SENAI CIMATEC

SISTEMA FIEB

Introdução à linguagem Python

Disciplina: Programação com Python

Prof. Braian Varjão

Agenda

1. Características da linguagem;
2. Operadores;
3. Variáveis;
4. Manipulação de strings;
5. Estruturas de controle;
6. Estruturas de repetição;
7. Listas;
8. Dicionários;
9. Funções;
10. Classes, objetos e módulos;
11. Tratamento de exceções.

Características da linguagem

Características (1/4)

Linguagem de programação livre;

É uma linguagem interpretada (como Perl, Shell script, etc.), ou seja, não é necessário compilar o programa;

É uma linguagem Orientada a Objeto

- Tudo em Python é um objeto.

Características (2/4)

É uma linguagem de aprendizado fácil, com sintaxe clara e concisa;

É uma linguagem com tipagem forte, porém dinâmica;

É uma linguagem *case sensitive*;

Características (3/4)

Estrutura simples

Java

```java
public class Main {
  public static void main(String[] args) {
      System.out.println("hello world");
    }
}
```

Python

```python
print("hello world");
```

Características (4/4)

Oferece ferramentas para:

- Programação funcional;
- Processamento de imagens;
- Interface gráfica;
- Processamento distribuído;
- Integração com C e C#;
- Ciência e análise de dados.

Operadores

Operadores aritméticos

Operador	Descrição	Exemplo
+	adição	2 + 3 (5)
-	subtração	2 - 2 (0)
*	multiplicação	2 * 3 (6)
/	divisão	5 / 4 (1.25)
//	divisão inteira	4 / 3 (1)
**	exponenciação	2 ** 2 (4)
%	resto da divisão (mod)	10 % 2 (0)

Operadores de atribuição

Operador	Descrição	Exemplo
=	igual	x = 5 (5)
-=	menos igual	x -= 2 (3)
*=	vezes igual	x *= 3 (9)
/=	dividido igual	x /= 2 (4.5)
//=	dividido igual (inteiro)	x //= 2 (2.0)
**	potência igual	x **= 3 (8.0)
%	resto igual	x %= 3 (2.0)

Operadores lógicos

Operador	Descrição	Exemplo
==	igual	2 == 2
!=	diferente	2 != 3
<	menor	2 < 3
<=	menor ou igual	5 <= 5
>	maior	4 > 3
>=	maior ou igual	2 >= 1
is	identidade	2 is 2
and	e	2 < 3 and 3 < 5
or	ou	2 < 3 or False
in	contido	1 in [1,2,3]
not	negação	5 not in [1,2,3]

Variáveis

Nomenclatura de variáveis (1/3)

O que pode:

- x
- x100
- caixão
- VariávelComNomeGigantescoAcentoECaps
- linguagem_de_programação
- c0m_num3r0_m3i0

Nomenclatura de variáveis (2/3)

O que deve:

- nome_descritivo
- minusculo
- sem_acento
- com_numero_no_fim2
- separado_por_underline

Nomenclatura de variáveis (3/3)

O que não pode:

- 01numero_no_inicio
- caracteres_especiais$%&*){
- espaco entre palavras
- palavras reservadas (def, int, for...)

Atribuição múltipla

a = 1 e b = 2

Como trocar o valor dessas variáveis?

Atribuição múltipla

a = 1 e b = 2

Como trocar o valor dessas variáveis?

Outras linguagens	Python
aux = a a = b b = aux	a, b = b, a

Tipos de dados (1/2)

Execute as seguintes instruções

```
x = 3
type(x)

x /= 2
type(x)

x//=2
type(x)

x = int(x)
type(x)

x = 'fim'
type(x)
```

Tipos de dados (2/2)

Tipo	Descrição	Exemplo
int	inteiro	-1, 0, 1, 2...
long	inteiro longo	61111212454545545
bool	booleano	True ou False
float	real	1.3, 2.0, 5.348...
str	string (texto)	“texto1” ‘texto2’
complex	complexo	2 + 3j
tuple	tupla	(x,y) (x,y,z)
list	lista	[1,”dois”,True, 4.0]
dict	dicionário	{“key1”: ‘casa’, “key2”:4 }

Exercícios 1

Braian é um professor legal e quer evitar que o aluno João seja reprovado. João tirou 8,66, 5,35, 5 e 1, respectivamente, nas provas P1, P2, P3 e P4. Para isso, ele pode calcular a nota final usando a média aritmética (M.A.), média geométrica (M.G.) ou média harmônica (M.H.)

Qual dessas médias dá a maior nota para João? E qual das médias dá a pior nota?

$$M.A. = \frac{P_1 + P_2 + P_3 + P_4}{4}$$

$$M.G. = \sqrt[4]{|P_1 P_2 P_3 P_4|}$$

$$M.H. = \frac{4}{\frac{1}{P_1} + \frac{1}{P_2} + \frac{1}{P_3} + \frac{1}{P_4}}$$

Exercício 2

Josefa deseja fazer compras na China. Ela quer comprar um celular de U$ 299,99, uma chaleira de U$ 23,87, um gnomo de jardim de U$ 66,66 e 6 adesivos de unicórnio de U$ 1,42 cada um. O frete para Salvador é U$12,34. Considerando o dólar a R$ 5,28 (:/), responda:

a. Qual o preço final da compra em reais? Lembre-se que o valor do IOF é de 6,28%.
b. Quanto ela pagará apenas de IOF?

Manipulação de strings

Atribuição de strings (1/2)

Strings são tipos que armazenam uma sequência de caracteres.

```
>>> "Texto bonito"
'Texto bonito'
>>> "Texto com acentos de cedilhas: hoje é dia de caça!"
'Texto com acentos de cedilhas: hoje é dia de caça!'
```

As strings aceitam aspas simples também.

```
>>> nome = 'Silvio Santos'
>>> nome
'Silvio Santos'
```

Atribuição de strings (1/2)

Também é possível utilizar aspas simples, duplas e triplas no conteúdo de uma string.

```
>>> todas_as_aspas = """Essa é uma string que tem:
... - aspas 'simples'
... - aspas "duplas"
... - aspas '''triplas'''
... Legal né?"""
>>> todas_as_aspas
'Essa é uma string que tem:\n- aspas \'simples\'\n- aspas "duplas"\n- aspas \'\'\
↪'triplas\'\'\'\nLegal né?'
>>> print(todas_as_aspas)
Essa é uma string que tem:
- aspas 'simples'
- aspas "duplas"
- aspas '''triplas'''
Legal né?
```

Operações com strings

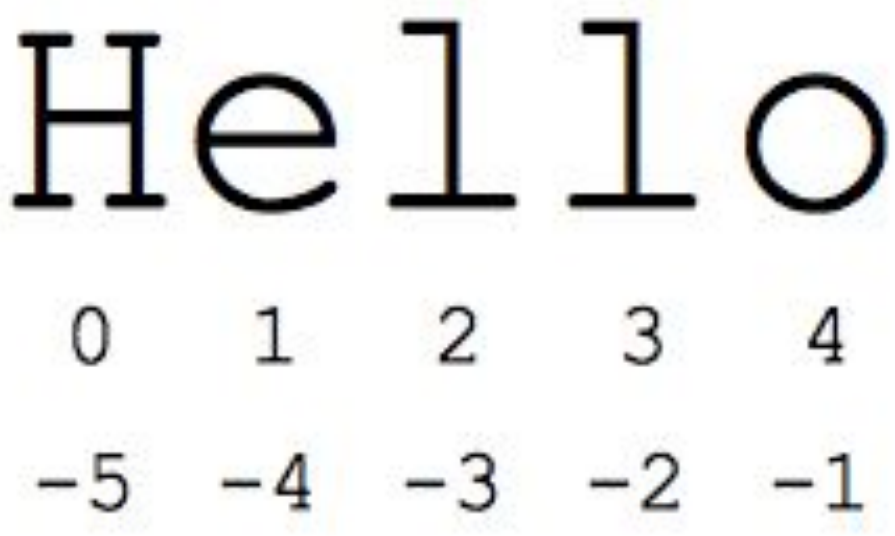
```
>>> nome * 3
'Silvio SantosSilvio SantosSilvio Santos'
>>> nome * 3.14
Traceback (most recent call last):
    ...
TypeError: can't multiply sequence by non-int of type 'float'
>>> canto1 = 'vem aí, '
>>> canto2 = 'lá '
>>> nome + ' ' + canto1 + canto2 * 6 + '!!'
'Silvio Santos vem aí, lá lá lá lá lá lá !!'
```

Tamanho de uma string

```
>>> help(len)
Help on built-in function len in module builtins:

len(obj, /)
    Return the number of items in a container.
```

Índices (1/2)



Índices (2/2)

```
>>> palavra = 'Python'
>>> palavra[0] # primeira
'P'
>>> palavra[5] # última
'n'
>>> palavra[-1] # última também
'n'
>>> palavra[-3] # terceira de tras pra frente
'h'
```

Fatiamento de strings (1/2)

frase = "Aprender Python é muito divertido!"

Execute

- frase[0]
- frase[5]
- frase[0:5]
- frase[:]
- frase[6:]
- frase[:6]
- frase[2:-3]
- frase[0:-7]
- frase[2:-2]

Fatiamento de strings (2/2)

É também possível controlar o passo que a fatia usa.

<fatiável>[<começo>:<fim>:<passo>]

```
>>> frase[::1] # do começo, até o fim, de 1 em 1. Ou seja, tudo do jeito que ja era,␣
↪não faz diferença nenhuma.
'Aprender Python é muito divertido!'
>>> frase[::2] # do começo, até o fim, de 2 em 2
'Arne yhnémiodvrio
>>> frase[2:-2:2] # Do terceiro, até o ante penúltimo, de 2 em dois
'rne yhnémiodvri'
```

Separação de strings

```
]    1 help(str.split)
```

```
Help on method_descriptor:

split(...)
    S.split(sep=None, maxsplit=-1) -> list of strings

    Return a list of the words in S, using sep as the
    delimiter string.  If maxsplit is given, at most maxsplit
    splits are done. If sep is not specified or is None, any
    whitespace string is a separator and empty strings are
    removed from the result.
```

Separação de strings

```
[ ]
1 frase = "texto separado por espaço."
2 texto = "Parágrafo 1. Parágrafo 2. Parágrafo 3."
3
4 palavras = frase.split()
5 paragrafos = texto.split(".")
6
7 print(palavras)
8 print(paragrafos)
```

```
['texto', 'separado', 'por', 'espaço.']
['Parágrafo 1', ' Parágrafo 2', ' Parágrafo 3', '']
```

Formatação de strings

```
[1]  1 "{} é professor de {} da turma {}.".format("Braian", "Python", 2)
     2
```

```
'Braian é professor de Python da turma 2.'
```

A formatação pode ser aplicada tanto ao texto quanto à variável que o contém.

```
[11] 1 resposta = "{} possui {} anos."
     2
     3 nome = input("Informe seu nome: ")
     4 idade = int(input("Informe sua idade (somente números): "))
```

```
Informe seu nome: MARIA
Informe sua idade (somente números): 47
```

```
[12] 1 print(resposta.format(nome,idade))
```

```
MARIA possui 47 anos.
```

Exercício 3

1. Dada a frase “Python é muito legal.”, use fatiamento para dividir a frase e armazenar cada palavra em uma variável diferente.
2. Agora que conhecemos atribuição múltipla e o método str.split() refaça o exercício anterior usando essas técnicas.
3. Qual o tamanho dessa frase?
4. Use fatiamento (mais especificamente o passo do fatiamento) para inverter a string “Python”.

Listas

Declaração

<nome_variavel> = [valor1, valor2, ..., valorN]

```
>>> nomes_frutas = ["maçã", "banana", "abacaxi"]
>>> nomes_frutas
['maçã', 'banana', 'abacaxi']

>>> numeros = [2, 13, 17, 47]
>>> numeros
[2, 13, 17, 47]
```

Declaração

Uma lista pode conter elementos de diferentes tipos de dados.

```
>>> ['lorem ipsum', 150, 1.3, [-1, -2]]
['lorem ipsum', 150, 1.3, [-1, -2]]

>>> vazia = []
>>> vazia
[]
```

Índices

lista = [100, 200, 300, 400, 500]

Execute

- lista[2]
- lista[-1]
- lista[:]
- lista[:3]
- lista[2:]
- lista[2:4]
- lista[2:-1]
- lista[::2]
- lista[10]

Checar existência de elemento na lista

```
>>> lista_estranha = ['duas palavras', 42, True, ['batman', 'robin'], -0.84, 'hipófise
↪']
>>> 42 in lista_estranha
True

>>> 'duas palavras' in lista_estranha
True

>>> 'dominó' in lista_estranha
False

>>> 'batman' in lista_estranha[3]  # note que o elemento com índice 3 também é uma␣
↪lista
True
```

Encontrar o índice de um elemento

```
1 lista = ['dois', 42, True, ['B1', 'B2'], 42, -0.84, 'hipófise']
2 lista.index(42)
```

```
1
```

E se eu quiser o índice da última ocorrência de 42?

Encontrar o índice de um elemento

```
1 lista = ['dois', 42, True, ['B1', 'B2'], 42, -0.84, 'hipófise']
2 lista.index(42)
```

```
1
```

E se eu quiser o índice da última ocorrência de 42?

```
[ ]  1 len(lista) - lista[::-1].index(42) - 1
```

```
4
```

Tamanho da lista

```
>>> len(lista)
5

>>> len(lista_estranha)
6

>>> len(lista_estranha[3])
2
```

Remoção de elementos

```
>>> lista_estranha
['duas palavras', 42, True, ['batman', 'robin'], -0.84, 'hipófise']

>>> del lista_estranha[2]
>>> lista_estranha
['duas palavras', 42, ['batman', 'robin'], -0.84, 'hipófise']

>>> del lista_estranha[-1]   # Remove o último elemento da lista
>>> lista_estranha
['duas palavras', 42, ['batman', 'robin'], -0.84]
```

Inserindo elementos

Método append():

```
>>> lista = ['a', 'b', 'c']
>>> lista
['a', 'b', 'c']

>>> lista.append('e')
>>> lista
['a', 'b', 'c', 'e']
```

Método insert():

```
>>> lista.insert(3, 'd')
>>> lista
['a', 'b', 'c', 'd', 'e']
```

Ordenação

```
1 lista_desordenada = ['b', 'z', 'k', 'a', 'h']
2 lista_desordenada.sort()
3 lista_desordenada
```

```
['a', 'b', 'h', 'k', 'z']
```

Operações com listas

```
[13]  1 a = ['a', 'b', 'c']
      2 b = ['d', 'e']
      3 a+b
```

```
['a', 'b', 'c', 'd', 'e']
```

```
[14]  1 (a+b)*2
```

```
['a', 'b', 'c', 'd', 'e', 'a', 'b', 'c', 'd', 'e']
```

Copiar uma lista

lista = ['dois', 42, True, ['B1', 'B2'], -0.84, 'hipófise']

Como remover um item da lista e manter um backup em memória?

Copiar uma lista

lista = ['dois', 42, True, ['B1', 'B2'], -0.84, 'hipófise']

Como remover o 4º item da lista e manter um backup em memória?

```
[ ]
1 lista = ['dois', 42, True, ['B1', 'B2'], -0.84, 'hipófise']
2 backup = lista
3 del lista[3]
4 lista
```

```
['dois', 42, True, -0.84, 'hipófise']
```

E o backup?

```
[ ]
1 backup
```

```
['dois', 42, True, -0.84, 'hipófise']
```

Copiar uma lista

lista = ['dois', 42, True, ['B1', 'B2'], -0.84, 'hipófise']

Como remover o 4º item da lista e manter um backup em memória?

```
[ ]  1 lista = ['dois', 42, True, ['B1', 'B2'], -0.84, 'hipófise']
     2 backup = lista
     3 del lista[3]
     4 lista
```

```
['dois', 42, True, -0.84, 'hipófise']
```

E o backup?

```
[ ]  1 backup
```

```
['dois', 42, True, -0.84, 'hipófise']
```

Copiar uma lista - copy()

```
[2]  1 lista = ['dois', 42, True, ['B1', 'B2'], -0.84, 'hipófise']
     2 backup = lista.copy()
     3 del lista[3]
     4 lista
```

```
['dois', 42, True, -0.84, 'hipófise']
```

E o backup?

```
[3]  1 backup
```

```
['dois', 42, True, ['B1', 'B2'], -0.84, 'hipófise']
```

Exercício 4

Dada a lista = [12, -2, 4, 8, 29, 45, 78, 36, -17, 2, 12, 8, -52] faça um programa que:

a) imprima o maior e o menor elemento;

c) imprima a lista invertida;

d) imprima os 5 maiores elementos, com exceção do maior de todos;

e) imprima a média dos elementos (pesquisem);

f) remova o elemento de menor valor;

g) insira um elemento de valor -500 na terceira posição e outro de valor 500 no final da lista.

Dicionários

Declaração

```
1 telefones = {"ana": 123456, "yudi": 40028922, "julia": 4124492}
2 telefones
```

```
{'ana': 123456, 'julia': 4124492, 'yudi': 40028922}
```

```
[ ]
1 constantes = dict(pi=3.14, e=2.7, alpha=1/137)
2 constantes
```

```
{'alpha': 0.007299270072992700 5, 'e': 2.7, 'pi': 3.14}
```

```
1 numeros = dict({1: "um", 2: "dois", 3: "três"})
2 numeros
```

```
{1: 'um', 2: 'dois', 3: 'três'}
```

Acessando valores

```
[2]  1 capitais = {"SP": "São Paulo", "AC": "Rio Branco", "TO": "Palmas",
     2             "RJ": "Rio de Janeiro", "SE": "Aracaju",
     3             "MG": "Belo Horizonte"}
```

```
[3]  1 capitais["MG"]
```

```
'Belo Horizonte'
```

```
[4]  1 capitais[1]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-4-fb9d828b65a7> in <module>()
----> 1 capitais[1]

KeyError: 1
```

Adicionar, editar e remover elementos

```
[16]  1 pessoa = {"nome": "Cleiton", "idade": 34}
      2 #adicionando
      3 pessoa["pais"] = {"mãe": "Maria", "pai": "Enzo"}
      4 pessoa
```

```
{'idade': 34, 'nome': 'Cleiton', 'pais': {'mãe': 'Maria', 'pai': 'Enzo'}}
```

```
[17]  1 #editando
      2 pessoa["idade"] = 35
      3 pessoa["idade"]
```

```
35
```

```
[18]  1 #excluindo
      2 del pessoa["idade"]
      3 pessoa
```

```
{'nome': 'Cleiton', 'pais': {'mãe': 'Maria', 'pai': 'Enzo'}}
```

```
[19]  1 #limpando o dicionário
      2 pessoa.clear()
      3 pessoa
```

```
{}
```

Método get()

```
[21]  1 meses = {1: "Janeiro", 2: "Fevereiro", 3: "Março", 4: "Abril"}
      2 meses[1]
```

```
'Janeiro'
```

```
[22]  1 meses.get(1)
```

```
'Janeiro'
```

```
[23]  1 meses[6]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-23-97ac25217209> in <module>()
----> 1 meses[6]

KeyError: 6
```

SEARCH STACK OVERFLOW

```
      1 meses.get(6, "Não tem!")
```

```
'Não tem!'
```

Exercício 5

1. Faça um programa utilizando um `dict` que leia dados de entrada do usuário. O usuário deve entrar com os dados de uma pessoa como nome e cidade onde mora (fique livre para acrescentar outros). Após isso, você deve imprimir os dados como o exemplo abaixo:

```
nome: João
cidade: São Paulo
```

1. Utilize o exercício anterior e adicione a pessoa em uma lista. Em seguida, cadastre uma nova pessoa nessa lista. Por fim, você deve imprimir todos os dados de cada pessoa cadastrada.

Estruturas de controle

Instrução IF

```
if <condição lógica>:
    <bloco de instrução>
```

Instrução IF

```
if <condição lógica>:
    <bloco de instrução>
```

ATENÇÃO

Em python os blocos de instrução são identificados pela indentação, não se utiliza colchete ou chaves como em outras linguagens de programação.

Por esse motivo, códigos indentados incorretamente resultam em erros de execução.

Instrução IF

```
if <condição lógica>:
    <bloco de instrução>
elif <condição lógica>:
    <bloco de instrução>
else:
    <bloco de instrução>
```

if e elif

```
>>> a = 7
>>> if a > 3:
...     print("estou no if")
... else:
...     print("cai no else")
...
estou no if
```

if e elif

```
>>> valor_entrada = 10
>>> if valor_entrada == 1:
...     print("a entrada era 1")
... elif valor_entrada == 2:
...     print("a entrada era 2")
... elif valor_entrada == 3:
...     print("a entrada era 3")
... elif valor_entrada == 4:
...     print("a entrada era 4")
... else:
...     print("o valor de entrada não era esperado em nenhum if")
...
o valor de entrada não era esperado em nenhum if
```

if inline

```
1 num = 10
2 if num >= 0:
3     print("positive")
4 else:
5     print("negative")
```

```
positive
```

```
[ ] 1 print("positive") if num >= 0 else print("negative")
```

```
positive
```

A **if** <condição> **else** B

Exercício 6

1. Faça um programa que, dada a idade, peso e horas de sono nas últimas 24h de uma pessoa, determine se ela pode ou não doar sangue. Os requisitos são obrigatórios para doação são:
   a. Ter entre 16 e 69 anos;
   b. Pesar mais de 50kg;
   c. Estar descansado (6 horas de sono nas últimas 24 horas).
2. Determine, dados os coeficientes de uma equação de segundo grau, se esta possui duas raízes, uma ou se não possui.

   Dica: $\Delta = b^2 - 4 \cdot a \cdot c$ ta é maior que 0, possui duas raízes reais; se delta é 0, possui uma raíz, caso delta seja menor que 0, não possui raiz real.

Estruturas de repetição

Laço FOR

```
for <instância> in <iterador>:
    <bloco de instrução>
```

Laço FOR

```
[6]  1 lista = ['a', 1, 'b', 2]
     2
     3 for x in lista:
     4   print(x, end=" ")
```

```
a 1 b 2
```

```
[8]  1 for i in list(range(1,10)):
     2   print(i, end=" ")
```

```
1 2 3 4 5 6 7 8 9
```

```
[9]  1 gatinhos = {"Português": "gato", "Inglês": "cat"}
     2
     3 for chave, valor in gatinhos.items():
     4   print(chave + ": " + valor)
```

```
Português: gato
Inglês: cat
```

Exercício 7

1. Solicite que o usuário informe um valor N inteiro e apresente a tabuada de N. N deve ser um valor entre 0 e 10 (incluso), se o valor informado estiver fora destes limites, deve ser solicitado um novo valor.

1. Dada a lista [2, 8, 3, 13, 7, 24, “str”, [1, 2, 11], 8, True], exiba todos os números primos existentes em seu conteúdo em ordem crescente (incluindo a lista interna).

Laço WHILE

while <condição lógica de parada>**:**

    <bloco de instrução>

Laço While

```
>>> while True:
...     string_digitada = input("Digite uma palavra: ")
...     if string_digitada.lower() == "sair":
...         print("Fim!")
...         break
...     if len(string_digitada) < 2:
...         print("String muito pequena")
...         continue
...     print("Tente digitar \"sair\"")
...
Digite uma palavra: oi
Tente digitar "sair"
Digite uma palavra: ?
String muito pequena
Digite uma palavra: sair
Fim!
```

Exercício 8

1. Escreva um algoritmo que receba dados cadastrais de pessoas (nome, idade e sexo). Ao fim de cada cadastro, o usuário deve ser questionado se deseja realizar um novo cadastro. Quando não houverem mais cadastros a serem realizados, exibir os cadastros realizados e o percentual de homens e mulheres.

2. Imprima todos os valores da sequência de fibonacci inferiores a 100.

List comprehension

[<expressão> **for** item **in** lista]

```
[24]  1 lista1, lista2 = [], []
```

```
[25]  1 for x in range(10):
      2   lista1.append(x**2)
```

```
[26]  1 lista2 = [x**2 for x in range(10)]
```

```
      1 print(lista1, end=" ")
      2 print(lista2, end=" ")
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81] [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

List comprehension + if

[<expressão> **for** item **in** lista **if** <condição>]

Seleção de números ímpares

```
[40]  1 lista1, lista2 = [], []
```

```
[41]  1 for x in range(10):
      2   if x % 2 == 0:
      3     lista1.append(x)
```

```
[42]  1 lista2 = [numero for numero in range(10) if numero % 2 == 0]
```

```
[43]  1 print(lista1, end=" ")
      2 print(lista2, end=" ")
```

```
[0, 2, 4, 6, 8] [0, 2, 4, 6, 8]
```

List comprehension + if else

[<resul_if > **if** <condição> **else** <result_else > **for** item **in** lista]

```
[63]  1 resultado = []
      2 for numero in range(16):
      3   if numero % 5 == 0:
      4     resultado.append('1')
      5   else:
      6     resultado.append('0')
      7
      8 print(resultado)
```

```
['1', '0', '0', '0', '0', '1', '0', '0', '0', '0', '1', '0', '0', '0', '0', '1']
```

```
1 resultado = ['1' if numero % 5 == 0 else '0' for numero in range(16)]
2
3 print(resultado)
```

```
['1', '0', '0', '0', '0', '1', '0', '0', '0', '0', '1', '0', '0', '0', '0', '1']
```

List comprehension (aninhadas)

```
1  transposta = []
2  matriz = [[1, 2, 3, 4], [4, 5, 6, 8], [9, 10, 11, 12]]
3
4  for i in range(len(matriz[0])):
5      linha_transposta = []
6
7      for linha in matriz:
8          linha_transposta.append(linha[i])
9      transposta.append(linha_transposta)
10
11 print(transposta)
```

```
[[1, 4, 9], [2, 5, 10], [3, 6, 11], [4, 8, 12]]
```

[52]

```
1  transposta = [[linha[i] for linha in matriz] for i in range(len(matriz[0]))]
2  transposta
```

```
[[1, 4, 9], [2, 5, 10], [3, 6, 11], [4, 8, 12]]
```

Exercício 9

1. Utilizando list comprehension escreva um algoritmo receba a frase “três pratos de trigo para três tigres tristes” e gere uma lista com todas as palavras que não começam com a letra t.

2. Utilizando list comprehension, escreva um algoritmo que transforme a lista [34.6, -203.4, 44.9, 68.3, -12.2, 44.6, 12.7] em uma lista de inteiros positivos.

Funções

Definição e chamada

```
def NOME_DA_FUNÇÃO(LISTA DE PARÂMETROS):
    COMANDOS
```

```
>>> def soma():
...     print(1 + 1)
...
>>> soma()
2

>>> def soma():
...     return 1 + 1
...
>>> soma()
2
```

Função com parâmetros

```
[1]
1 def concatena(texto1 = "faltou 1", texto2 = "faltou 2"):
2   print(texto1 + " " + texto2)
3
4 concatena("foi 1", "foi 2")
5 concatena("foi 1")
6 concatena(texto2="foi 2")
7 concatena()
```

```
foi 1 foi 2
foi 1 faltou 2
faltou 1 foi 2
faltou 1 faltou 2
```

Função com retorno

```
[6]
1 def sqrt(x):
2   return x**(1/2)
3
4 x = 4
5 print("A raíz quadrada de {} é {}.".format(x, sqrt(x)))
```

```
A raíz quadrada de 4 é 2.0.
```

Função com múltiplos retornos

```python
def raiz_e_quadrado(x):
  return x**(1/2), x**2

x = 4
retorno = raiz_e_quadrado(x)
print(type(retorno))

raiz, quadrado = retorno

print(raiz == retorno[0])
print(quadrado == retorno[1])

print("{}^1/2 = {}, {}^2 = {}.".format(x, raiz, x, quadrado))

```

```
<class 'tuple'>
True
True
4^1/2 = 2.0, 4^2 = 16.
```

Exercício 10

1. Dada a função y = 5x + 2, determine os valores de y para x entre -10 e 10, onde x é inteiro.a

2. O exercício 7 pediu que fosse feito um programa que, Dada a lista [2, 8, 3, 13, 7, 24, “str”, [1, 2, 11], 8, True], exiba todos os números primos existentes em seu conteúdo em ordem crescente (incluindo a lista interna).

   Refaça esse exercício utilizando funções. Seu programa deve estar apto a lidar com qualquer lista que possua elementos de qualquer tipo de dado. Se o elemento da lista for um float, converta-o para inteiro, se for uma lista, inclua seus elementos na verificação, se for outro tipo de dado (como string ou bool), este deverá ser desconsiderado. Lembrem-se que podem existir listas dentro de listas.

Classes, objetos e módulos

Classe

Crie um arquivo python com o seguinte código e o salve com o nome modulo.py

```
class Potencia:

  def __init__(self, x):
    self.x = x

  def quadrado(self):
    print("{}**2 = {}".format(self.x, self.x**2))

  def raiz(self):
    print("{}**(1/2) = {}".format(self.x, self.x**(1/2)))

  def custom(self, y):
    print("{}**{} = {}".format(self.x, y,(self.x**y)))
```

Módulo

from <nome_do_modulo> **import** <classe>, <método>

import nome_do_modulo

from <nome_do_modulo> **import** *

Módulo

Acrescente o código abaixo no seu arquivo modulo.py

```
1 def soma(x, y):
2     print("{} + {} = {}".format(x, y, x+y))
3
4 def subtracao(x, y):
5     print("{} - {} = {}".format(x, y, x-y))
```

Objeto

Execute o código abaixo em seu notebook jupyter

```
from modulo import Potencia
x = Potencia(4)
x.quadrado()
x.raiz()
x.custom(2)
```

Desafio

Construa um módulo conta_bancaria que possui uma classe Cliente e outra classe Extrato.

A classe **Extrato** guarda a data de criação da conta e uma lista de mensagens informando as transações já realizadas.

Funções: adicionar_transacao e gerar_extrato

A classe **Cliente** possui os atributos nome, agencia, conta, saldo e extrato (do tipo Extrato)

Os atributos são inicializados na criação do objeto;

Funções: sacar, depositar, exibir_saldo, transfererir e exibir_extrato.

Lembre-se que o usuário não pode gastar mais do que tem.

Tratamento de exceções

Syntax error x Exception error

```
>>> print( 0 / 0 ))
  File "<stdin>", line 1
    print( 0 / 0 ))
                  ^
SyntaxError: invalid syntax


>>> print( 0 / 0)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: integer division or modulo by zero
```

Estrutura básica

```
1 while True:
2   try:
3       x = int(input("Por favor, informe um número: "))
4       break
5   except ValueError:
6       print("Oops!  Isso não é um número válido.  Tente novamente...")
```

Múltiplos tratadores

```
1 class B(Exception):
2     pass
3
4 class C(B):
5     pass
6
7 class D(C):
8     pass
9
10 for cls in [B, C, D]:
11     try:
12         raise cls()
13     except D:
14         print("D")
15     except C:
16         print("C")
17     except B:
18         print("B")
```

Exceção curinga

```
1 import sys
2
3 try:
4     f = open('myfile.txt')
5     s = f.readline()
6     i = int(s.strip())
7 except OSError as err:
8     print("Erro de SO: {0}".format(err))
9 except ValueError:
10    print("Dado não pôde ser convertido para inteiro.")
11 except:
12    print("Erro inesperado:", sys.exc_info()[0])
13    raise
```

Recuperando mensagem de erro padrão

```
1 try:
2   x = 1/0
3 except ZeroDivisionError as err:
4   print('Erro de execução:', err)
```

Cláusula else

```
1 try:
2   checar_so()
3 except OSError as err:
4   print("Erro de SO: {0}".format(erro))
5 else:
6   try:
7     f = open('myfile.txt')
8   except FileNotFoundError:
9     print("Arquivo não encontrado.")
10
```

Cláusula finally

```
1 try:
2   checar_so()
3 except OSError as err:
4   print("Erro de SO: {0}".format(erro))
5 else:
6   try:
7     f = open('myfile.txt')
8   except FileNotFoundError:
9     print("Arquivo não encontrado.")
10 finally:
11   print("Funcionando ou não, sempre passamos por aqui.")
12
```

Dúvidas?