0
```

OBS: a função print  
será explicada  
depois

Bases de valores numéricos:

```
>>> d = 31 #base decimal
>>> d
31
>>> b = 0b11111 #base binaria
>>> b
31
>>> o = 0o37 #base octal
>>> o
31
>>> h = 0x1F #base hexadecimal
>>> h
31
```



# Exercício

---

3

Faça  $a + b$  dar o mesmo resultado de  $c + d$ :

```
>>> a = 1; b = 2; c = '1'; d = '2'
```



# Exercício

---

3

Faça `a + b` dar o mesmo resultado de `c + d`:

```
>>> a = 1; b = 2; c = '1'; d = '2'
```

```
>>> a + b; c + d
```

```
3
```

```
'12'
```

```
>>> str(a) + str(b); int(c) + int(d)
```

```
'12'
```

```
3
```





---

# Comandos e Funções

---





# Comandos e Funções

---

4

Funções:

Chamadas

Parâmetros

Retorno

```
retorno1, retorno2 = função(parâmetro1, parâmetro2)
```

```
>>> x = sqrt(25)
```

A variável *x* receberá o valor 5.



# Importação

---

4

```
>>> import math
>>> sqrt(9)
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
NameError: name 'sqrt' is not defined
>>> math.sqrt(9)
3.0
```

Importação Otimizada:

```
>>> from math import cos, pi
>>> cos(pi)
-1.0
```



# Importação

4

Find, install and publish Python packages  
with the Python Package Index

Search projects



Or [browse projects](#)

189,305 projects

1,399,877 releases

2,041,810 files

351,166 users

<https://pypi.org/>



# Exercício Para Casa\*\*

---

4

\*\*Precisa de Internet

Baixe as seguintes bibliotecas para importarmos no próximo dia do curso:

opencv-python

numpy

matplotlib



# Exercício Para Casa\*\*

4

Modo 1:

Este Computador > DATA (D:) > Arquivos de Programas > Python

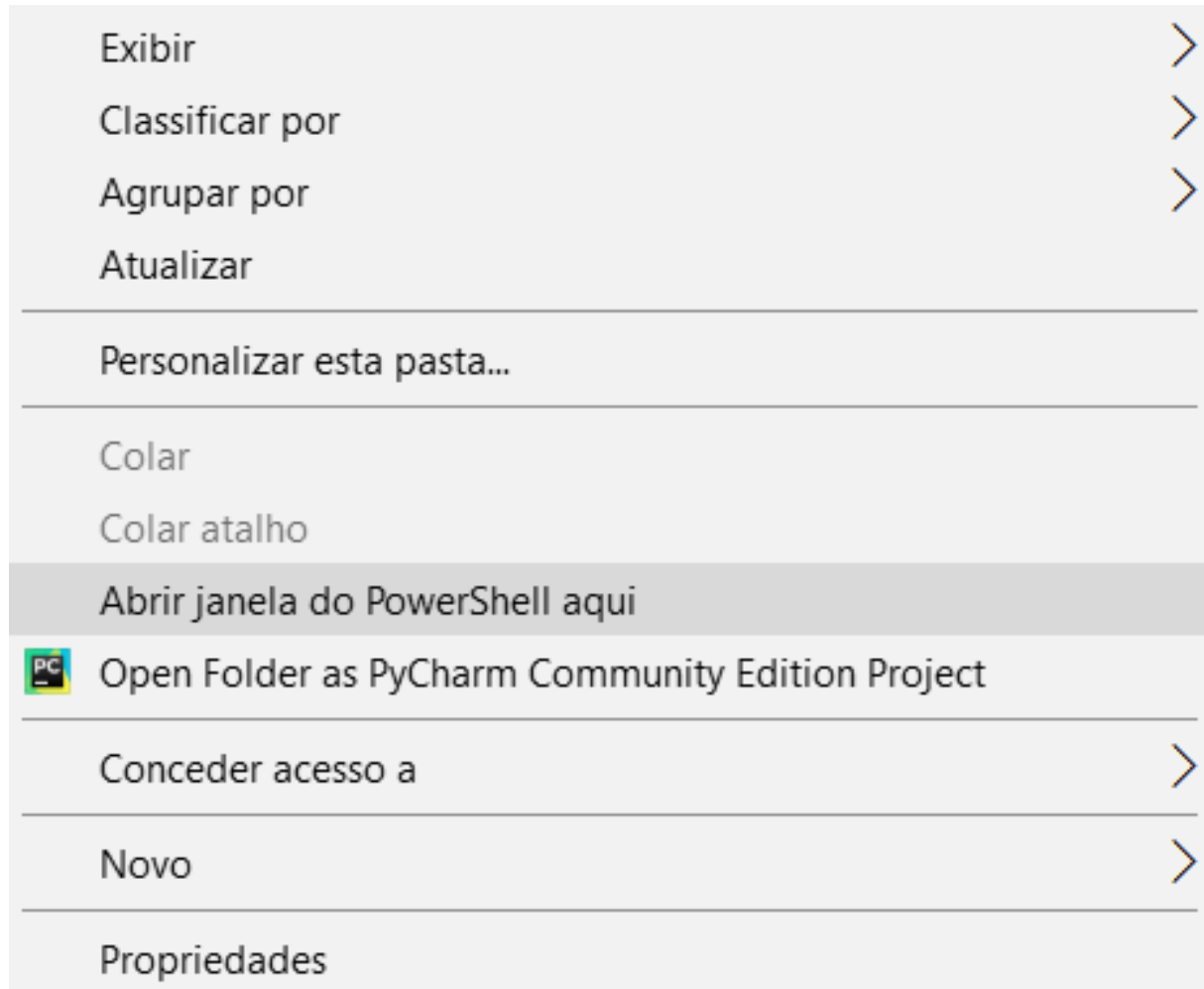
	Nome	Data de modificaç...	Tipo	Tamanho
	^			
	DLLs	17/06/2019 13:31	Pasta de arquivos	
	Doc	17/06/2019 13:31	Pasta de arquivos	
	include	08/07/2019 10:53	Pasta de arquivos	
	Lib	17/06/2019 13:31	Pasta de arquivos	
	libs	17/06/2019 13:31	Pasta de arquivos	
	Scripts	17/06/2019 13:31	Pasta de arquivos	
	tcl	17/06/2019 13:31	Pasta de arquivos	
	Tools	17/06/2019 13:31	Pasta de arquivos	
	LICENSE.txt	25/03/2019 22:26	Documento de Te...	30 KB
	NEWS.txt	25/03/2019 22:26	Documento de Te...	650 KB
	python.exe	25/03/2019 22:23	Aplicativo	98 KB
	python3.dll	25/03/2019 22:22	Extensão de aplica...	58 KB
	python37.dll	25/03/2019 22:22	Extensão de aplica...	3.661 KB
	pythonw.exe	25/03/2019 22:23	Aplicativo	97 KB
	vcruntime140.dll	25/03/2019 21:22	Extensão de aplica...	88 KB



# Exercício Para Casa\*\*

4

Modo 1: Shift + botão direito



# Exercício Para Casa\*\*

4

Modo 1:

```
PS D:\Arquivos de Programas\Python> pip install pygame
Collecting pygame
  Using cached https://files.pythonhosted.org/packages/ed/56/b63ab3724acf
f69f4080e54c4bc5f55d1fbdeeb19b92b70acf45e88a5908/pygame-1.9.6-cp37-cp37m-
win_amd64.whl
Installing collected packages: pygame
Successfully installed pygame-1.9.6
You are using pip version 19.0.3, however version 19.2.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pi
p' command.
PS D:\Arquivos de Programas\Python>
```

```
pip install < bib >
```



# Exercício Para Casa\*\*

---

4

Modo 2:

```
python -m pip install < bib >
```

Em qualquer diretório

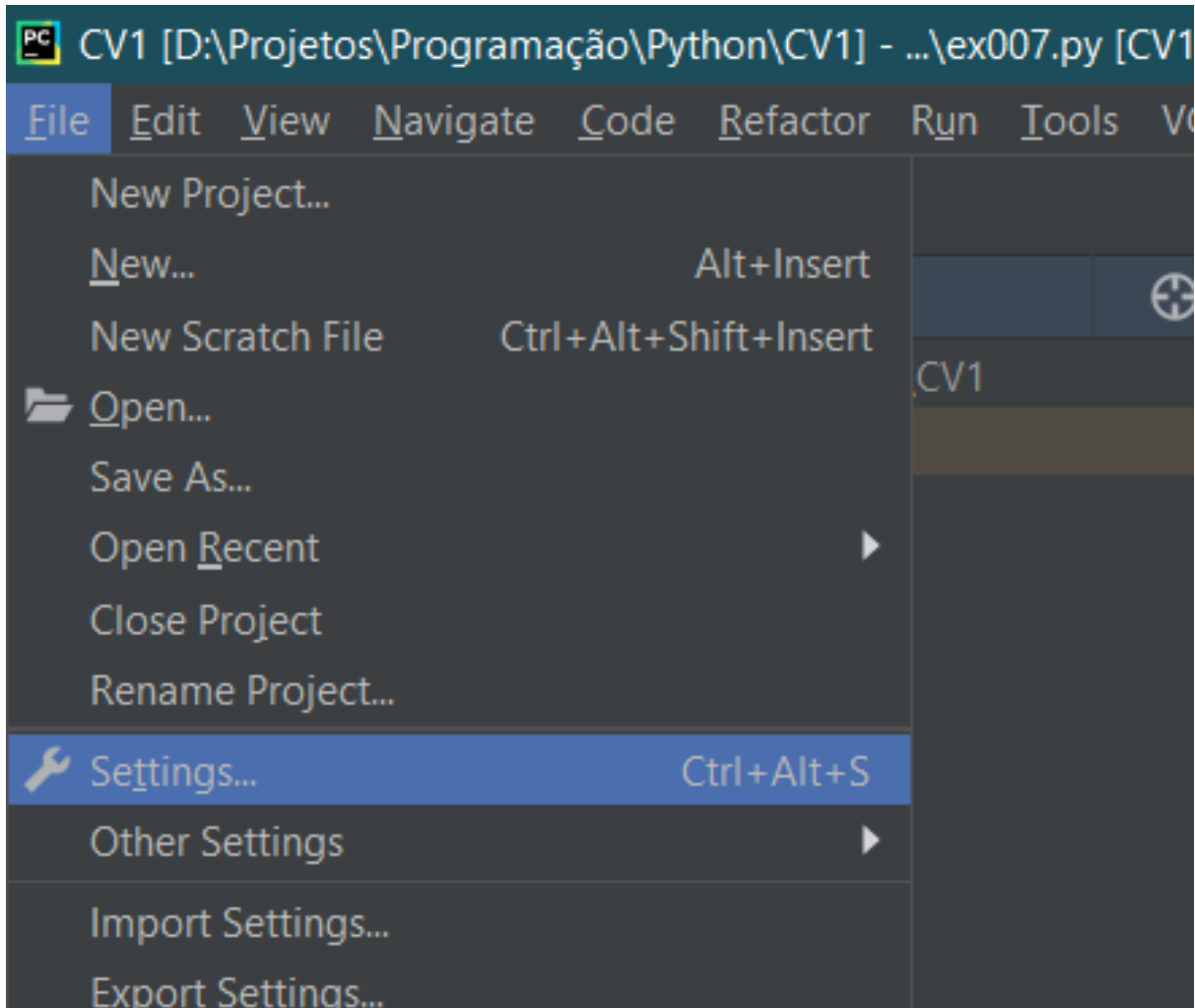




# Exercício Para Casa\*\*

4

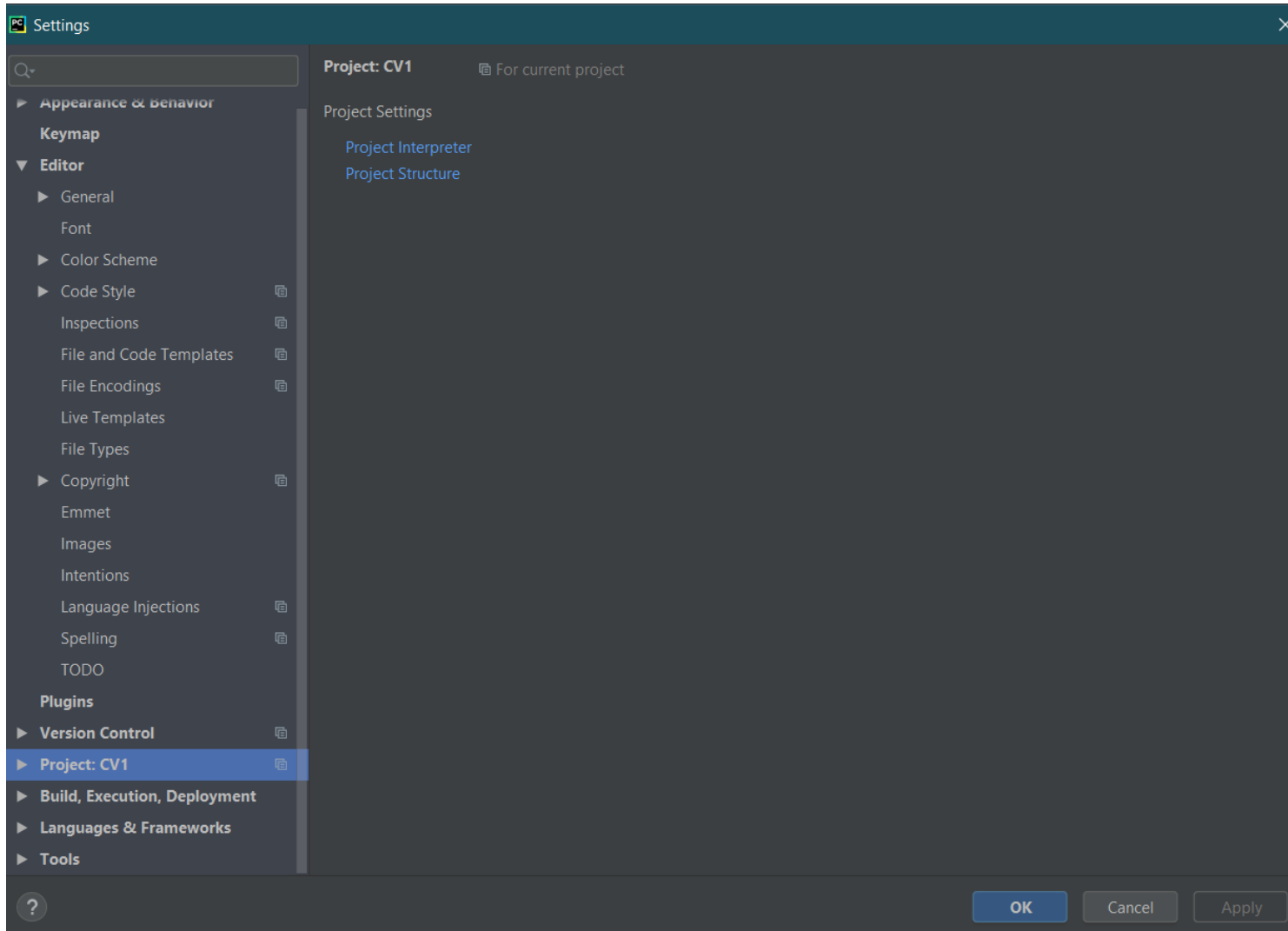
Modo 3:



# Exercício Para Casa\*\*

4

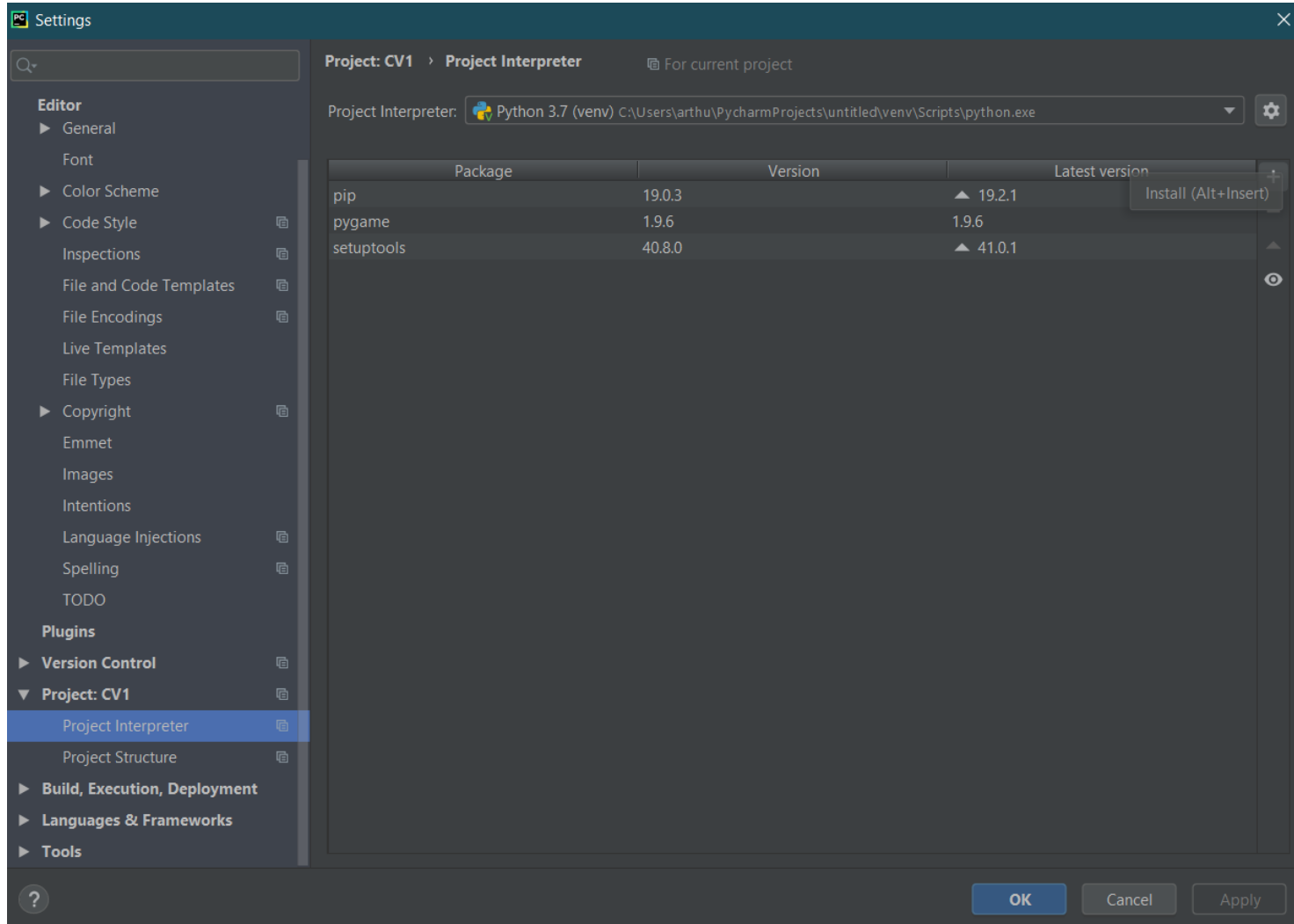
Modo 3:



# Exercício Para Casa\*\*

4

Modo 3:



Settings

Project: CV1 > Project Interpreter For current project

Project Interpreter: Python 3.7 (venv) C:\Users\arthur\PycharmProjects\untitled\venv\Scripts\python.exe

Package	Version	Latest version
pip	19.0.3	▲ 19.2.1
pygame	1.9.6	1.9.6
setuptools	40.8.0	▲ 41.0.1

Install (Alt+Insert)

OK Cancel Apply



# Entrada de Dados

---

4

Input():

```
>>> frase = input('digite uma frase\n')
digite uma frase
Ola
>>> frase
'Ola'
```

Nesta função, mesmo que o usuário digite valores numéricos, os dados serão armazenados como strings. Para usá-los como outros tipos de dados, deve convertê-los.

```
>>> x = int( input('digite um valor\n') )
digite um valor
2
>>> x
2
```



# Saída de Dados

print():

```
>>> A = 'Uma frase'
```

```
>>> B = 'separada'
```

```
>>> x = 123
```

separator

```
>>> print(A, B, x)
```

```
Uma frase separada 123
```

end

```
>>> print(A, B, x, sep='-')
```

```
Uma frase-separada-123
```

```
>>> print(A, B, x, end='nada')
```

Impressão

```
Uma frase separada 123nada>>> print(A, B, x, sep='—', end  
= '\n fim\n')
```

De variáveis

```
Uma frase—separada—123
```

```
 fim
```

```
>>>
```





---

# Operadores

---



# Aritméticos

---

5

Comuns: + - \* /

Especiais:

// Divisão Inteira

% Resto da divisão

\*\* Potenciação



# Lógicos

---

5

0 -> Falso  
1 (qualquer valor diferente de 0) -> Verdadeiro

and e

not não

or ou





== != <> > >= < <=

!= e <> representam o mesmo operador

Atenção para não confundir == com =



# Atribuição

---

5

Atribuição: =

Aritmético + Atribuição

```
>>> x = 4; y = 2
```

```
>>> x += y #equivale a x = x + y
```

```
>>> x
```

```
6
```



>>> 1 & 2

0

>>> 2^1

3

>>> (~2 & 1) | (2 & ~1)

3

>>> 8>>1

4

>>> 2<<3

16

Considera a representação binária

Operador	Descrição
&	E
	Ou
^	Ou exclusivo
~	Complemento de um
<<	Deslocamento à esquerda
>>	Deslocamento à direita



# Filiação e Identidade

---

5

```
>>> x = 1
>>> x in [0, 1, 2]
True
>>> x in [0, 2, 3, 4]
False
>>> y = 1
>>> x is y
True
>>> x is 1
True
```

Serão úteis mais tarde



# Precedência

5

- 1) () Parênteses
- 2) \*\* Potenciação
- 3) ~ + - Complemento, positivo e negativo (unários)
- 4) \* / % // Multiplicação e divisão
- 5) + - Adição e subtração
- 6) >> << Deslocamento binário
- 7) & E binário
- 8) ^ | Não binário e ou binários
- 9) <= < > >= Maior e menor que
- 10) <> == != Igual a e diferente de
- 11) = %= /= // = -= += \*= \*\* =  
Atribuição
- 12) is Identidade
- 13) in Filiação
- 14) not or and Relacionais



# Exercício

---

5

Faça um programa que receba um número  $R$ , em reais, e calcule a quantidade mínima de notas necessárias para dar a quantia. Notas disponíveis: 100, 50, 10 e 1.



# Exercício

5

Faça um programa que receba um número R, em reais, e calcule a quantidade mínima de notas necessárias para dar a quantia. Notas disponíveis: 100, 50, 10 e 1.

```
n = int(input('Digite o valor que deseja
receber:\n'))

nota_100 = n//100
n = n%100
nota_50 = n//50
n = n%50
nota_10 = n//10
n = n%10
nota_1 = n//1
n = n%1

print('Notas de 100,00 reais:', nota_100)
print('Notas de 50,00 reais:', nota_50)
print('Notas de 10,00 reais:', nota_10)
print('Notas de 1,00 real:', nota_1)
```





---

# Controle de Fluxo

---





# Expressão

---

6

Identação, “:”

Subordinação

Aninhamento

ESTRUTURA <expressão> :

procedimento 1

procedimento 2

...



# Condicional - If

---

6

```
1 x = int(input('digite um valor: '))
2 if x>10:
3     print('x maior que 10')
```



# Condicional - If

---

6

```
1 x = int(input('digite um valor: '))
2 if x>10:
3     print('x maior que 10')
4 elif x<=10 and x>=5:
5     print('x entre 5 e 10')
6 else:
7     print('x menor que 5')
```



# Repetição - While

---

6

```
1 i = 0
2 while i < 10:
3     print(i)
4     i += 1
```

Se a condição for sempre verdadeira, o loop é infinito



# Repetição - For

---

6

Ao invés de avaliar uma condição como o while, for itera uma variável

```
1 for i in [0, 'a', 2, 5, 1]:
2     print(i)
```

Neste caso, serão impressos os valores, na ordem que aparecem: 0, a, 2, 5 e 1.

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

Neste caso, serão impressos os valores: 0, 1, 2, 3, 4, 5, 6, 7, 8 e 9.



# Desvio Incondicional

---

6

**break:** Sai do laço

**continue:** Passa para a próxima iteração

**pass:** Passe



# Desvio Incondicional

---

6

```
>>> from random import randint
>>> while True:
    x = randint(0, 10)
    print(x)
    if x == 5:
        break
```

```
>>> for x in range(0, 10):
    if x == 5:
        continue
    print(x)
```



# Exercício

---

6

Faça um programa que receba um número  $n$  e imprima os  $n$  primeiros números da sequência de fibonacci





# Exercício

6

Faça um programa que receba um número  $n$  e imprima os  $n$  primeiros números da sequência de fibonacci

```
n = int(input('Ate qual numero da sequencia de
fibonacci deseja ver?\n'))
f = 0
prev_1 = 1
prev_2 = 0
print('Indice 1 f = 1')
for i in range(2, n+1):
    f = prev_1 + prev_2
    prev_2 = prev_1
    prev_1 = f
    print('Indice', i, 'f =', f)
```



# Sumário

---

## Parte III - Conceitos Avançados

7 - Estrutura de Dados II

8 - Comandos e Funções II

9 - Controle de Fluxo II

10 - Orientação a Objetos: Introdução





---

# Estrutura de Dados II

---



# Estruturas Compostas

---

7

Compor estruturas dados primitivos e armazená-las outras estruturas

```
type()  
dir()
```



Ordenadas – itens indexados (a partir do 0)

Mutáveis – itens podem mudar

Heterogêneas – itens de tipos variados

```
>>> L = [1, 4.5, 'abc', 3 + 4j, [1, 2, 3, 4]]
>>> L[2] = 5
>>> L[4] = 5
>>> L
[1, 4.5, 5, (3+4j), 5]
```

OBS: Índices negativos(a partir do -1)  
percorrem a lista de trás para frente.



Operador “+” equivale ao método append().

```
>>> A = [1, 2, 3]; B = [4, 5, 6]
>>> C = A + B
>>> C
[1, 2, 3, 4, 5, 6]
>>> C += [7] #equivale a C = C + [7]
>>> C
[1, 2, 3, 4, 5, 6, 7]
>>> C.append('abc')
>>> C
[1, 2, 3, 4, 5, 6, 7, 'abc']
```

Listas também podem armazenar outras listas



Alguns operadores aritméticos e relacionais podem ser usados.

+ concatenação

\* múltiplas concatenações

== != <> < <= > >= comparação



Funções (métodos):

**append(A)** Adiciona A no fim da lista;

**clear()** Limpa a lista;

**count(A)** Conta quantos A;

**index(A)** (Primeiro) Índice que tem A;

**insert(i, A)** Insere A na posição i;

**pop(i)** Retorna e remove o item na posição i;

**remove(A)** Remove A;

**reverse()** Inverte a lista;

**sort()** Ordena a lista;





# Listas

---

7

Laço for:

for i in lista:

NovaLista = [ Elemento for Variavel in  
VelhaLista if Condicao ]



# Listas

7

```
>>> A = [0, 0, 5, 3, 2, 7, 4, 10, 4, 7]
>>> N = list()
>>> for i in A:
    if i%2 == 0:
        N += [i*i]
```

```
>>> N
[0, 0, 4, 16, 100, 16]
```

Este exemplo atribui a N os valores de A ao quadrado, mas apenas os que são pares.

```
>>> A = [0, 0, 5, 3, 2, 7, 4, 10, 4, 7]
>>> N = [i*i for i in A if i%2 ==0]
>>> N
[0, 0, 4, 16, 100, 16]
```



Fatiamento:

Lista[ início : fim : passo ]

Padrão: início=0; passo=1.

OBS: fim-1

OBS: Valem índices e passos negativos.



Fatiamento:

```
>>> X = [1, 2, 3, 4, 5, 6, 7, 8, 9, 0]
>>> X[:2:1]
[1, 2]
>>> X[:4]
[1, 2, 3, 4]
>>> X[::2]
[1, 3, 5, 7, 9]
>>> X[:] #equivale a X
[1, 2, 3, 4, 5, 6, 7, 8, 9, 0]
```



# Tuplas

---

7

Ordenadas

Imutáveis

Heterogêneas

Podem ser usadas normalmente em iterações for.



Tuplas são imutáveis, mas podem armazenar listas, que são mutáveis.

```
>>> x = ([0, 1, 2], 2, 2)
```

```
>>> x[1]
```

```
2
```

```
>>> x[1] = 2
```

```
Traceback (most recent call last):
```

```
  File "<pyshell#12>", line 1, in <module>
```

```
    x[1] = 2
```

```
TypeError: 'tuple' object does not support item assignment
```

```
>>> x[0]
```

```
[0, 1, 2]
```

```
>>> x[0][0] = 'mudei'
```

```
>>> x
```

```
(['mudei', 1, 2], 2, 2)
```



Alguns operadores aritméticos e relacionais podem ser usados.

+ concatenação

\* múltiplas concatenações

== != <> < <= > >= comparação



Alguns operadores aritméticos e relacionais podem ser usados.

+ concatenação

\* múltiplas concatenações

== != <> < <= > >= comparação





# Tuplas

7

```
>>> X = (1, 2, 3)
```

```
>>> Y = (1, 2, 4)
```

```
>>> Z = (4, 0, 0)
```

```
>>> X < Y
```

```
True
```

```
>>> X < Z
```

```
True
```

```
>>> X + Y
```

```
(1, 2, 3, 1, 2, 4)
```

```
>>> Z*4
```

```
(4, 0, 0, 4, 0, 0, 4, 0, 0, 4, 0, 0)
```

```
>>> X += (4,)
```

```
>>> X
```

```
(1, 2, 3, 4)
```

Não há quebra da imutabilidade



# Tuplas

---

7

Funções (métodos):

**count(A)** Conta quantos A;

**index(A)** (Primeiro) Índice que tem A;



# Dicionários

---

7

Não ordenados, “etiquetados”

Mutáveis

Heterogêneas



```
>>> X = {'idade': 21, 'matricula': 10000, 'nome': 'Joao'}
>>> X
{'idade': 21, 'matricula': 10000, 'nome': 'Joao'}
```

Como dicionários não são ordenados, para acessar elementos individualmente, usa-se suas etiquetas:

```
>>> X = {'idade': 21, 'matricula': 10000, 'nome': 'Joao'}
>>> X['idade']
21
>>> X['nome']
'Joao'
```



Funções (métodos):

**clear()** Esvazia o dicionário;

**get(Chave, default)** Retorna o valor armazenado na chave passada. Caso a chave não exista, é retornado default.



Laço for:

```
>>> d = {'nome': 'Joao', 'idade': 20, 'matricula': 1}
>>> for i,j in d.items():
    print('i = ', i, 'j = ', j)
```

```
i = nome j = Joao
i = idade j = 20
i = matricula j = 1
```



# Exercício

---

7

Faça um programa que leia uma lista de nomes e depois conte quantas vezes cada nome se repete



# Exercício

7

Faça um programa que leia uma lista de nomes e depois conte quantas vezes cada nome se repete

```
#Conta quantas vezes os nomes aparecem
conta = dict()
nomes = list()

a = ' '
while a != '':
    a = input('Inserir mais um nome?\n')
    nomes += [a.capitalize()]

nomes.remove('')
#nomes = ['Joao', 'Arthur', 'Ary', 'Isabela',
        'Israel', 'Vitor', 'Joao']
for nome in nomes:
    if nome in conta:
        conta[nome] += 1
    else:
        conta[nome] = 1
print(conta)
```





# Strings

---

7

Ordenadas

Imutáveis

Homogêneas



Alguns operadores aritméticos e relacionais podem ser usados.

+ concatenação

\* múltiplas concatenações

== != <> < <= > >= comparação



# Strings

---

Laço for:

```
>>> P = 'Palavra'  
>>> for i in P:  
    print(i)
```

Neste caso, será impresso letra por letra de P, "Palavra".



# Strings

---

7

Funções (métodos):

**capitalize()** Retorna com primeiro caractere maiúsculo;

**count('A')** Conta quantos 'A';

**index('A')** (Primeiro) Índice que tem 'A';

**replace('A', 'B')** Retorna trocando todos 'A' por 'B';



# Strings

---

Funções (métodos):

**lower()** Retorna todos os caracteres minúsculos;

**upper()** Retorna todos os caracteres maiúsculos;

**split('A')** Retorna separando por 'A', por padrão é ' ';



# Strings

---

Funções (métodos):

isalnum()

isalpha()

isdecimal()

islower()

isnumeric()

isupper()



# Exercício

---

7

Faça um programa que troque nomes próprios por sequências de \*



# Exercício

7

Faça um programa que troque nomes próprios por sequências de \*

```
msg = input('Digite a mensagem:\n')

nova_msg = str()

for i in msg:
    if i.isupper():
        estou_em_nome = True
        nova_msg += '*'
        continue

    if estou_em_nome:
        if i.isalpha():
            nova_msg += '*'
            continue

        else:
            nova_msg += i
            estou_em_nome = False

    else:
        nova_msg += i

print(nova_msg)
```





# Exercício

---

7

Faça um programa que faça uma encriptação/decriptação de uma mensagem deslocando as letras no alfabeto.

Dica: Trabalhe apenas com caracteres minúsculos ou apenas com maiúsculos

Dica: Faça uma variável alfabeto = 'abcde...'



# Exercício

---

7

```
for i in mensagem.lower():
    if i in alfabeto:
        index = alfabeto.find(i)
        nova_msg += alfabeto[(index - offset)
                               % 26]
    else:
        nova_msg += i
```





# Comandos e Funções II



# Definição de Funções

---

8

```
def minhafuncsoma(a, b):  
    s = a + b  
    return s
```



# Escopo

---

8

```
def exemplo1():  
    A = 10  
    print(A)
```

```
#Outras partes
```

```
#####
```

```
exemplo1() # Chamando a funcao
```

```
print(A) # Tentando imprimir A
```



```
def exemplo1():  
    #A = 10  
    print(A)
```

```
# Outras partes
```

```
#####
```

```
exemplo1() # Chamando a funcao  
A = 10
```



```
# Partes iniciais  
#####
```

```
def exemplo1():  
    #A = 10  
    print(A)
```

```
# Outras partes  
#####
```

```
A = 10
```

```
exemplo1() # Chamando a funcao
```



# Escopo

---

```
# Partes iniciais  
#####
```

```
def exemplo1():  
    A = 1  
    print(A)
```

```
# Outras partes  
#####
```

```
A = 10  
exemplo1() # Chamando a funcao
```





# Recursividade

---

```
def fat(n):  
    if n == 1:  
        return 1  
    return n*fat(n-1)  
  
n = int(input('Digite um numero: '))  
  
r = fat(n)  
  
print('n! = ', r)
```



# Exercício

---

8

Faça uma função recursiva que calcule o  $n$ -ésimo número da sequência de fibonacci



# Exercício

---

8

Faça uma função recursiva que calcule o enésimo número da sequência de fibonacci

```
def fib(n):  
    if n == 1:  
        return 1  
    if n == 2:  
        return 1  
    return fib(n-1) + fib(n-2)
```



# Parâmetros Variados

---

8

\*args e \*\*kwargs

```
>>> def foo(*args, **kwargs):  
    print(len(args))  
    print(len(kwargs))
```

```
>>> foo('Minicurso', 'de', 'Python', curso = 'legal', programar = 'on')  
3  
2
```



# Arquivos

---

8

handle = `open`( 'arquivo', 'modo', encoding = 'codificação' )

encoding é opcional, por padrão é utf8.

Modo:

'r' read

'w' write

'a' append

Também pode ser 'r+', 'w+', 'a+'.



## Escrever

```
>>> escreve = open('new.txt', 'w')
>>> escreve.writable()
True
>>> escreve.write('Escrevendo a primeira linha\n')
28
>>> escreve.write('Agora a segunda\n')
16
>>> escreve.close()
```



Ler

```
>>> ler = open('new.txt', 'r')
>>> ler.read()
'Escrevendo a primeira linha\nAgora a segunda\n'
>>> x = ler.read() #o conteudo ja foi lido
>>> x #Logo, x nao recebera nada
''

>>> ler.close()
>>> ler = open('new.txt', 'r')
>>> x = ler.read()
>>> x
'Escrevendo a primeira linha\nAgora a segunda\n'
>>> print(x)
Escrevendo a primeira linha
Agora a segunda

>>> ler.close()
```



# Arquivos

---

8

O handle de lida é iterável

```
>>> ler = open('new.txt', 'r')
```

```
>>> for i in ler:  
    print(i, end='')
```

Escrevendo a primeira linha

Agora a segunda

```
>>> ler.close()
```





Escrever

```
>>> a = open('new.txt', 'a')
>>> a.readable()
False
>>> a.writable()
True
>>> a.write('Finalmente, a terceira linha\n')
29
>>> a.close()
```



# Exercício

---

8

Defina uma função de encriptação com offset e escreva a mensagem em um arquivo. Defina uma função de deciptação com offset e escreva a mensagem em um arquivo



# Exercício

---

8

```
for i in mensagem.lower():
    if i in alfabeto:
        index = alfabeto.find(i)
        nova_msg += alfabeto[(index - offset)
                             % 26]
    else:
        nova_msg += i
```



# Exercício

8

```
def decripte_msg(mensagem, offset):
    nova_msg = str()
    arquivo = 'mensagem_decriptada.txt'

    with open(arquivo, 'w') as arquivo_decriptado:
        for i in mensagem.lower():
            if i in alfabeto:
                index = alfabeto.find(i)
                nova_msg += alfabeto[(index - offset)
                                     % 26]
            else:
                nova_msg += i

        arquivo_decriptado.write(nova_msg)

    print('A mensagem foi decriptada:', nova_msg, 'E
          escrita no arquivo', arquivo, sep='\n')

alfabeto = 'abcdefghijklmnopqrstuvwxyz'
msg = input('Digite a mensagem que deseja
            decriptar:\n')
offset = int(input('Digite o offset:\n'))
decripte_msg(msg, offset)
```



# Modularização

---

8

## Módulos e Pacotes

Principal.py

```
import modulo
```

modulo.py

...

Pacote

\_\_init\_\_.py

modulo1.py

modulo2.py

...

...



# Modularização

---

8

Módulos e Pacotes:

Salvos na pasta Lib, ou no diretório corrente

`dir(modulo)`



# Exercício

---

8

Faça um módulo com algum arquivo anterior





---

# Controle de Fluxo II

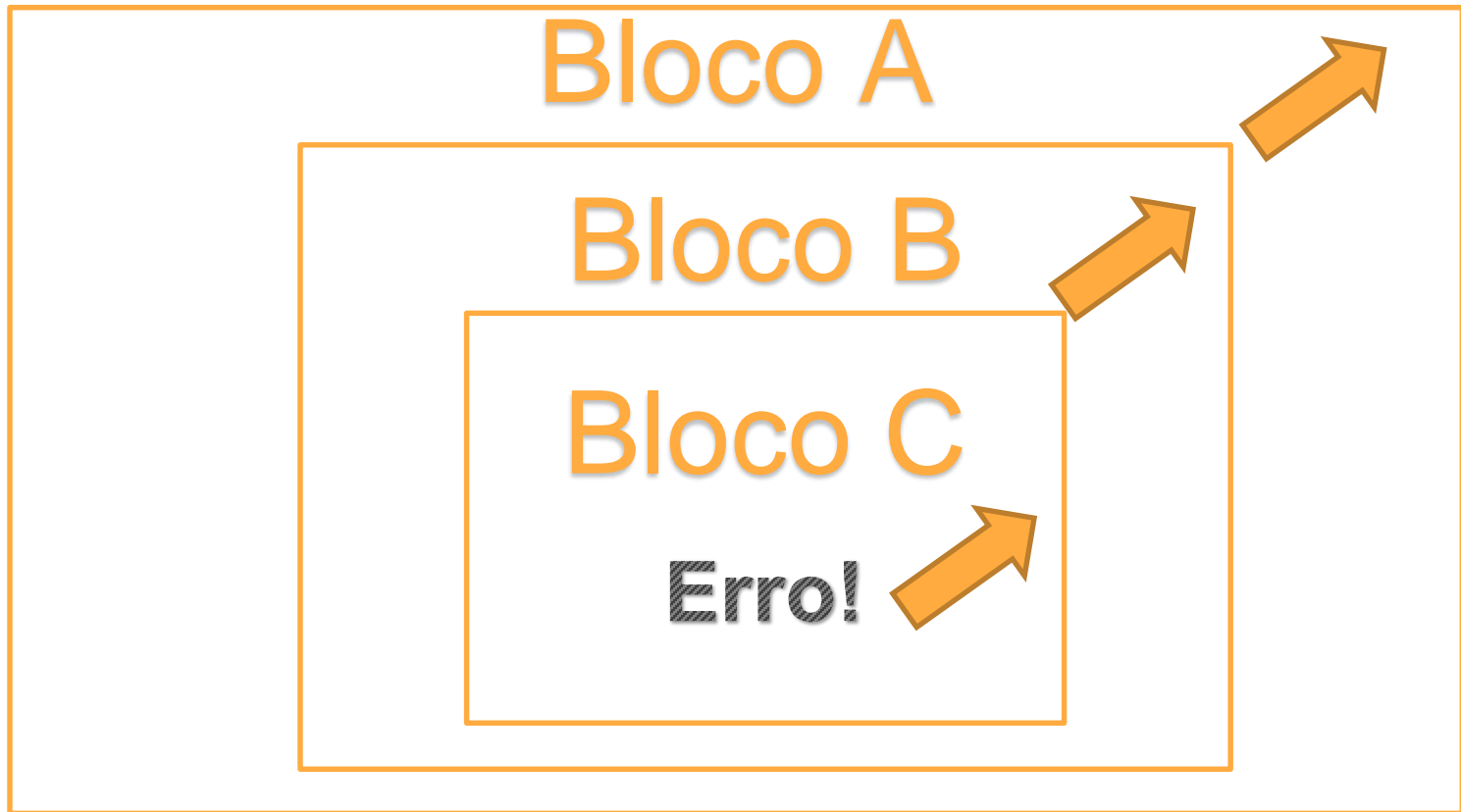
---





# Exceções

Erros geram exceções que podem ser tratadas



# Exceções

---

```
>>> Lista = ['a', 0, 1, 2]
>>> for i in Lista:
    x = 1/i
    print(x)
```

Traceback (most recent call last):

File "<pyshell#8>", line 2, in <module>

x = 1/i

TypeError: unsupported operand type(s) for /: 'int' and 'str'



## Tratar a exceção

```
import sys #Modulo usado para informar o tipo do erro

Lista = ['a', 0, 1, 2]
for i in Lista:
    try:
        x = 1/i
        print('x =', x)
    except:
        print('Nao foi possivel dividir 1 por',
              i)
        print(sys.exc_info()[0]) #Info sobre o
                                  tipo do erro
```

TypeError e ZeroDivisionError



## Especificar a exceção

```
Lista = ['a', 0, 1, 2]
for i in Lista:
    try:
        x = 1/i
        print('x =', x)
    except TypeError:
        print('Nao foi possivel dividir numero
              por caractere')
    except ZeroDivisionError:
        print('Nao foi possivel dividir por
              zero')
```



# Exceções

## Raise

```
try:
    a = int(input('Insira um valor inteiro
                  positivo:\n'))
    if a <= 0:
        raise ValueError('ERRO: 0 valor deve ser
                           positivo')
except ValueError as erro:
    print(erro)
```



# Exceções

---

Finally e with – boas práticas

```
try:  
    a = open('arquivo.txt', 'w')  
finally:  
    a.close()
```

```
with open('arquivo.txt', 'w') as a
```





---

# Orientação à Objetos

---



Instaciadas em objetos

Os tipos de dados vistos anteriormente são, na verdade, classes.

Sugestão: crie a classe caderno em um script

```
>>> class Vazia():  
    pass
```

```
>>> X = Vazia()
```





Até agora, uma classe se parece com uma struct

```
>>> class Caderno():
    cor = 'preto'
    np = 10

>>> NovoCaderno = Caderno()
>>> NovoCaderno.cor
'preto'
>>> NovoCaderno.cor = "roxo"
>>> NovoCaderno.cor
'roxo'
```



Alteração no conteúdo durante o tempo de execução

```
>>> class Vazio():
    pass

>>> X = Vazio()
>>> X.numero = 10
>>> X.numero
10
>>> dir(X)
['__class__', '__delattr__', '__dict__', '__dir__',
 '__doc__', '__eq__', '__format__', '__ge__',
 '__getattr__', '__gt__', '__hash__', '__init__',
 '__init_subclass__', '__le__', '__lt__', '__module__',
 '__ne__', '__new__', '__reduce__', '__reduce_ex__',
 '__repr__', '__setattr__', '__sizeof__', '__str__',
 '__subclasshook__', '__weakref__', 'numero']
```



“Funções” intrínsecas das classes

Vários métodos especiais

```
>>> class Caderno():
    cor = 'preto'
    np = 10
    def Descreva_se(self):
        print('Este objeto é um caderno, que contém cor e
              número de páginas')

>>> NovoCaderno = Caderno()
>>> NovoCaderno.Descreva_se()
Este objeto é um caderno, que contém cor e número de
páginas
```



Acessores, modificadores e deletores

```
>>> class Caderno():
    def getCor(self):
        return self.cor
    def getNp(self):
        return self.np
    def setCor(self, novaCor):
        self.cor = novaCor
    def setNp(self, novoNp):
        self.np = novoNp
    def delNp(self):
        del self.np
```



## Acessores, modificadores e deletores

```
>>> C = Caderno()
>>> C.setNp(10)
>>> C.np
10
>>> C.delNp()
>>> C.np
Traceback (most recent call last):
  File "<pyshell#143>", line 1, in <module>
    C.np
AttributeError: 'Caderno' object has no attribute 'np'
```



```
>>> class Pessoa():
    def setIdade(self, idade):
        print('O metodo setter foi chamado')
        if type(idade) != type(int()):
            raise ValueError('Idade invalida')
        self.__idade = idade
    def getIdade(self, idade):
        print('O metodo getter foi chamado')
        return self.__idade
    idade = property(getIdade, setIdade)

>>> P1 = Pessoa()
>>> P1.idade = 10
O metodo setter foi chamado
>>> P1.idade = ''
O metodo setter foi chamado
Traceback (most recent call last):
  File "<pyshell#77>", line 1, in <module>
    P1.idade = ''
  File "<pyshell#74>", line 5, in setIdade
    raise ValueError('Idade invalida')
ValueError: Idade invalida
```



```
# 1
idade = property(getIdade, setIdade)
# 2
idade = property(fget = getIdade, fset = setIdade)
# 3
idade = property()
idade = idade.getter(getIdade)
idade = idade.setter(setIdade)
```



## Construtores

```
>>> class Caderno():
    cor = 'preto'
    np = 10
    def __init__(self):
        print('Caderno iniciado')
    def getCor(self):
        return self.cor
    def getNp(self):
        return self.np
    def setCor(self, novaCor):
        self.cor = novaCor
    def setNp(self, novoNp):
        self.np = novoNp

>>> C = Caderno()
Caderno iniciado
```





# Exercício

---

10

Defina uma classe `fracao`, que contém dois atributos, `num` e `den`. Adicione o método de inicialização: dois parâmetros passados na definição sejam referentes à `num` e `den`, respectivamente, classe pode ser instanciada em um objeto com nenhum parâmetro, atribuindo  $0/1$ , instanciada com apenas um parâmetro inteiro qualquer  $n$ , atribuindo  $n/1$  à ela.



## Representação

```
>>> class Caderno():
    conteudo = ''
    def __init__(self, cor, np):
        print('Caderno iniciado')
        self.cor = cor
        self.np = np
    def __str__(self):
        return 'Caderno de %d páginas, da cor
            %s.\nConteúdo:\n%s' % (self.np, self.cor,
                self.conteudo)
    def escrever(self, msg):
        self.conteudo += msg
    def getCor(self):
        return self.cor
    def getNp(self):
        return self.np
    def setCor(self, novaCor):
        self.cor = novaCor
    def setNp(self, novoNp):
        self.np = novoNp
```



## Representação

```
>>> C.escrever('Ola mundo')
>>> print(C)
Caderno de 50 páginas, da cor rosa.
Conteúdo:
Ola mundo
```



## Built-in

- `object.__round__(self[, ndigits])`
- `object.__trunc__(self)`
- `object.__floor__(self)`
- `object.__ceil__(self)`
- `object.__complex__(self)`
- `object.__int__(self)`
- `object.__float__(self)`



## Estáticos

```
>>> class somador():
    def soma(self, a, b):
        return a + b
    @staticmethod
    def soma_estatica(a, b):
        return a + b

>>> somador.soma_estatica(2, 2)
4
>>> somador.soma(2, 2)
Traceback (most recent call last):
  File "<pyshell#58>", line 1, in <module>
    somador.soma(2, 2)
TypeError: soma() missing 1 required positional argument:
    'b'
>>> S = somador()
>>> S.soma(2, 2)
4
```



# Sobrecarga Operadores

10

```
>>>class Caderno():
    conteudo = ''
    def __init__(self, cor, np):
        print('Caderno iniciado')
        self.cor = cor
        self.np = np
    def __str__(self):
        return 'Caderno de %d páginas, da cor
                %s.\nConteúdo:\n%s' % (self.np, self.cor,
                self.conteudo)
    def __add__(self, other):
        return Caderno('branco', self.np + other.np)
    def escrever(self, msg):
        self.conteudo += msg
    def getCor(self):
        return self.cor
    def getNp(self):
        return self.np
    def setCor(self, novaCor):
        self.cor = novaCor
    def setNp(self, novoNp):
        self.np = novoNp
```



# Sobrecarga Operadores

10

+ object.\_\_add\_\_(self, other)

- object.\_\_sub\_\_(self, other)

\* object.\_\_mul\_\_(self, other)

@ object.\_\_matmul\_\_(self, other)

/ object.\_\_truediv\_\_(self, other)

// object.\_\_floordiv\_\_(self, other)

% object.\_\_mod\_\_(self, other)

\*\* object.\_\_pow\_\_(self, other[, modulo])

<< object.\_\_lshift\_\_(self, other)

>> object.\_\_rshift\_\_(self, other)

& object.\_\_and\_\_(self, other)

^ object.\_\_xor\_\_(self, other)

| object.\_\_or\_\_(self, other)

A / B chama A.\_\_truediv\_\_(B)



## Unários

+ object.\_\_pos\_\_(self)

- object.\_\_neg\_\_(self)

~ object.\_\_invert\_\_(self)

## Relacionais

< object.\_\_lt\_\_(self, other)

<= object.\_\_le\_\_(self, other)

== object.\_\_eq\_\_(self, other)

!= object.\_\_ne\_\_(self, other)

> object.\_\_gt\_\_(self, other)

>= object.\_\_ge\_\_(self, other)





# Sobrecarga Operadores

---

10

Operador à esquerda

Em  $A + B$ , é chamado  $A\_add\_ (B)$ ,  
Caso não esteja definido, é  
chamado  $B\_iadd\_ (A)$



# Exercício

---

10

Adicione também o método especial de impressão, para que a classe seja mostrada do tipo "num/den".

Adicione a sobrecarga de operadores para realizar as quatro operações matemáticas básicas, de modo que, por exemplo, dois objetos  $A + B$ , resultem em uma terceira fração. Não se preocupe em simplificar as frações



# Exercício

---

10

Adicione um método estático responsável por simplificar a uma fração qualquer, isto é, dado dois valores  $a$  e  $b$ , encontre a fração irredutível que represente  $a/b$ , e o sinal negativo deve sempre ficar em  $a$ .

Depois, adicione o método de simplificação no método construtor da classe.



```
>>> class Acesso():  
    public = 2  
    _protected = 2  
    __private = 2
```



```
>>> class Papel():
    _conteudo = ''
    _tamanho = 200
    def escrever(self, mensagem):
        if len(self._conteudo + mensagem) <=
            self._tamanho:
            self._conteudo += mensagem
        else:
            print('Nao cabe')
    def ler(self):
        return self._conteudo
```



```
>>> class Caderno(Papel):
        def __init__(self, paginas):
            self._tamanho *= paginas
>>> A = Caderno(10)
>>> A._tamanho
2000
>>> A.escrever('aaa')
>>> A.ler()
'aaa'
```





---

# MINICURSO PYTHON 1.0

---

Slide-aula minicurso 1.0

