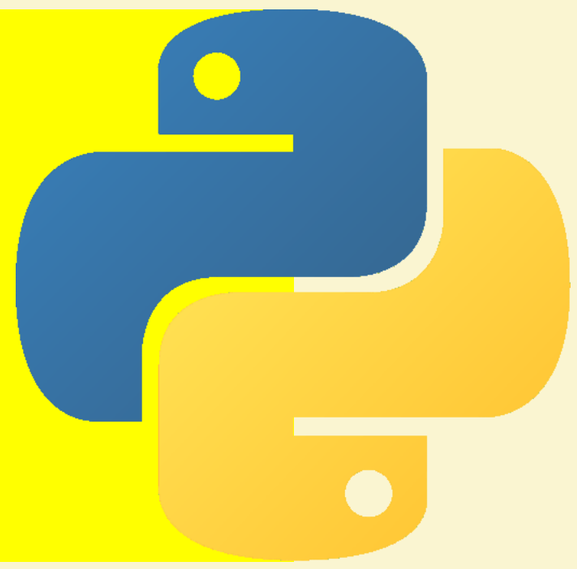
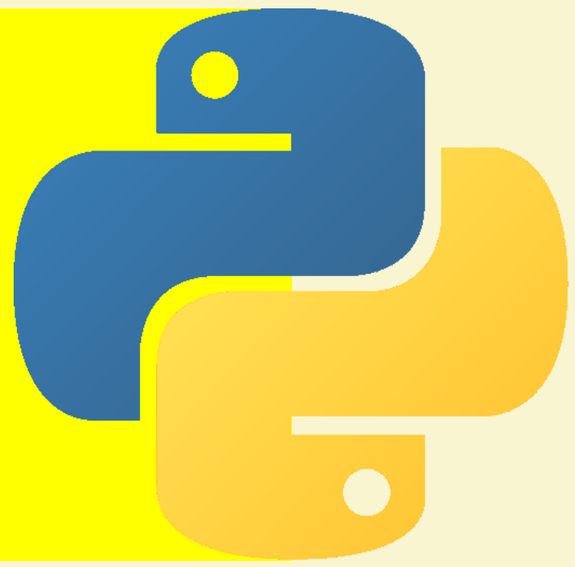
PYTHON

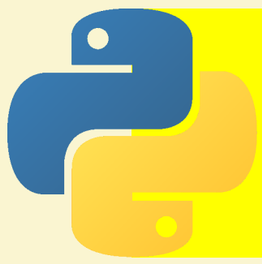


JOSÉ FABRÍCIO FIGUEIREDO

BÁSICO

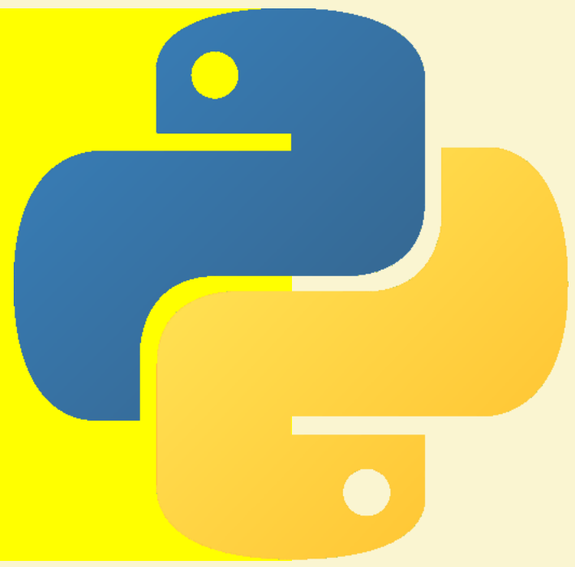


Variáveis: int, float, bool, str
Condição: if, elif, else
Loop: for-in, for-in-range, for-in-enumerate, while, [break]
Operadores Lógicos: and, or, not, in
Exceções: try/except TipoDoErro/else/finally
Funções anônima: lambda x: 3 * x + 1



JOSÉ FABRÍCIO FIGUEIREDO

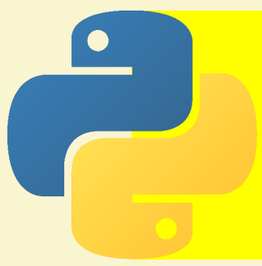
STRING



len() -> Tamanho.

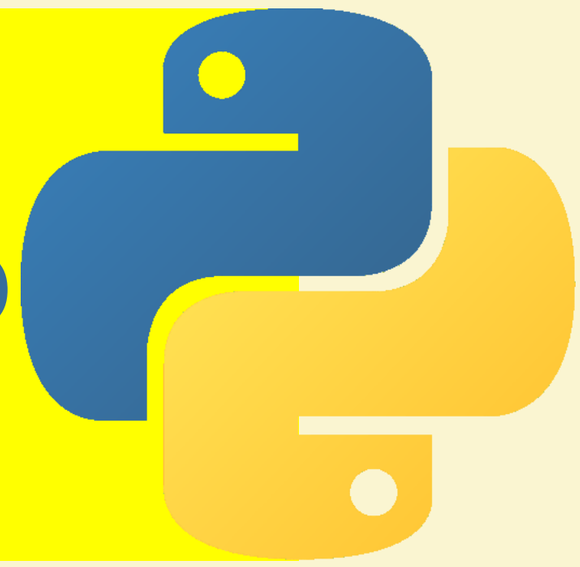
frase.count('conteúdo') -> Quantidade do conteúdo.
frase.count('conteúdo', 'inicio', 'parada') -> Quantidade do conteúdo no espaço limitado.
frase.find('conteúdo') -> Indica a posição que começa o conteúdo.
frase[inicio:termina:passo] -> Recolhe parte da String.

.replace('Curso', 'Python') -> Substitui o conteúdo.
.upper() -> Todos os caracteres são maiúsculos.
.lower() -> Todos os caracteres são minúsculos.
.capitalize() -> Primeiro caractere maiúsculo.
.title() -> Cada palavra o inicio fica em maiúsculo.
.strip() -> Remove os espaços inúteis do inicio e final.
.rstrip() -> Remove os espaços inúteis da direita.
.lstrip() -> Remove os espaços inúteis da esquerda.
.split() -> Divide uma String em uma lista.
'-'.join(frase) -> Junta a frase.



JOSÉ FABRÍCIO FIGUEIREDO

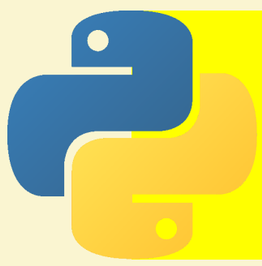
Lista/Tupla/Dicionário



 # Lista
.append() -> Adicionar elemento no final da lista.
.insert(posição, elemento) -> Inserir na posição indicada.
del lista[] -> Remover na posição indicada.
.pop() -> Remover no final da lista.
.sort() -> Ordenar crescente.
.sort(reverse=True) -> Ordenar decrescente.
.index(elemento) -> Posição do elemento indicado.
.copy() -> Copiar.
.clear() -> Deletar.
.remove(elemento) -> Remover elemento indicado.
.reverse() -> Inverter a lista.

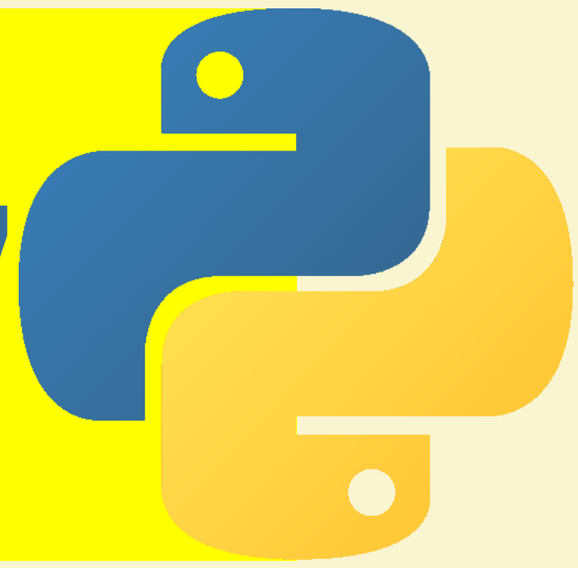
 # Tupla: São imutáveis.
tupla1 = (1, 2, 3, 4, 5)
tupla2 = 1, 2, 3, 4, 5
sorted(tupla) -> Ordenar.

 # Dicionário
dic = {'dado': 'valor'}
dic.values() -> Valores.
dic.keys() -> Dados.
dic.items() -> Valores e dados.



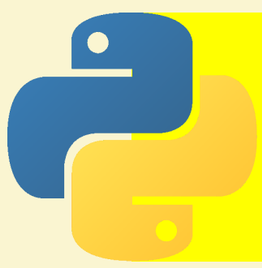
JOSÉ FABRÍCIO FIGUEIREDO

Conjuntos 'SET'



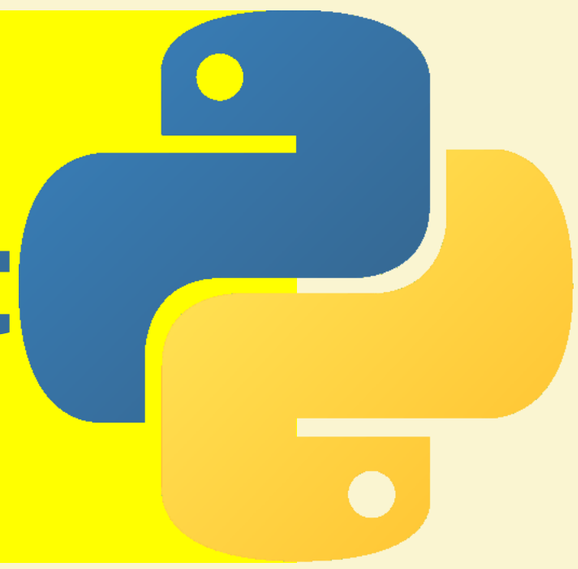
s1 = set({1, 2, 3, 4, 5}) -> Não contem repetições.
s2 = {1, 2, 3}
.add()
.remove()
.discard()

s3 & s4 -> '&' pega o s3 e s4, mostra qual valor é igual entre eles.
s3.difference(s4) -> Compara qual valor está diferente.
s3.union(s4) -> Junta sem as repetições.
s3.intersection(s4) -> Valores iguias.
s3.symetric(_difference(s4)) -> Valores diferentes.
s3.issubset(s4) -> Se os resultados do s3 se encontra no s4 (True, False).

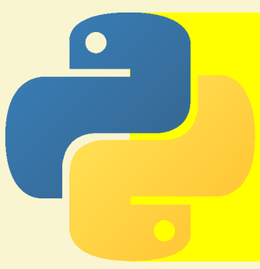


JOSÉ FABRÍCIO FIGUEIREDO

Counter/Ordereddict

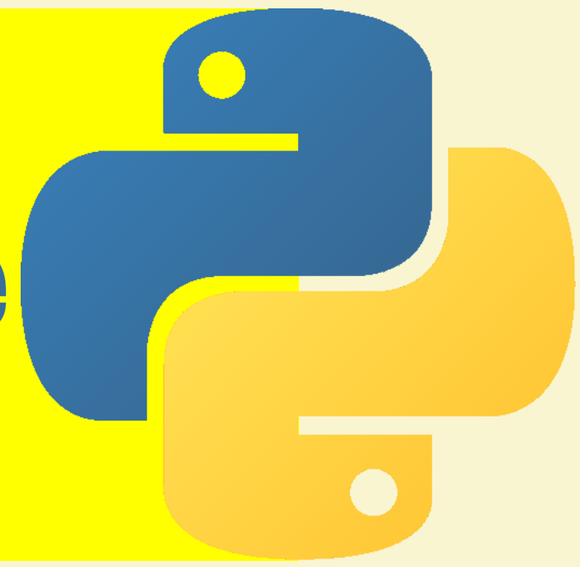


```
# Counter
from collections import Counter

frase = 'Curso de Python'
res = Conter(frase)
print(res)
>>>Counter({'C': 1, 'u':1, 'r':1, 's':1, 'o':2, ' ':2, 'd':1, 'e':1, 'P':1,
 'y':1, 't':1, 'h':1, 'n':1})

lista = [1, 1, 1, 2, 2, 3, 3, 3, 3]
print(Counter(lista))
>>>Counter({'1':3, '2':2, '3':4})

# OrderedDict
from collections import OrderedDict

dict1 = {'a':1, 'b':2}
dict2 = {'b':2, 'a':1}
print(dict1 == dict2) # A ordem não importa.
>>>True

dict1 = OrderedDict({'a':1, 'b':2})
dict2 = OrderedDict({'b':2, 'a':1})
print(dict1 == dict2) # A ordem importa.
>>>False
```

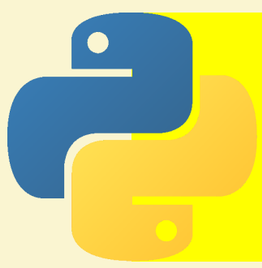


JOSÉ FABRÍCIO FIGUEIREDO

Namedtuple/Duque

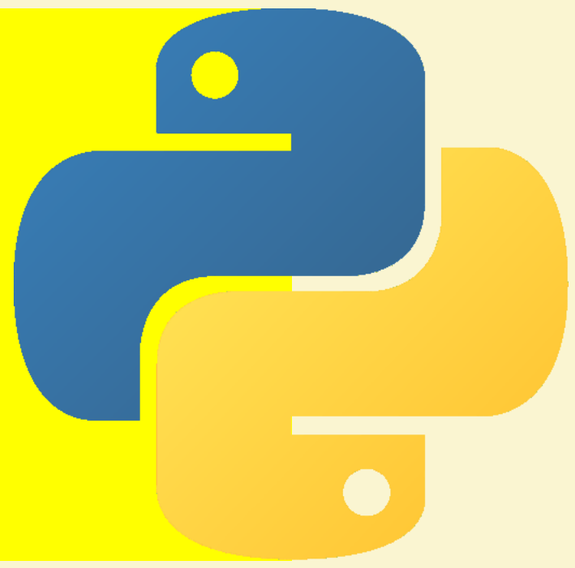


```
# Namedtuple
from collections import namedtuple

cahorro = nametuple('cachorro', ['idade', 'nome'])
beethoven = cachorro(idade=2,
nome='beethoven')
print(beethoven[0])
>>> 2 #idade
print(beethoven[1])
>>> beethoven #nome

# Deque
from collections import deque

res = deque('Python')
print(res)
>>> deque(['P', 'y', 't', 'h', 'o', 'n'])
res.append() -> Adicionar no final.
res.appendleft() -> Adicionar no inicio.
```

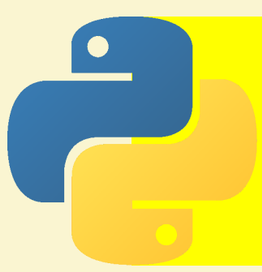


JOSÉ FABRÍCIO FIGUEIREDO

FUNÇÃO

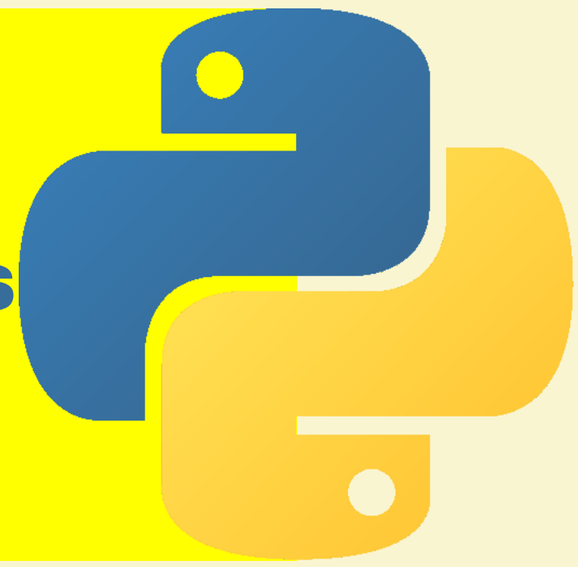


```global var

# Funções com retorno
def diz_Hello():
    return 'Hello'

# Funções com parâmetro
def soma(a, b):
    print(a + b)

# Funções com parâmetro padrão
def soma(a=0, b=0):
    print(a + b)

# Documentando funções com Docstrings
def soma(a, b):
    """descrição"""
    return a + b

# *args
def numero(*args):
    print(args)
print(numero(1, 2, 3, 4, 5))

# **kwargs
def cores(**kwarge):
    print(kwarge)
cores(A='azul', B='amarelo')
```



JOSÉ FABRÍCIO FIGUEIREDO

List Comprehension/Listas Aninhadas



List Comprehension

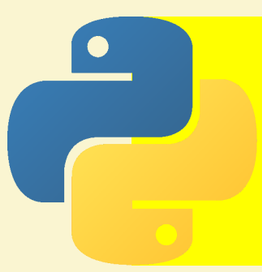
[dado for dado in iterável]
EX:

```
num = [1, 2, 3, 4, 5]
res = [num * 10 for num in num]
res = [funcao(num) for num in num]
res = [num * 10 for num in range(1, 11)]
res = [num for num in num if not num % 2]
res = [num * 2 if num % 2 else num / 2 for num in num]
```

Listas Aninhadas

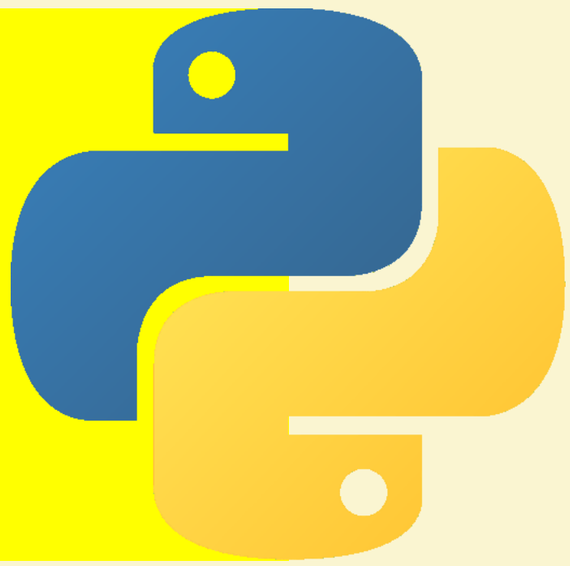
[[dado for dado in iterável]for dado in iterável]

```
num = 3
res = [['x' if i == int(num) else ' ' for i in range(1, 4)] for j in
 range(1, 4)]
```

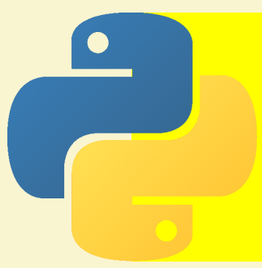


JOSÉ FABRÍCIO FIGUEIREDO

Generators/Dicionário Comprehension

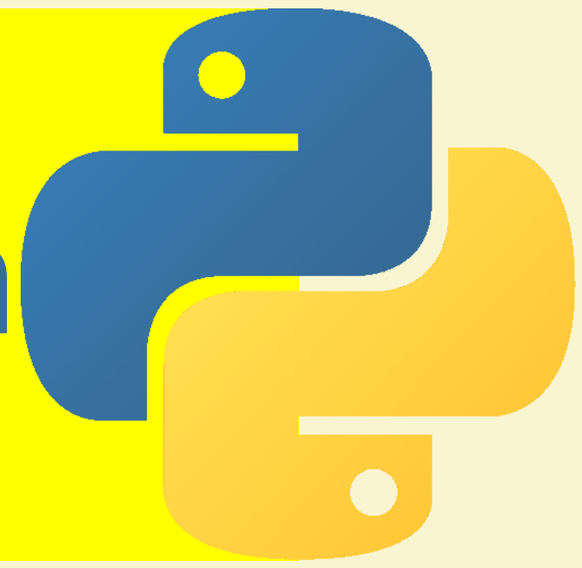


```
# Generators
res = (x * 10 for x in range(1000))
res = (num for num in [1, 2, 3, 4])

frase = 'Curso em Python'
res = (letra.upper() for letra in frase)

res = (num for num in [1, 2, 3, 4] if not num % 2)
res = (num * 2 if num % 2 == 0 else num / 2 for num in [1, 2,
 3, 4])

# Dicionário Comprehension
num = {'a':1, 'b':2}
quadrado = {chave:valor * 2 for chave, valor in num.items()}
num = [1, 2, 3, 4, 5]
res = {num: ('par' if num % 2 else 'impar') for num in num}
```

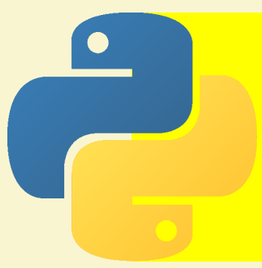


JOSÉ FABRÍCIO FIGUEIREDO

Set Comprehension

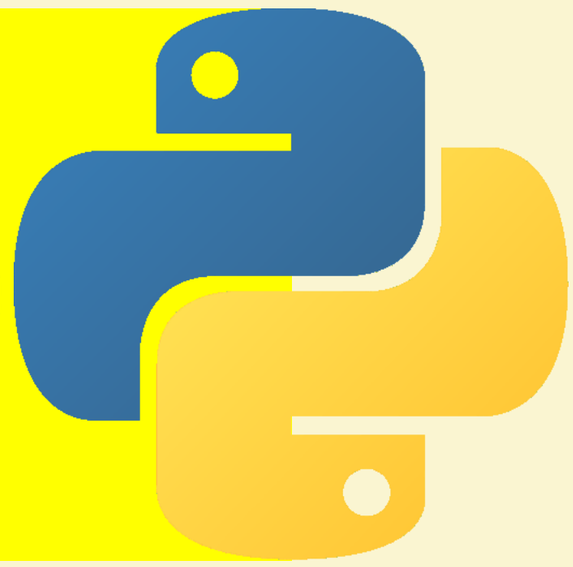


```
num = {num for num in range(1, 10)}

num = {x ** 2 for num in range(1, 10)}

letra = {letra for letra in 'Curso de Python'}

num = {num * 2 if num % 2 else num for num in range(1, 10)}
```

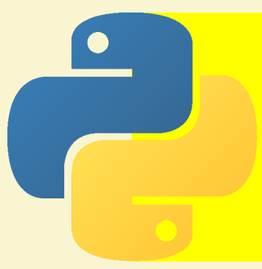


JOSÉ FABRÍCIO FIGUEIREDO

Map/Filter/Reduce

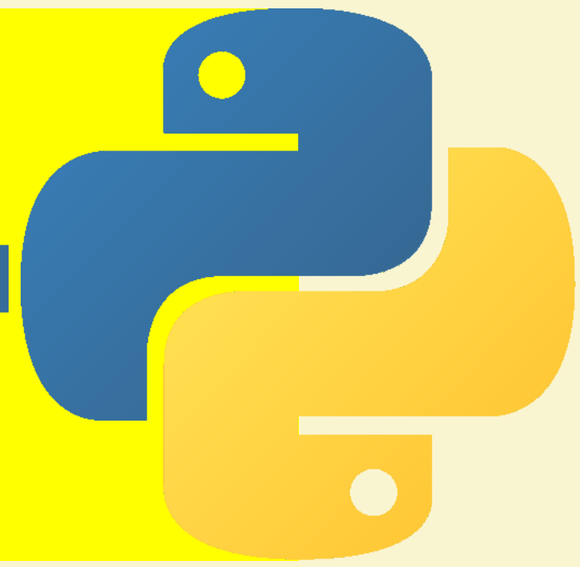


```
# Map: Após utilizar, o resultado de map() zera.
map(função, dados)
Exemplo:
dobro = [1, 2, 3, 4, 5]
res = map(lambda d: d * 2, dobro)

# Filter
filter(função, dados)
Exemplo:
media = 2.18
dados = [1.3, 2.7, 0.8, 4.1]
res = filter(lambda valor: valor > media, dados)
>>> [2.7, 4.1]

# Reduce
res1 = f(a1, a2)
res2 = f(res1, a3)

Exemplo:
from functools import reduce

dados = [2, 3, 4, 11, 13, 19, 30]
mult = lambda x, y: x * y
res = reduce(mult, dados)
>>> 2445300
```



JOSÉ FABRÍCIO FIGUEIREDO

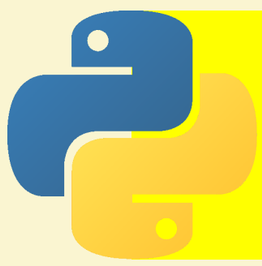
any/all/min/max/len/abs/sum/round



all() -> Retorna True se todas as iteráveis são verdadeiras ou ainda o iterável está vazio.
any() -> Retorna True se qualquer elemento da iterável for verdadeiro. Se tiver vazio retorna False.

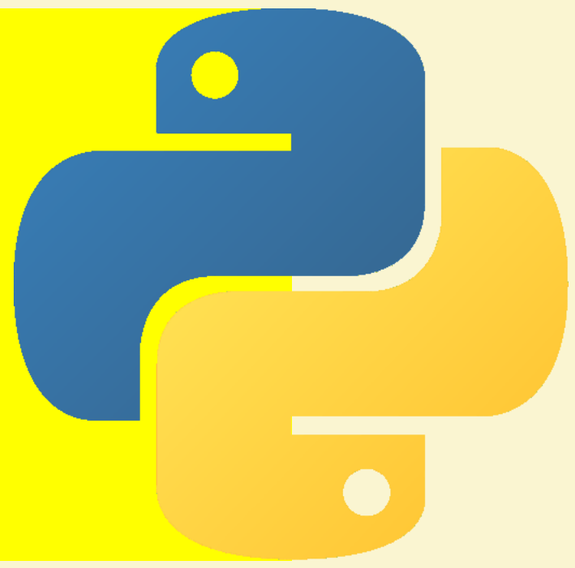
min() -> Retorna o menor valor.
max() -> Retorna o maior valor.

abs() -> Retorna um valor absoluto de um número inteiro ou real.
sum() -> Retorna a soma total dos elementos incluindo o valor inicial 'default=0'.
round() -> Retorna um número arredondado para 'n' digitos de precisão após a casa decimal.

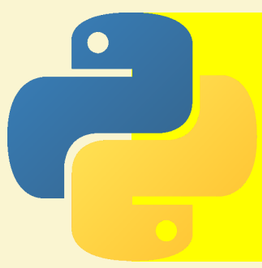


JOSÉ FABRÍCIO FIGUEIREDO

ZIP/RAIZE

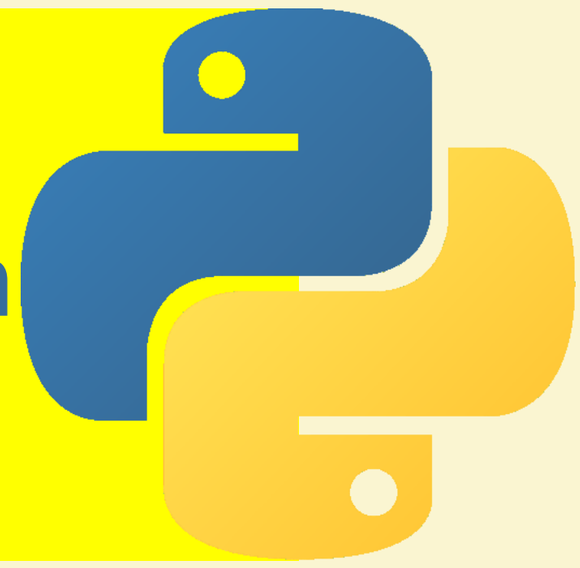


```
# Zip
lista1 = [1, 2, 3]
lista2 = [4, 5, 6]

print(list(zip(lista1, lista2)))
>>>[(1, 4), (2, 5), (3, 6)]

# Raise
raise TipoDoErro('Mensagem de Erro')

texto = input('Digita: ')
if type(texto) in not str:
raise TypeErro('')
```



JOSÉ FABRÍCIO FIGUEIREDO

Erros Mais Comuns em Python



SyntazError -> Ocorre quando o Python encontra um erro de sintaxe. Ou seja, você escreve algo que o Python não reconhece como parte da linguagem.

NameError -> Ocorre quando uma variável ou função não foi definida.

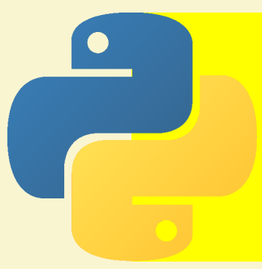
TypeError -> Ocorre quando uma função/operação/ação é aplicada a um tipo errado.

IndexError -> Ocorre quando tentamos acesar um elemento em uma lista ou outro tipo de dado indexado utilizando um indice invalida.

ValueError -> Ocorre quando uma função/operação built-in recebe um argumento com tipo correto mas valor inapropriado.

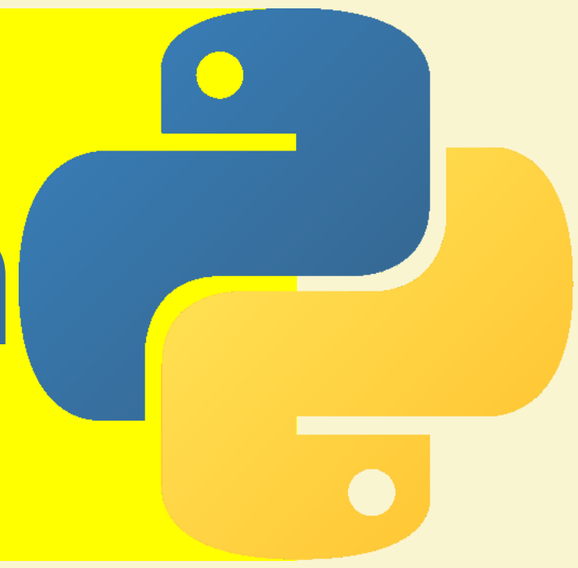
KyeError -> Ocorre quando tentamos acessar um dicionario com uma chave que não existe.

AttributeError -> Ocorre quando um variável não tem um atributo/função. IndentationError -> Ocorre quando não respeitamos a indentação do Python(4 espaços).



JOSÉ FABRÍCIO FIGUEIREDO

Módulo Random



import random

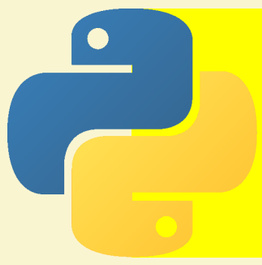
random.random() -> Gera um número pseudo-aleatório entre 0 e 1.

random.uniforme(0, 0) -> Entre os valores estabelecidos gera era um número pseudo-aleatório, sendo int ou float.

random.randint() -> Valores aleatórios entre os valores estabelecidos, sendo int.

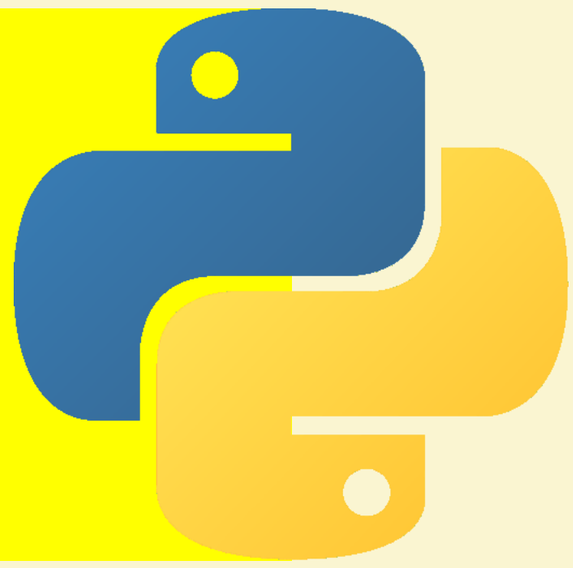
random.choice() -> mostra um valor aleatorio em um iterável.

random.shuffle() -> Embaralha dados.

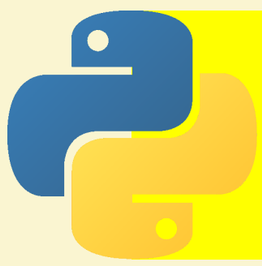


JOSÉ FABRÍCIO FIGUEIREDO

bult-in/Dunder Main & dunder name

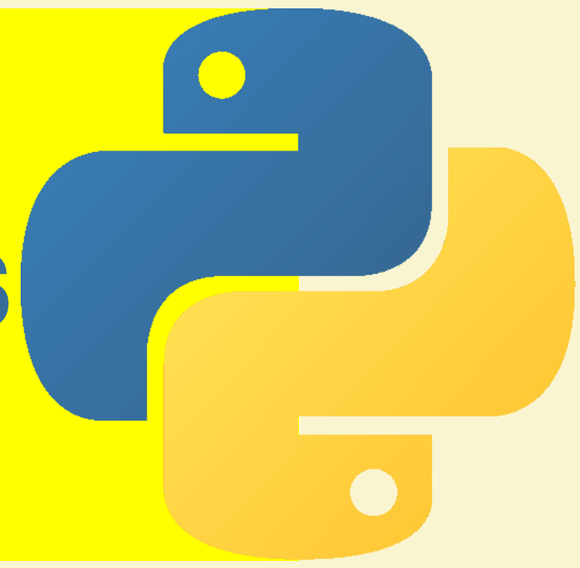


```
# Utilizando apelidos para módulos/funções
    import random as rdm
# Podemos importar toas as funções
    from random import *
# Dunder main & Dunder name
if __name__ == '__main__':
    # Teste do módulo, que será importado.
else:
    print('O módulo foi importado')
```

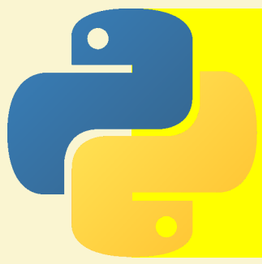


JOSÉ FABRÍCIO FIGUEIREDO

Abertura de Arquivos

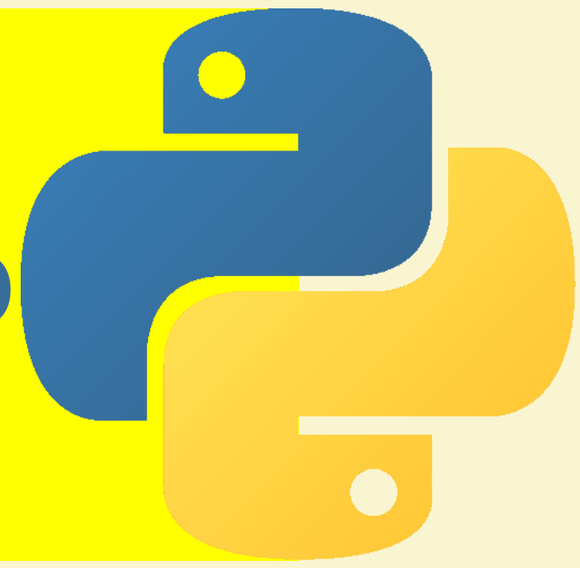


r - > Abre para leitura.
w -> Abre para escrita - Sobrescreve caso o arquivo já existe.
x -> Abre para escrita somente se o arquivo não existe. Caso contrário, gera FileExistsError.
a -> Abre para escrita, adicionando o conteúdo sempre ao final do arquivo.
+ -> abre para o modo de atualização: Leitura e escrita.



JOSÉ FABRÍCIO FIGUEIREDO

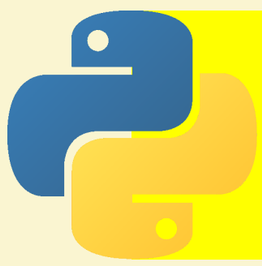
Ler/Escrita de Arquivo



open() -> Abre o arquivo ou cria.
arquivo.write() -> Escreve em arquivos.
arquivo.read() -> Lê o conteúdo.
arquivo.seek() -> Movimenta o cursor.
arquivo.readline() -> Função que lê o arquivo linha a linha.
arquivo.close() -> Fecha o arquivo.
arquivo.closed() -> False ou True se o arquivo esta fechado ou aberto.

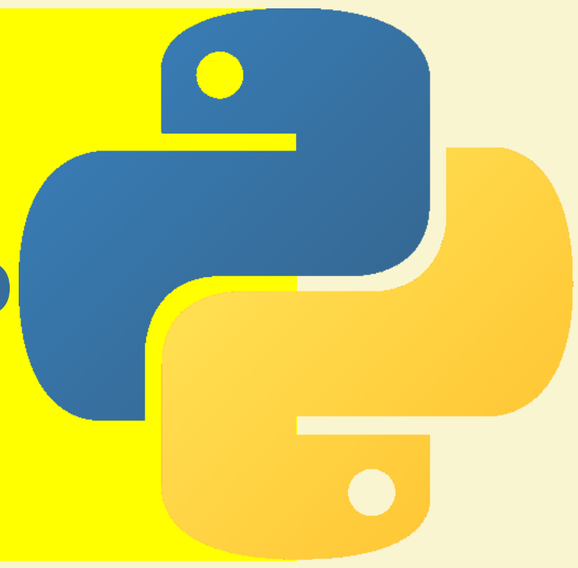
```
with open('arquivo', 'função') as apelido:
    # Comando utilizados para manipular um arquivo.
    # Fechar.
```

```
# StringIO: Utilizado para ler e criar arquivos em memória.
Exemplo:
from io import StringIO

arquivo = StringIO()
# Funções de leitura e escrita.
```

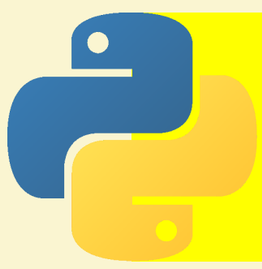


JOSÉ FABRÍCIO FIGUEIREDO

Sistema de Arquivo-Aavegação

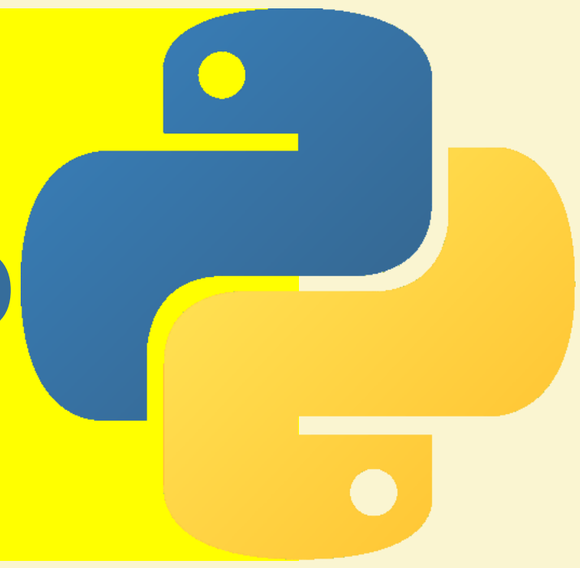


import os, sys
os.getcwd() -> Path do arquivo.
os.path.isbs('url') -> Diretório absoluto ou relativo.
sys.platform -> Detalhe no sistema.
os.name -> Identificar sistema operacional (posix(linux & mac), nt (Windows)).
os.path.join('path01', 'path02')
os.chdir() -> Nova Path.
os.listdir() -> Exibe os arquivos e diretórios.
os.scandir() -> Exibe os arquivos e diretórios com mais detalhes.
os.inode() -> Numeração do elemento.
os.is_dir() -> É um diretório?
os.is_file() -> É um arquivo?
os.is_symlink() -> É um link simbólico?
os.path -> Caminho do arquivo.
os.stat() -> Estatística.
os.path.exists('nome_do_arquivo_ou_path') -> O arquivo existe?
os.mkdir('nome_do_arquivo') -> Criar diretório.
os.mkdirs('nomes_dos_arquivos_separados_pos_'/', exist_ok=True) -> Cria todos os diretórios.
os.rename('nome_do_arquivo_que_será_subs tituido', 'nome')
os.remove('nome_do_arquivo')
os.rmdir('diretorio')
os.removedirs('path')

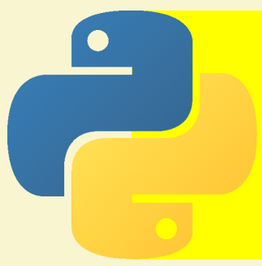


JOSÉ FABRÍCIO FIGUEIREDO

Diretório temporario

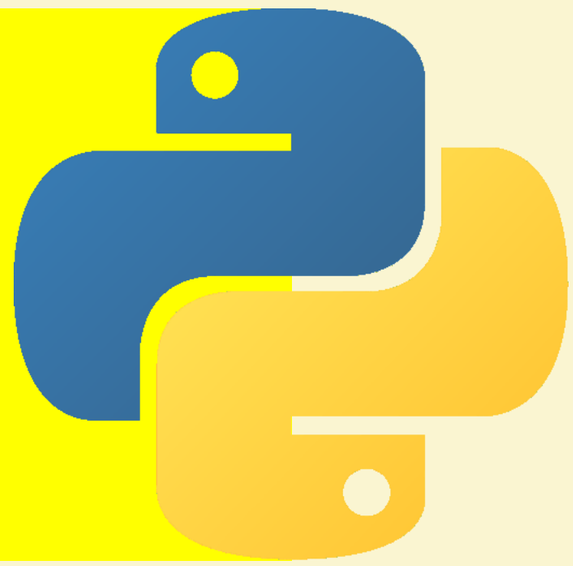


```
import os
import tempfile

with tempfile.TemporaryDirectory() as tmp:
    print(f'diretorio{tmp}')
    with opne(os.path.join(tmp, 'arguivo.txt'), 'w') as ar:
        ar.write('Curso de Python')
    input()
```



JOSÉ FABRÍCIO FIGUEIREDO

iteráveis & Iterators / Generator functions



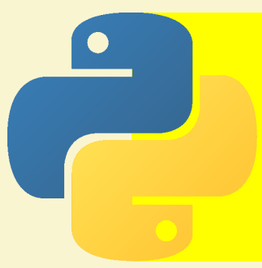
 # Iteráveis & Iterators

inter() -> Um objeto que irá retornar um iterator quando a função inter() for chamada.
next() -> Um objeto que pode ser iterado / Retorna um elemento por vez.

 # Generator Functions

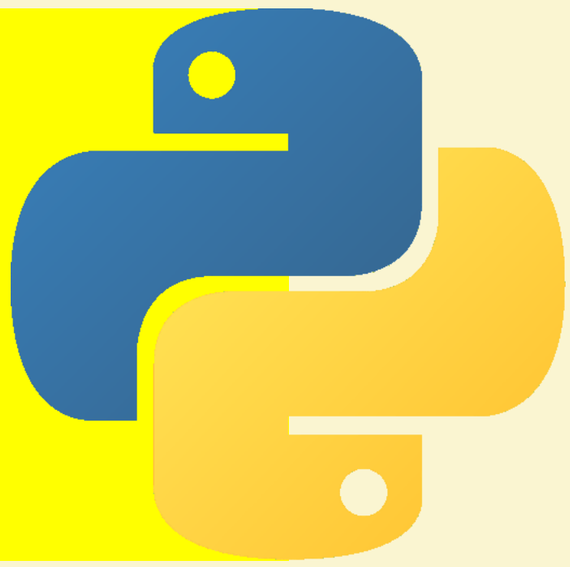
```
def cont_ate(valor_maximo):
    cont = 1
    while cont <= valor_maximo:
        yield cont
        cont++
```

```
print(list(conta_ate(10))
>>>1, 2, 3, 4, 5, 6, 7, 8, 9, 10
```

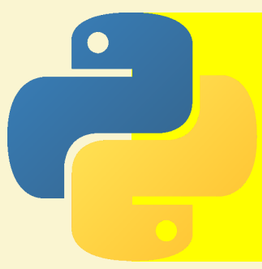


JOSÉ FABRÍCIO FIGUEIREDO

csv: Reader / Dictreader / Writer / Dictwriter

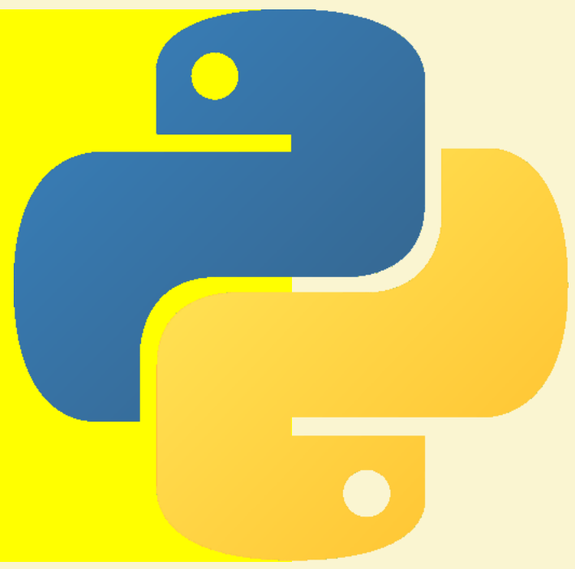


```
# Reader
from csv import reader
with open('arquivo', encoding='utg-8') as arq:
    leitor_csv = reader(arq)
    next(leitor_csv) # Pular o cabeçalho
    for linha in leitor_csv:
        print(linha)

# DictReader
form csv import DictReader
with open('arquivo', ecoding='utf-8') as arq:
    leitor_css = DictReader(arq)
    for linha in leitor_csv:
        print(linha['titulo1'], linha['titulo2'])

# Writer
from csv import writer
with open('arquivo', 'a') as arq:
    escritor_csv = whiter(arq)
    escritor_csv.writerrow([t1, t2])

# DictWriter
form csv import DictWriter
with open('arquivo', 'w') as arq:
    cabacalho = ['titulo', 'genero', 'duração']
    escritor_csv = DictWriter(arq, fieldnames=cabecalho)
    escritor_csv.writeheader() # Pular linha
    escritor_csv.writerow({'titulo':'YU-GI-HO, 'genero':'Anime',
 'Duração':'3h'})
```

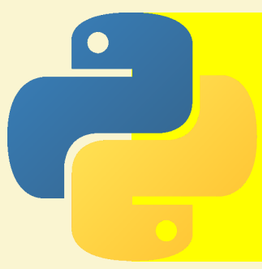


JOSÉ FABRÍCIO FIGUEIREDO

DATA E HORA

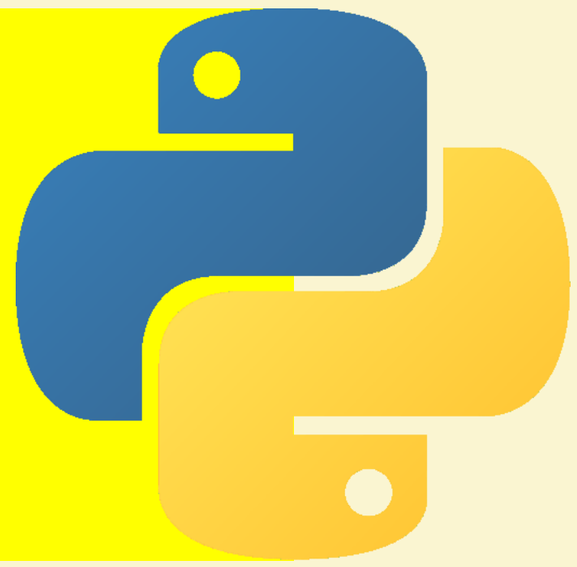


```
import datetime
datetime.datetime.now().replace(year=0, hour=0, minute=0,
 segond=0, microsecond=0)

# Acessar individualmente os elementos de data e hora.
inicio.year() -> ano
inicio.month() -> Més.
inicio.day() -> Dia.
inicio.hour() -> Hora.
inicio.minute() -> Minuto.
inicio.second() -> Segundo.
inicio.microsecond() -> Microssegundo.

# Delta
ret = datetime.timedelta(day=0, seconds=0,
 microseconds=0)
delta = data_inicio ± ret

ret = datetime.datetime.combine(inicio + ret +
 detetime.time(hour=0, minute=0, microsecond=0)

ret.weekday() -> Dia da semana.
#0 - Segunda-feira
#1 - Terça-feira
#2 - Quarta-feira
#3 - Quinta-feira
#4 - Sexta-feira
#5 - Sábado
#6 - Domingo
```



JOSÉ FABRÍCIO FIGUEIREDO

DATA E HORA



inicio.strftime('%d/%m/%Y) -> Formato da data.

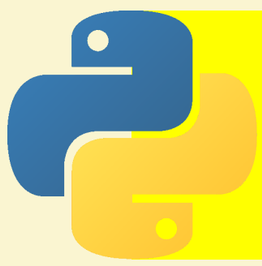
 # TextBlod

from textblob import TextBlob

ret = TextBlob(inicio.strftime('%B').translate(to='pt-br') ->

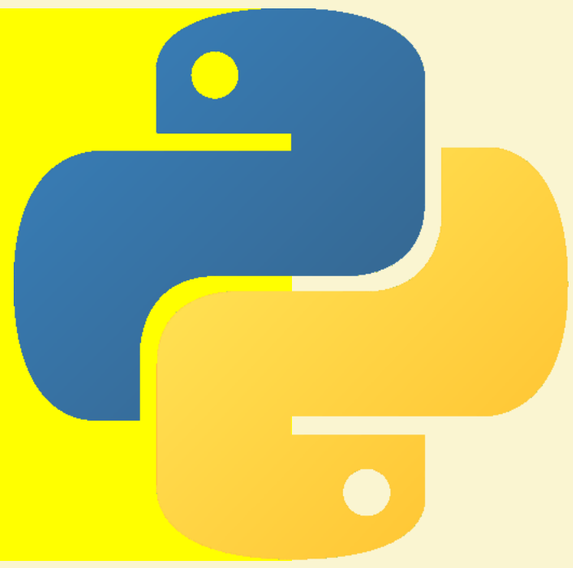
Mostra o nome do mês em português.

ret = datetime.datetime.strptime('10/04/1998', '%d/%m/%Y')

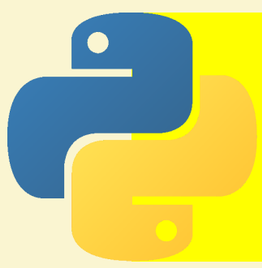


JOSÉ FABRÍCIO FIGUEIREDO

Assertions / Doctests

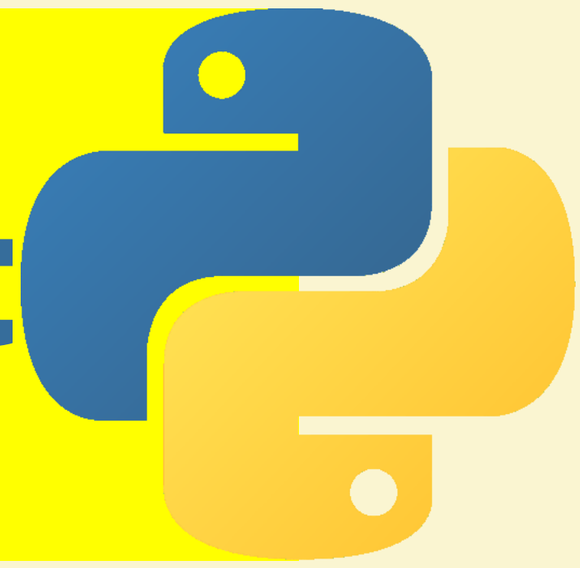


```
# Assertions
def soma(a):
    assert a>=0, 'Mensagem'
    return a + a

# Doctests
def soma(a, b):
    """Soma os númeors a e b
    >>>soma(1, 2)
    3
    """
    return a + b

# Terminal
    python -m doctest -v arquivo.py
```

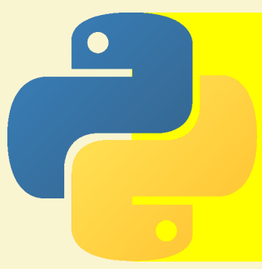


JOSÉ FABRÍCIO FIGUEIREDO

Módulo Unittest

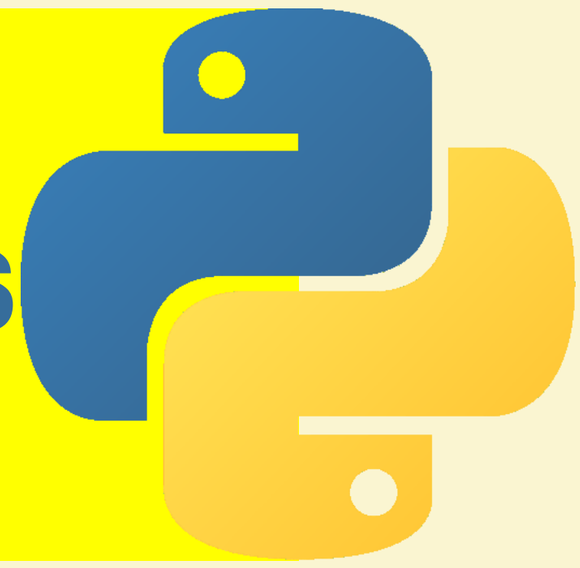


```
import unittest
from arquivo import funções

class AtividadesTestes(uniteest.TestCase):
    def nome_teste(self):
        self.assertEsqual(
            função(), resultado_previsto
        )
if __name == '__main__':
    unittest.main()

# Terminal
python nome_arquivo.py

#Conhecendo os assertions.
assertEqual(a, b) -> a == b
assertNotEqual(a, b) -> a != b
assertTrue(x) -> x = True
assertFalse(x) -> x = False
assertIs(a, b) -> a é b
assertIsNot(a, b) -> a não é b
assertIsNone(x) -> x é None
assertIfNone(x) -> x é None
assertIsNotNone(a, b) -> x não é None
assertIn(a, b) -> x não é None
assertNoteIn(a, b) -> a não está em b
assertIsInstance(a, b) -> a é instancia de b
assertNotIsInstance(a, b) -> a não é instância de b
```



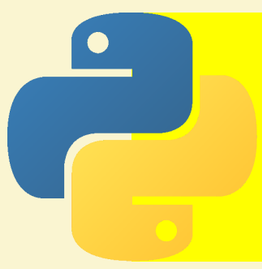
JOSÉ FABRÍCIO FIGUEIREDO

Antes e Após hooks



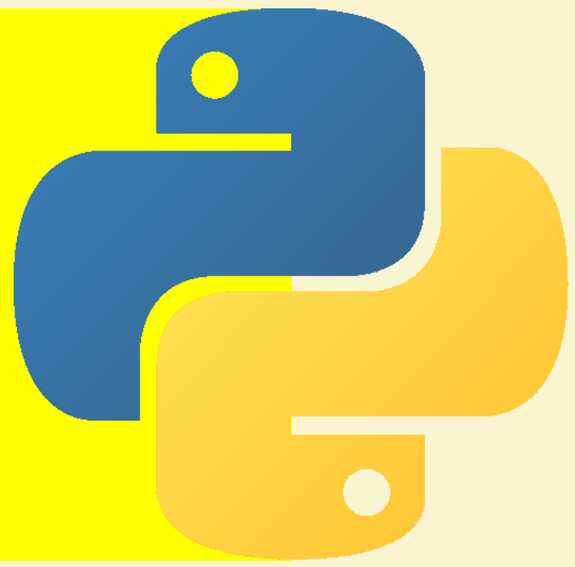
hooks -> Execução dos testes.

```
# Exemplo:
import unittest
class ModuloTest(unittest.TestCase):
    def setup(self):
        # Configurações do setup.
        pass
    def test_primeiro():
        # setup() rodará antes do teste
        # tearDown() rodará após o teste
        pass
    def tearDown(self):
        #Configuração do tearDown()
        pass
```

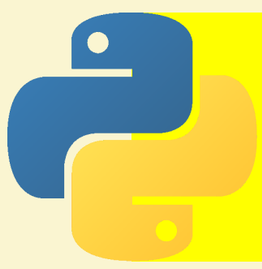


JOSÉ FABRÍCIO FIGUEIREDO

TYPE HINTING

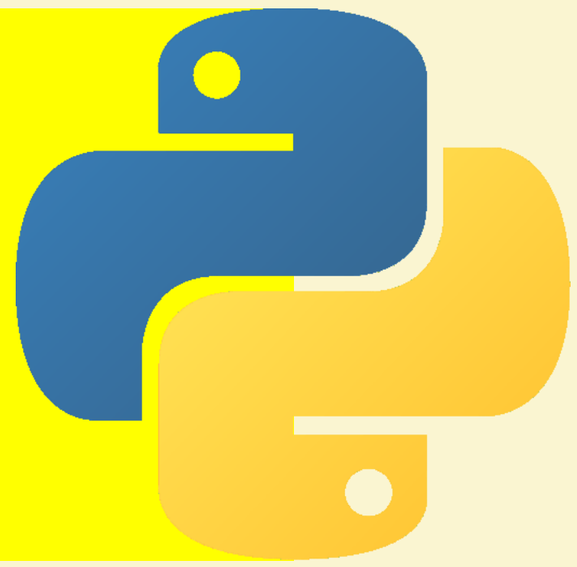


```
# Exemplo 01:
def funcao(x: str) -> str:

# Exemplo 02:
def funcao(x):
    #type: (float) -> float

# Exemplo 03:
def funcao(x, y):
    #type: (str, bool) - str

# Terminal
mypy nome_arquivo.py

# Exemplo 04:
def funcao(
    a, #type: bool
    ): type:(...) -> str

# Exemplo 05:
res:str = 'Ola'

# Exemplo 06:
from typing import List, Tuple, Set, Dict

res1:List[str] = ['Ola', 'Mundo']
res2:Tuple[str] = ('Ola', 'Mundo')
res3:Set[str] = {'Ola', 'Mundo'}
res4:Dict[str, int] = {'Ola': 13}
```

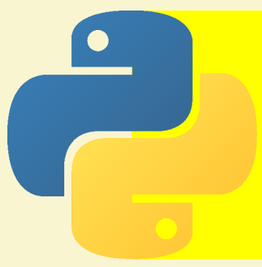


JOSÉ FABRÍCIO FIGUEIREDO

DECORADORES

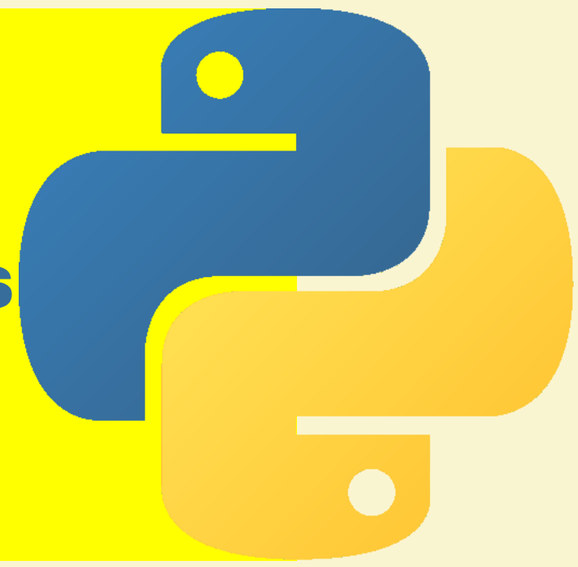


```
def função01(função):
    def função03():
        print('1')
        função()
        print('3')
    return função03

@função01
def função02():
    print('2')

função02()
>>> 1
>>> 2
>>> 3
```

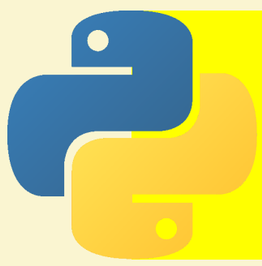


JOSÉ FABRÍCIO FIGUEIREDO

Decoradores com diferentes assinaturas

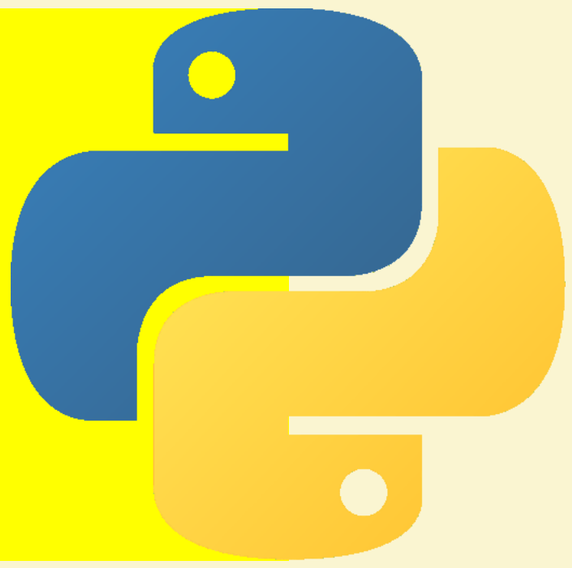


```
def função02(valor):
    def função03(função):
        def função04(*args, **kwargs):
            if args and args[0] != valor:
                return 'valor incorreto'
            return função03(*args, **kwargs)
        return função04

@função02('valor') #@função02(str, int)
def função01(*args):
    return args

#OBS: Caso os valores estejam de acordo com o valor
 padrão, retorna a função01.
```

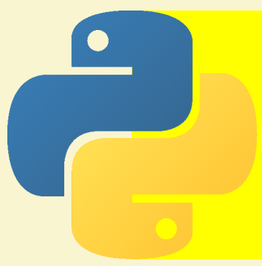


JOSÉ FABRÍCIO FIGUEIREDO

WRAPS

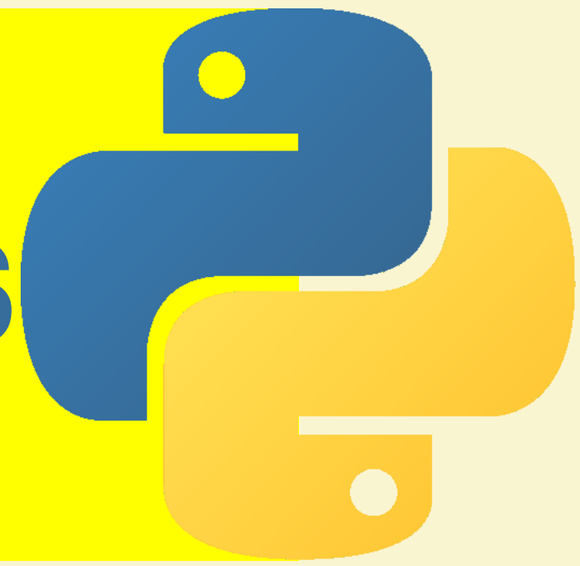


```
from functools import wraps

def funcao02(funcao):
    @wraps(funcao)
    def funcao03(*args, **kwargs):
        """Docstrings"""
        print(funcao.__name__)
        print(funcao.__doc__)
        return funcao(*args, **kwargs)
    return funcao03

@funcao02
def funcao01():
    """Docstrings_1"""
    pass

print(somar.__name__) #funcao01
print(somar.__doc___)#Docstrings_1
```



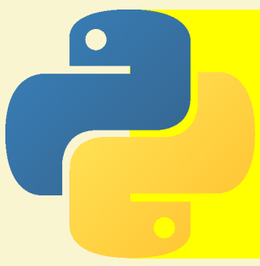
JOSÉ FABRÍCIO FIGUEIREDO

POO - Atributos



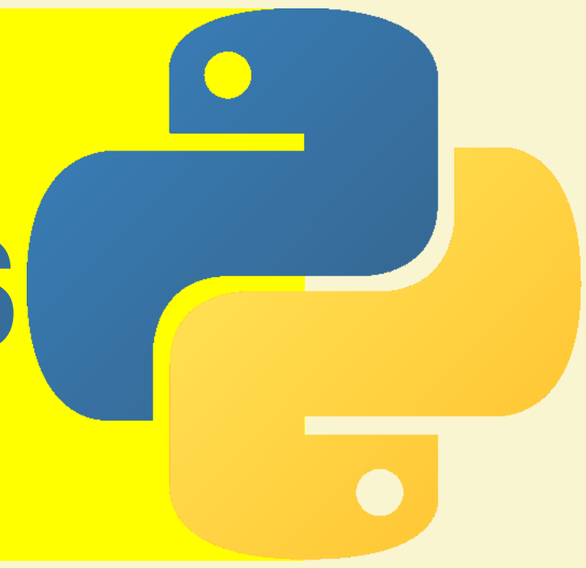
#Atributos de instâncias:

```
class MyClass:
    def __init__(self, x, y):
        self.x = x
        self.y = y

# Atributo de classes:
class MyClass:
    x = 'Valor'
    def __init__(self):
        pass

# Atributo Dinâmicos:
class MyClass:
    contador=0
        def __init__(self):
            self.id = Produto.contador += 1
            Produto.contador = self.id

# Criando atributo
MyClass().novo = 'Nova Variável'

# Deletar atributos
del MyClass().novo

# Acessar variável
MyClass().nomeVariável
print(MyClass().__dict__)
>>>{'var01':value, 'var02':value}
```

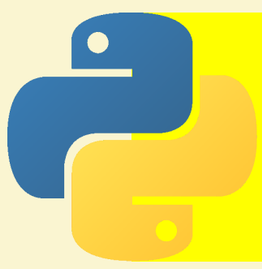


JOSÉ FABRÍCIO FIGUEIREDO

POO - Objetos

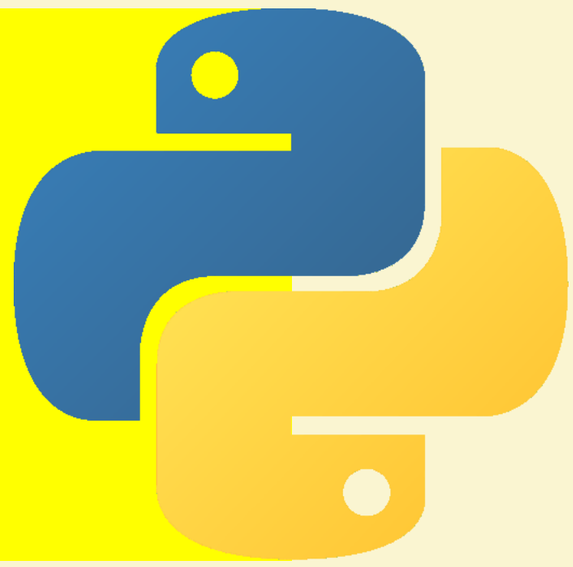


```
class MyClass:
    def __init__(self):
        pass
    def funcao01(self):
        pass

x = MyClass()
x.funcao01()
```



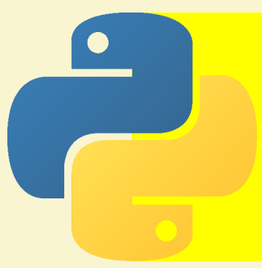
JOSÉ FABRÍCIO FIGUEIREDO

POO - abstração e encapsulamento



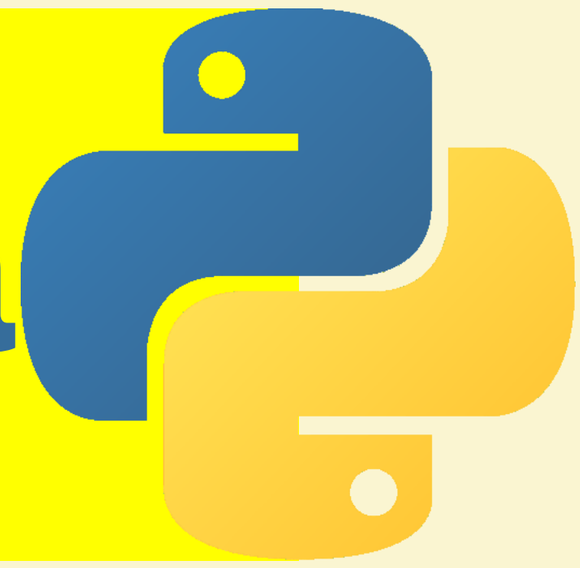
 # Os elementos privados só devem/deveriam ser acessados dentro da classe, mas Python não bloqueia este acesso fora da classe. Com Python acontece um fenômeno chamado Name Mangling.

```
class MyClass:
    def __init__(self, x):
        self.__x = x
    def acessar(self):
        return self.__x
```

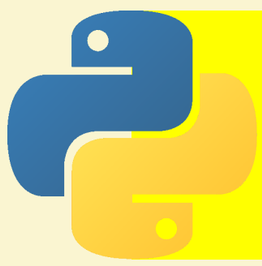


JOSÉ FABRÍCIO FIGUEIREDO

POO - Herança

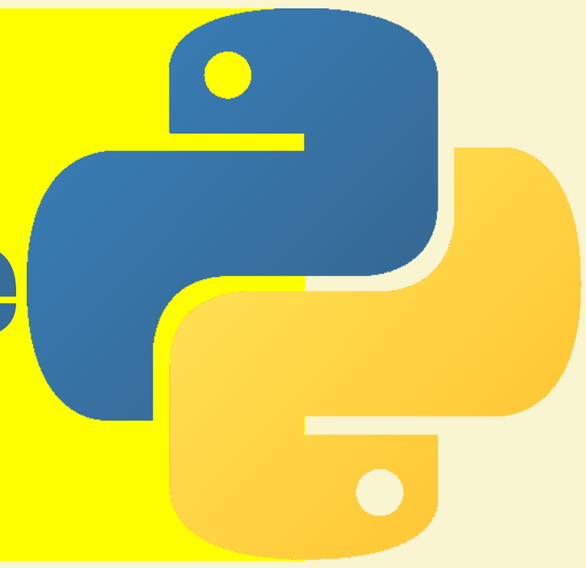


```
class MyClassFilha:
    def __init__(self, x):
    self.x = x

class MyClassMae(MyClassFilha):
    def __init__(self, x, y):
        # MyClassFilha.__init__(self, x)
        super().__init__(self, x)
        self.y = y

    def funcao(self):
        pass

x = Cliente(x, y)
print(x.__dict__)
```



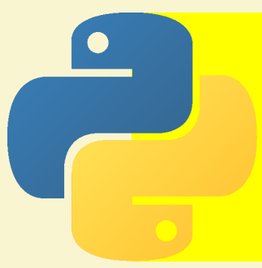
JOSÉ FABRÍCIO FIGUEIREDO

POO - Propriedade



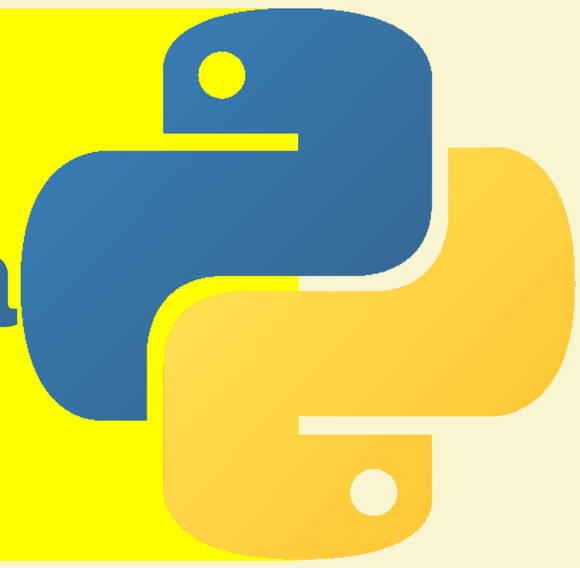
getters (@property) -> Retorna o valor do atributo.
setters (@funcao.setter) -> Altera o valor do mesmo.

```
class MyClass:
    def __init__(self, x):
        self.x = x
    @property
    def funcao01(self):
        return self.x
    @num.setter
    def funcao02(self, novo):
        self.x = novo

x = MyClass(13)
x.funcao01
x.funcao02 = 10
```

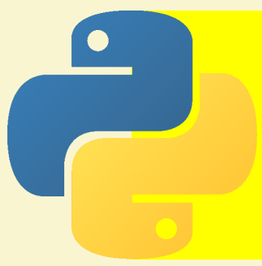


JOSÉ FABRÍCIO FIGUEIREDO

POO - Herança Múltipla

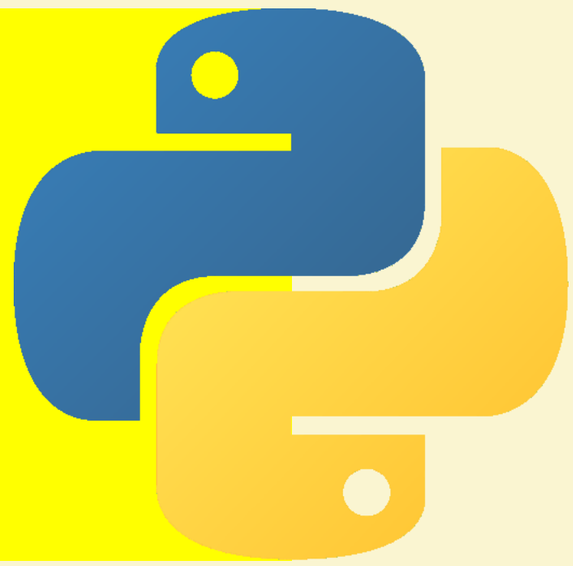


```
# Multi derivação Direta:
class Base1:
    pass
class Base2:
    pass
class Multi derivação(Base1,Base2,Base3):
    pass

# Multi derivação Indireta:
class Base1:
    pass
class Base2(Base1):
    pass
class Multiderivação(Base2):
    pass
```

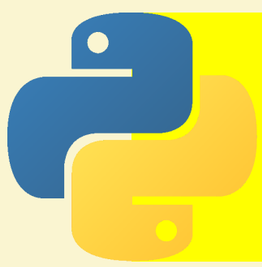


JOSÉ FABRÍCIO FIGUEIREDO

POO-MRO-Method Resolution Order

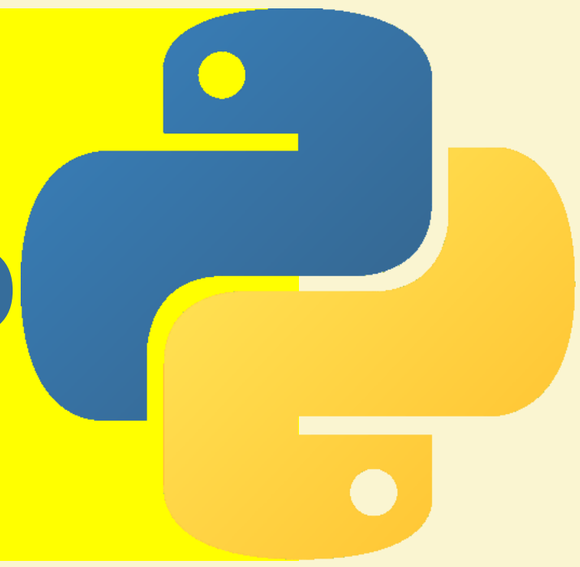


```
# Mostra a ordem da classe mãe e classes filhas:
print(MyClass.__mro__)
```



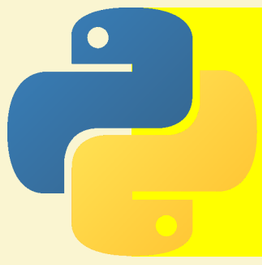
JOSÉ FABRÍCIO FIGUEIREDO

POO - Polimorfismo



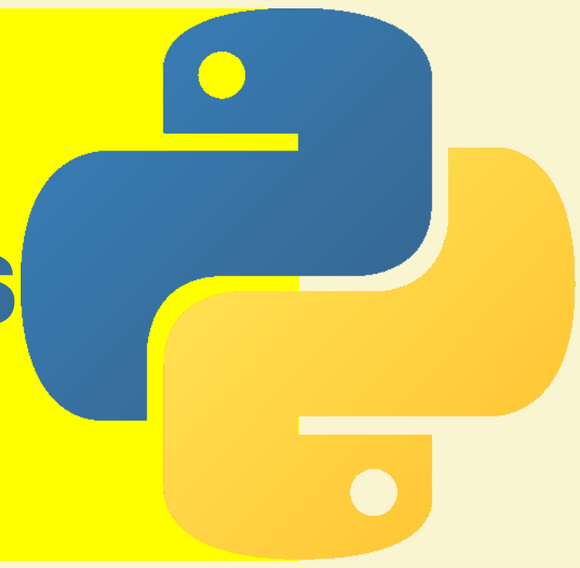
Sobrescrever a função:

EX: Temos a funcao01, mas criamos outra funcao01.



JOSÉ FABRÍCIO FIGUEIREDO

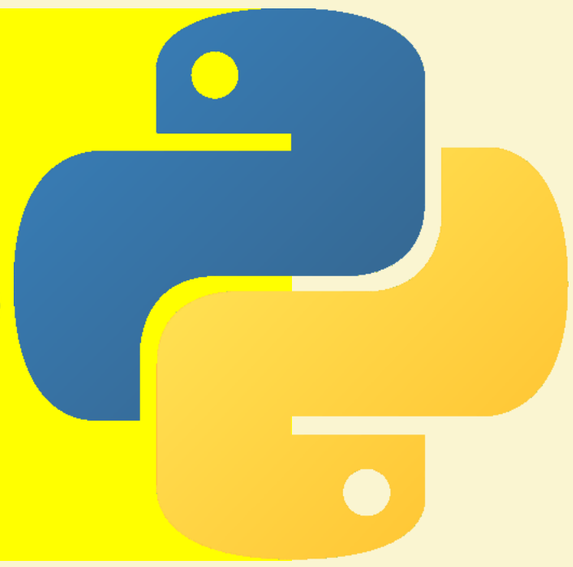
POO - Métodos Magicos



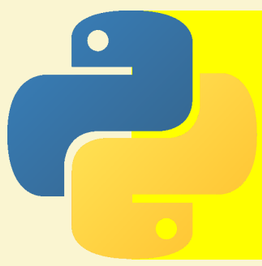
```
class MyClass:
def __init__(self):
    pass
def __str__(self):
    # Sobrescreve a função existente dopython
```

JOSÉ FABRÍCIO FIGUEIREDO

PYPDF2 / URLLIB



```
# PYPDF2: União de dois ou mais pdfs.
from PyPDF2 import PdfFileMerger

pdfs = ['file1.pdf', 'file2.pdf', ...]
merger = PdfFileMerger()
for pdf in pdfs:
    merger.append(pdf)

merger.write('result.pdf')
merger.close()

# URLLIB: Leitura do HTML da página na Web.
import urllib.request

pagina = urllib.request.urlopen("https://")
texto = pagina.read().decode("utf8")
print(texto
```



JOSÉ FABRÍCIO FIGUEIREDO