Capítulo 4 (DSA)

domingo, 14 de fevereiro de 2021 21:32

Arquivos

Método	Utilização
open()	Usada para abrir o arquivo
read()	Leitura do arquivo
write()	Gravação no arquivo
seek()	Retorna para o início do arquivo
readlines()	Retorna a lista de linhas do arquivo
close()	Fecha o arquivo

- **Abrindo um arquivo para leitura (read)**

arquivo = open("caminho/nome.txt", "r")

```
arq = open("C:/Users/Mochi/Documents/Python-DS/teste.txt","r",encoding='utf-8')
```

arq.read()

```
print(arq.read())
```

```
Olá este arquivo foi gerado pelo próprio Jupyter Notebook.
Podemos gerar quantas linhas quisermos e o Jupyter gera o arquivo final.
```

arq.read(n) → Lê uma quantidade n de caracteres.

```
print(arq.read(10))
```

```
Olá este a
```

arq.tell() → Conta o número de cracteres no arquivo.

```
print(arq.tell())
```

```
134
```

arq.seek(0,0) → Vai para um ponto específico do arquivo.
0,0 = primeira linha e primeira coluna do arquivo.

- **Abrindo um arquivo para escrita (write)**

arquivo = open("caminho/nome.txt", "w")

```
arq2 = open("arquivos/arquivo1.txt", "w")
```

arq.write("Frase que quero gravar")

```
arq2.write("Testando gravação de arquivos em Python ")
```

- **Abrindo um arquivo para acrescentar (append)**

arquivo = open("caminho/nome.txt", "a")

```
arq2 = open("arquivos/arquivo1.txt", "a")
```

Com o arquivo aberto no modo append eu posso acrescentar com o .write mesmo:

```
arq2.write(" Acrescentando conteúdo")
```

- É importante que no final das operações feche o arquivo

arquivo.close()

```
arq2.close()
```

Exemplo 1 - Separando o arquivo em linhas

```
f = open('arquivos/salarios.csv', 'r')
```

Abre o arquivo salarios.csv

```
data = f.read()
```

Salva seu conteúdo numa variável data

```
rows = data.split('\n')
```

Em uma variável chamada rows, terei uma lista do meu arquivo separado por linhas. Ou seja, do início da linha até a quebra de linha ('\n') tenho um termo da minha lista.

```
print(rows)
```

```
['Name,Position Title,Department,Employee Ann
R MGMNT,$88967.00', '"AARON,  JEFFERY M",POLI
E OFFICER,POLICE,$80778.00', '"AARON,  KIMBER
```

Exemplo 2 - Separando o arquivo em colunas

```
f = open('arquivos/salarios.csv', 'r')
```

Abre o arquivo salarios.csv

```
data = f.read()
```

Salva seu conteúdo numa variável data

```
rows = data.split('\n')
```

Cria uma lista rows, onde meus dados estão separados por linha

```
full_data = []
```

Cria uma lista vazia chamada full_data

```
for row in rows:
    split_row = row.split(",")
    full_data.append(split_row)
```

Percorro cada linha e divido essa linha pelas vírgulas e adiciono os dados na lista full_data.
Ou seja, a full_data, ao final, me trará uma lista onde cada linha do arquivo salarios.csv é uma lista, e os termos deste csv separados por vírgula, são os termos desta minha lista de linhas que está dentro da lista full_data.

```
print(full_data)
```

```
[['Name', 'Position Title', 'Department',
ER RATE TAKER', 'WATER MGMNT', '$88967.00'
E', '$80778.00'], ['"AARON', '  KARINA"',
```

Exemplo 3 - Contando o número de linhas

```
f = open('arquivos/salarios.csv', 'r')
```

```
data = f.read()
```

```
rows = data.split('\n')
```

```
full_data = []
```

```
f = open('arquivos/salarios.csv', 'r')
data = f.read()
rows = data.split('\n')
full_data = []
for row in rows:
    split_row = row.split(",")
    full_data.append(split_row)
count = 0
for row in full_data:
    count += 1
```

```
full_data = []
```

```
for row in rows:
    split_row = row.split(",")
    full_data.append(split_row)
```

```
count = 0
for row in full_data:
    count += 1   # Equivalente a: count = count + 1
```

```
print(count)
```

```
32184
```

```
    full_data.append(split_row)
count = 0
for row in full_data:
    count += 1
print(count)
```

Exemplo 4 - Contando o número de colunas

```
f = open('arquivos/salarios.csv', 'r')
data = f.read()
rows = data.split('\n')
full_data = []
```

```
for row in rows:
    split_row = row.split(",")
    full_data.append(split_row)
    first_row = full_data[0]
count = 0
```

```
for column in first_row:
    count = count + 1

# Outra solução possível
# for column in full_data[0]:
#     count = count + 1
```

```
print(count)
```

```
4
```

```
f = open('arquivos/salarios.csv', 'r')
data = f.read()
rows = data.split('\n')
full_data = []
for row in rows:
    split_row = row.split(",")
    full_data.append(split_row)
    first_row = full_data[0]
count = 0
for column in first_row:
    count = count + 1
print(count)
```

Automatização do processo de gravação

quarta-feira, 24 de fevereiro de 2021 14:15

```
fileName = input("Digite o nome do arquivo: ")
```

```
fileName = fileName + ".txt"
```

```
arq3 = open(fileName, "w")
```

```
arq3.write("Incluindo texto no arquivo criado")
```

```
arq3.close()
```

```
arq3 = open(fileName, "r")
```

```
print(arq3.read())
```

```
arq3.close()
```

```
fileName = input("Digite o nome do arquivo: ")
fileName = fileName + ".txt"
arq3 = open(fileName, "w")
arq3.write("Incluindo texto no arquivo criado")
arq3.close()
arq3 = open(fileName, "r")
print(arq3.read())
arq3.close()
```

import os
import csv
import panda as pd
import json

Pandas

quarta-feira, 24 de fevereiro de 2021 17:42

Chamando a biblioteca pandas

import pandas as pd

Abrindo um arquivo com pandas

```
import pandas as pd

file_name = "arquivos/binary.csv"

df = pd.read_csv(file_name)

df.head()
```

	admit	gre	gpa	rank
0	0	380	3.61	3
1	1	660	3.67	3
2	1	880	4.00	1
3	1	640	3.19	4
4	0	520	2.93	4

arquivo = "arquivos/planilha.csv"
df = pd.read_csv(arquivo)

TXT

quarta-feira, 24 de fevereiro de 2021 17:59

import os

- **Abrindo um arquivo TXT para escrita**

arquivo = open(os.path.join('nome_do_arquivo.txt'), 'w')

```
arquivo = open(os.path.join('arquivos/cientista.txt'),'w')
```

- **Abrindo um arquivo TXT para leitura**

arquivo = open('arquivos/nome_do_arquivo.txt', 'r')

```
arquivo = open('arquivos/cientista.txt','r')
conteudo = arquivo.read()
arquivo.close()

print(conteudo)
```

```
Cientista de Dados é a profissão que mais tem c
e especializar em Programação, Estatística e Ma
```

Usando o with

with open('arquivos/nome_do_arquivo.txt', 'r') as arquivo:
comando

```
with open('arquivos/cientista.txt','r') as arquivo:
    conteudo = arquivo.read()
```

```
texto = "Cientista de Dados é a profissão que mais tem crescido em todo mundo.\n"
with open('arquivos/cientista.txt','w') as arquivo:
    arquivo.write(texto[:21])
    arquivo.write('\n')
    arquivo.write(texto[:33])
arquivo = open('arquivos/cientista.txt','r')
conteudo = arquivo.read()
arquivo.close()

print (conteudo)
```

```
Cientista de Dados é
Cientista de Dados é a profissão
```

CSV

quarta-feira, 24 de fevereiro de 2021 19:11

import csv

- **Abrindo um arquivo CSV para escrita**

  with open('arquivos/nome_do_arquivo.csv', 'w') as arquivo:
  csv.writer(arquivo)
  writer.writerow(('item1', 'item2'))

```
with open('arquivos/numeros.csv','w') as arquivo:
    writer = csv.writer(arquivo)
    writer.writerow(('primeira','segunda','terceira'))
    writer.writerow((55,93,76))
    writer.writerow((62,14,86))
```

- **Abrindo um arquivo CSV para leitura**

  with open('arquivos/nome_do_arquivo.csv', 'r') as arquivo:
  comando

```
with open('arquivos/numeros.csv','r') as arquivo:
    leitor = csv.reader(arquivo)
    for x in leitor:
        print ('Número de colunas:', len(x))
        print(x)
```

```
Número de colunas: 3
['primeira', 'segunda', 'terceira']
Número de colunas: 0
[]
Número de colunas: 3
['55', '93', '76']
Número de colunas: 0
[]
Número de colunas: 3
['62', '14', '86']
```

with open('arquivos/numeros.csv','r', encoding='utf8', newline = '\r\n') as arquivo:

```
# Código alternativo para eventuais problemas com linhas em branco no arquivo
with open('arquivos/numeros.csv','r', encoding='utf8', newline = '\r\n') as arquivo:
    leitor = csv.reader(arquivo)
    for x in leitor:
        print ('Número de colunas:', len(x))
        print(x)
```

```
Número de colunas: 3
['primeira', 'segunda', 'terceira']
Número de colunas: 3
['55', '93', '76']
Número de colunas: 3
['62', '14', '86']
```

Gerando uma lista com os dados do csv

```
with open('arquivos/numeros.csv','r') as arquivo:
    leitor = csv.reader(arquivo)
    dados = list(leitor)


print (dados)
```

```
[['primeira', 'segunda', 'terceira'], [], ['55', '93', '76'], [], ['62', '14', '86'], []]
```

Imprimindo a partir da segunda linha

```
for linha in dados[1:]:
    print (linha)
```

```
['55', '93', '76']
['62', '14', '86']
```

Json

quarta-feira, 24 de fevereiro de 2021 19:18

import json

Criando um dicionário

```
dict = {'nome': 'Guido van Rossum',
        'linguagem': 'Python',
        'similar': ['c','Modula-3','lisp'],
        'users': 1000000}
```

Convertendo o dicionário para json

```
json.dumps(dict)
```

```
'{"nome": "Guido van Rossum", "linguagem": "Python", "similar": ["c", "Modula-3", "lisp"], "users": 10
00000}'
```

Abrindo um arquivo json para escrita

```
with open('arquivos/nome_do_arquivo.json', 'w') as arquivo:
    arquivo.write(json.dumps(dict))
```

```
with open('arquivos/dados.json','w') as arquivo:
    arquivo.write(json.dumps(dict))
```

Abrindo um arquivo json para leitura

```
with open('arquivos/nome_do_arquivo.json', 'r') as arquivo:
    texto = arquivo.read()
    data = json.loads(texto)
```

```
with open('arquivos/dados.json','r') as arquivo:
    texto = arquivo.read()
    data = json.loads(texto)
```

```
print (data)
```

```
{'nome': 'Guido van Rossum', 'linguagem': 'Python
```

```
print (data['nome'])
```

```
Guido van Rossum
```

Pegar um json da internet

from urllib.request import urlopen
response = urlopen("url").read().decode('utf8')

```
from urllib.request import urlopen

response = urlopen("http://vimeo.com/api/v2/video/57733101.json").read().decode('utf8')
data = json.loads(response)[0]
```

```
print ('Título: ', data['title'])
print ('URL: ', data['url'])
print ('Duração: ', data['duration'])
print ('Número de Visualizações: ', data['stats_number_of_plays'])
```

```
Título:  The Good Man trailer
URL:  https://vimeo.com/57733101
Duração:  143
Número de Visualizações:  5881
```

Copiar um arquivo json em um txt

arquivo_fonte = 'documentos/dados.json'
arquivo_destino = 'documentos/dados.txt'

```
arquivo_fonte = 'arquivos/dados.json'
arquivo_destino = 'arquivos/json_data.txt'
```

```
# Método 1
with open(arquivo_fonte,'r') as infile:
    text = infile.read()
    with open(arquivo_destino,'w') as outfile:
        outfile.write(text)
```

```
with open(arquivo_fonte,'r') as infile:
  text = infile.read()
  with open(arquivo_destino,'w') as outfile:
    outfile.write(text)
```

```
# Método 2
open(arquivo_destino,'w').write(open(arquivo_fonte,'r').read())
```

```
open(arquivo_destino,'w').write(open(arquivo_fonte,'r').read())
```

Módulos e Pacotes

quarta-feira, 24 de fevereiro de 2021 21:05

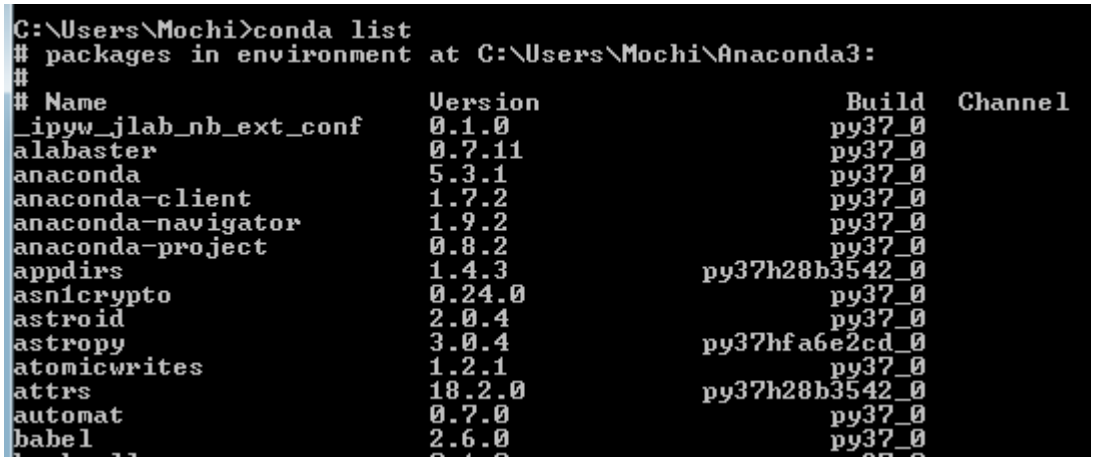
Módulos são arquivos Python (com extensão .py) que implementam um conjunto de funções.

- Importamos um módulo usando o comando import:

import math
from math import sqrt

- Para ver os pacotes que tenho instalado no anaconda

conda list



```
C:\Users\Mochi>conda list
# packages in environment at C:\Users\Mochi\Anaconda3:
#
# Name                    Version                   Build  Channel
_ipyw_jlab_nb_ext_conf    0.1.0                    py37_0
alabaster                 0.7.11                   py37_0
anaconda                  5.3.1                    py37_0
anaconda-client           1.7.2                    py37_0
anaconda-navigator        1.9.2                    py37_0
anaconda-project          0.8.2                    py37_0
appdirs                   1.4.3            py37h28b3542_0
asn1crypto                0.24.0                   py37_0
astroid                   2.0.4                    py37_0
astropy                   3.0.4            py37hfa6e2cd_0
atomicwrites              1.2.1                    py37_0
attrs                     18.2.0           py37h28b3542_0
automat                   0.7.0                    py37_0
babel                     2.6.0                    py37_0
```

- Para instalar um módulo novo

conda install "nome_do_modulo"

Pacotes são conjuntos de módulos Python.

import modulo
import pacote.modulo

Um módulo é um único arquivo Python.
Um pacote é um diretório de módulos Python, contém um arquivo __init__.py

Repositório de pacotes do Python: https://pypi.org/

Math

- Verificando os métodos disponíveis no módulo

```
import math

dir(math)
```

```
['__doc__',
 '__file__',
 '__loader__',
 '__name__',
 '__package__',
 '__spec__',
 'acos',
```

Utilizando o método sqrt do módulo math

```
math.sqrt(25)
```

```
5.0
```

Para tirar dúvidas

```
help(sqrt)
```

Random

import random
random.choice( )
random.sample( )

```
import random
```

```
random.choice(['Maça', 'Banana', 'Laranja'])
```

```
'Maça'
```

```
random.sample(range(100), 10)
```

```
[33, 21, 92, 23, 42, 65, 49, 73, 91, 36]
```

Statistics

import statistics
statistics.mean( )
statistics.median( )

```
import statistics
```

```
dados = [2.75, 1.75, 1.25, 0.25, 0.5, 1.25, 3.5]
```

```
statistics.mean(dados)
```

```
1.6071428571428572
```

```
statistics.median(dados)
```

```
1.25
```

Os

import os
os.getcwd( )

```
import os
```

```
os.getcwd()
```

```
'/Users/dmpm/Dropbox/DSA/PythonFundamentos/Cap04/Notebooks'
```

```
print(dir(os))
```

```
['CLD_CONTINUED', 'CLD_DUMPED', 'CLD_EXITED', 'CLD_TRAPPED', '
G', 'EX_DATAERR', 'EX_IOERR', 'EX_NOHOST', 'EX_NOINPUT', 'EX_N
R', 'EX_OSFILE', 'EX_PROTOCOL', 'EX_SOFTWARE', 'EX_TEMPFAIL',
K', 'F_OK', 'F_TEST', 'F_TLOCK', 'F_ULOCK', 'MutableMapping',
D', 'O_ASYNC', 'O_CLOEXEC', 'O_CREAT', 'O_DIRECTORY', 'O_DSYNC
```

Sys

import sys
sys.stdout.write( )
sys.version

```
import sys
```

```
sys.stdout.write('Teste')
```

```
Teste
```

```
sys.version
```

```
'3.7.6 (default, Jan  8 2020, 13:42:34) \n[Clang 4.0.1 (tags/RELEASE_401/final)]'
```

```
import urllib.request
```

```
# Variável resposta armazena o objeto de conexão à url passada como
# parâmetro
resposta = urllib.request.urlopen('http://python.org')
```

```
# Objeto resposta
print(resposta)
```

```
<http.client.HTTPResponse object at 0x1032028d0>
```

```
html = resposta.read()
```

```
# Imprimindo html
print(html)
```

```
b'<!doctype html>\n<!--[if lt IE 7
>\n<!--[if IE 7]>      <html class
```

Datetime

quarta-feira, 24 de fevereiro de 2021 22:08

import datetime

datetime.datetime.now( )
datetime.time( )
datetime.date( )

```
import datetime
```

```
agora = datetime.datetime.now()
```

```
agora
```

```
datetime.datetime(2021, 2, 24, 22, 8, 8, 370395)
```

```
t = datetime.time(7, 43, 28)
```

```
print (t)
```

```
07:43:28
```

```
print ('Hora  :', t.hour)
print ('Minute:', t.minute)
print ('Segundo:', t.second)
print ('Microsegundo:', t.microsecond)
```

```
Hora  : 7
Minute: 43
Segundo: 28
Microsegundo: 0
```

```
d1 = datetime.date(2015, 4, 28)
print ('d1:', d1)
```

```
d1: 2015-04-28
```

```
d2 = d1.replace(year=2016)
print ('d2:', d2)
```

```
d2: 2016-04-28
```

```
d2 - d1
```

```
datetime.timedelta(days=366)
```

Map

quarta-feira, 24 de fevereiro de 2021 22:18

É uma função built in
Recebe 2 parâmetros: função, item iterável
Pega essa função e aplica a todos os itens

map(função, sequência)

```
def fahrenheit(T):
    return ((float(9)/5)*T + 32)
```

```
temperaturas = [0, 22.5, 40, 100]
```

```
map(fahrenheit, temperaturas)
```

```
<map at 0x4f2af60>
```

```
list(map(fahrenheit, temperaturas))
```

```
[32.0, 72.5, 104.0, 212.0]
```

```
for temp in map(fahrenheit, temperaturas):
    print(temp)
```

```
32.0
72.5
104.0
212.0
```

```
list(map(lambda x: (5.0/9)*(x - 32), temperaturas))
```

```
[-17.77777777777778, -5.277777777777778, 4.444444444444445, 37.77777777777778]
```

```
# Somando os elementos de 2 listas
a = [1,2,3,4]
b = [5,6,7,8]
```

```
list(map(lambda x,y:x+y, a, b))
```

```
[6, 8, 10, 12]
```

```
def fahrenheit(T):
    return ((float(9)/5)*T + 32)
temperaturas = [0, 22.5, 40, 100]
map(fahrenheit, temperaturas)
list(map(fahrenheit, temperaturas))
```

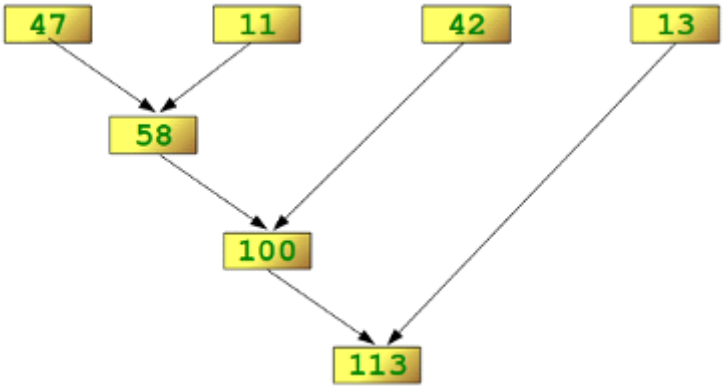
Reduce

quarta-feira, 24 de fevereiro de 2021 22:34

função built in, mas precisa importar:

from functools import reduce

reduce(função, sequência)




```
from functools import reduce
```

```
lista = [47,11,42,13]
```

```
def soma(a,b):
    x = a + b
    return x
```

```
reduce(soma, lista)
```

```
113
```

Filter

quarta-feira, 24 de fevereiro de 2021 22:44

Função built in

filter(função, sequência)

Filtra todos os elementos de uma sequência, para os quais a função retorne True.

```
def verificaPar(num):
    if num % 2 == 0:
        return True
    else:
        return False
```

```
lista = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18]
```

```
filter(verificaPar, lista)
```

```
<filter at 0x43755c0>
```

```
list(filter(verificaPar, lista))
```

```
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

```
list(filter(lambda x: x%2==0, lista))
```

```
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

List comprehention

quarta-feira, 24 de fevereiro de 2021 22:54

```
lst = [x for x in 'python']
lst
```

```
['p', 'y', 't', 'h', 'o', 'n']
```

```
lst = [x**2 for x in range(0, 11)]
lst
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

```
lst = [x for x in range(11) if x % 2 == 0]
lst
```

```
[0, 2, 4, 6, 8, 10]
```

```
celsius = [0,10,20.1,34.5]

fahrenheit = [ ((float(9)/5)*temp + 32) for temp in celsius ]

fahrenheit
```

```
[32.0, 50.0, 68.18, 94.1]
```

```
lst = [ x**2 for x in [x**2 for x in range(11)]]
lst
```

```
[0, 1, 16, 81, 256, 625, 1296, 2401, 4096, 6561, 10000]
```

Zip

quarta-feira, 24 de fevereiro de 2021 23:01

zip( ) pode ser usada quando o número de elementos for diferente em cada sequência.
O objeto resultante terá o mesmonúmero de elementos da sequência menor.

```
x = [1,2,3]
y = [4,5,6]
```

```
zip(x, y)
list(zip(x,y))
```

```
[(1, 4), (2, 5), (3, 6)]
```

```
list(zip('ABCD', 'xy'))
```

```
[('A', 'x'), ('B', 'y')]
```

```
d1 = {'a':1,'b':2}
d2 = {'c':4,'d':5}
```

```
list(zip(d1,d2))
```

```
[('a', 'c'), ('b', 'd')]
```

```
list(zip(d1, d2.values()))
```

```
def trocaValores(d1, d2):
    dicTemp = {}

    for d1key, d2val in zip(d1,d2.values()):
        dicTemp[d1key] = d2val

    return dicTemp
```

```
trocaValores(d1, d2)
```

```
{'a': 4, 'b': 5}
```

Enumerate

quarta-feira, 24 de fevereiro de 2021 23:03

A função enumerate permite retornar o índice de cada valor em uma sequência,à medida que você percorre toda a sequência.
Recebe apenas um parâmetro de entrada: enumerate(sequência)
Enumerate retorna uma tupla no formato: tupla(índice, valor)

```
seq = ['a','b','c']
```

```
enumerate(seq)
```

```
<enumerate at 0x5025360>
```

```
list(enumerate(seq))
```

```
[(0, 'a'), (1, 'b'), (2, 'c')]
```

```
for indice, valor in enumerate(seq):
    print (indice, valor)
```

```
0 a
1 b
2 c
```

```
for indice, valor in enumerate(seq):
    if indice >= 2:
        break
    else:
        print (valor)
```

```
a
b
```

```
lista = ['Marketing', 'Tecnologia', 'Business']
```

```
for i, item in enumerate(lista):
    print(i, item)
```

```
0 Marketing
1 Tecnologia
2 Business
```

```
for i, item in enumerate('Isso é uma string'):
    print(i, item)
```

```
0 I
1 s
2 s
3 o
4
5 é
6
7 u
8 m
9 a
10
11 s
12 t
13 r
14 i
15 n
16 g
```

```
for i, item in enumerate(range(10)):
    print(i, item)
```

```
0 0
1 1
2 2
3 3
4 4
5 5
6 6
7 7
8 8
9 9
```

Erros e exceções

quarta-feira, 24 de fevereiro de 2021 23:20

try, finally, except

```
try:
    8 + 's'
except TypeError:
    print("Operação não permitida")
```

```
Operação não permitida
```

```
try:
    f = open('arquivos/testandoerros.txt','w')
    f.write('Gravando no arquivo')
except IOError:
   print ("Erro: arquivo não encontrado ou não pode ser salvo.")
else:
   print ("Conteúdo gravado com sucesso!")
   f.close()
```

```
Conteúdo gravado com sucesso!
```

```
try:
    f = open('arquivos/testandoerros.txt','w')
    f.write('Gravando no arquivo')
except IOError:
   print ("Erro: arquivo não encontrado ou não pode ser salvo.")
else:
   print ("Conteúdo gravado com sucesso!")
   f.close()
finally:
   print ("Comandos no bloco finally são sempre executados!")
```

```
Conteúdo gravado com sucesso!
Comandos no bloco finally são sempre executados!
```

```
def askint():
        try:
            val = int((input("Digite um número: ")))
        except UnboundLocalError:
            print ("Você não digitou um número!")
        finally:
            print ("Obrigado!")
        print (val)
```

```
askint()
```

```
Digite um número: 5
Obrigado!
5
```

```
def askint():
        try:
            val = int(input("Digite um número: "))
        except:
            print ("Você não digitou um número!")
            val = int(input("Tente novamente. Digite um número: "))
        finally:
            print ("Obrigado!")
        print (val)
```

```
askint()
```

```
Digite um número:
Você não digitou um número!
Tente novamente. Digite um número: 12
Obrigado!
12
```

```
def askint():
    while True:
        try:
            val = int(input("Digite um número: "))
        except:
            print ("Você não digitou um número!")
            continue
        else:
            print ("Obrigado por digitar um número!")
            break
        finally:
            print("Fim da execução!")
        print (val)
```

```
askint()
```

```
Digite um número: 8
Obrigado por digitar um número!
Fim da execução!
```

exceções:
https://docs.python.org/3.7/library/exceptions.html

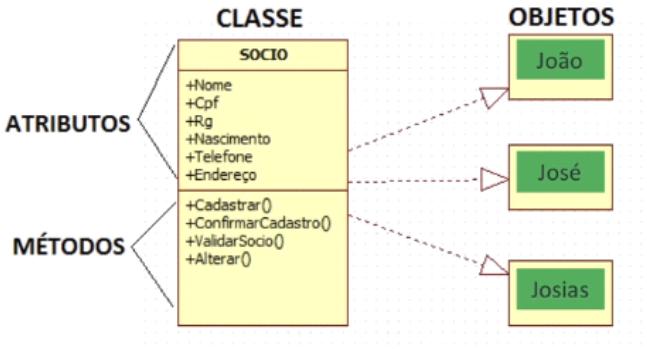
Capítulo 5

quarta-feira, 24 de fevereiro de 2021 23:40

Programação orientada a objetos

Principais conceitos da POO

- Classe
- Objeto
- Atributos
- Métodos
- Mensagem
- Herança
- Polimorfismo
- Encapsulamento



Por convenção, o nome de uma classe começa com maiúscula.

Classe, objeto, método

Exemplo 1

```
class Livro():

    def __init__(self):
        self.titulo = 'O Monge e o Executivo'
        self.isbn = 9988888
        print("Construtor chamado para criar um objeto desta classe")
    def imprime(self):
        print("Foi criado o livro %s e ISBN %d" %(self.titulo, self.isbn))
```

```
Livro1 = Livro()
```

```
Construtor chamado para criar um objeto desta classe
```

```
Livro1.titulo
```

```
'O Monge e o Executivo'
```

```
Livro1.imprime()
```

```
Foi criado o livro O Monge e o Executivo e ISBN 9988888
```

Exemplo 2

```
class Livro():
    def __init__(self, titulo, isbn):
        self.titulo = titulo
        self.isbn = isbn
        print("Construtor chamado para criar um objeto desta classe")

    def imprime(self, titulo, isbn):
        print("Este é o livro %s e ISBN %d" %(titulo, isbn))
```

```
Livro2 = Livro("A Menina que Roubava Livros", 77886611)
```

```
Construtor chamado para criar um objeto desta classe
```

```
Livro2.titulo
```

```
'A Menina que Roubava Livros'
```

```
class Livro():
    def __init__(self, titulo, isbn):
        self.titulo = titulo
        self.isbn = isbn
        print("Construtor chamado para criar um objeto desta classe")
    def imprime(self, titulo, isbn):
        print("Este é o livro %s e ISBN %d" %(titulo, isbn))
```

Exemplo 3

```
lst_num = ["Data", "Science", "Academy", "Nota", 10, 10]
```

```
lst_num.count(10)
```

```
2
```

Exemplo 4

```
class Estudantes:
    def __init__(self, nome, idade, nota):
        self.nome = nome
        self.idade = idade
        self.nota = nota
```

```
Estudante1 = Estudantes("Angelo", 36, 10)
```

Exemplo 5

```
class Funcionarios:
    def __init__(self, nome, salario):
        self.nome = nome
        self.salario = salario

    def listFunc(self):
        print("O nome do funcionário é " + self.nome + " e o salário é R$" + str(self.salario))
```

```
Func1 = Funcionarios("Rambo", 20000)
```

```
Func1.listFunc()
```

```
O nome do funcionário é Rambo e o salário é R$20000
```

hasattr - Verifica se aquele objeto possui aquele atributo

Exemplo 6 - Verifica se o objeto Func1 possui um atributo nome

```
hasattr(Func1, "nome")
```

```
True
```

getattr - Me retorna o valor do atributo

Exemplo 7 - Me retorna o atributo salario do Func1

```
getattr(Func1, "salario")
```

```
20000
```

delattr - Deleta o atributo

Exemplo 8 - Deleta o atributo salario do objeto Func1

```
delattr(Func1, "salario")
```

```
hasattr(Func1, "salario")
```

```
False
```

Exemplo 9

```
# Criando uma classe chamada Circulo
class Circulo():

    # O valor de pi é constante
    pi = 3.14

    # Este método será executado quando o obj for criado, e o valor default do raio = 5.
    def __init__(self, raio = 5):
        self.raio = raio

    # Esse método calcula a área. Self utiliza os atributos deste mesmo objeto
    def area(self):
        return (self.raio * self.raio) * Circulo.pi

    # Método para gerar um novo raio
    def setRaio(self, novo_raio):
        self.raio = novo_raio

    # Método para obter o raio do círculo
    def getRaio(self):
        return self.raio
```

```
# Criando o objeto circ. Uma instância da classe Circulo()
circ = Circulo()
```

```
# Executando um método da classe Circulo
circ.getRaio()
```

```
5
```

```
# Criando outro objeto chamado circ1. Uma instância da classe Circulo()
# Agora sobrescrevendo o valor do atributo
circ1 = Circulo(7)
```

```
# Executando um método da classe Circulo
circ1.getRaio()
```

```
7
```

```
# Gerando um novo valor para o raio do círculo
circ.setRaio(3)
```

```
# Imprimindo o novo raio
print ('Novo raio igual a: ', circ.getRaio())
```

```
Novo raio igual a:  3
```

Herança e Met. Esp.

quinta-feira, 25 de fevereiro de 2021 19:04

Herança

Exemplo 1

```
# Criando a classe Animal - Super-classe
class Animal():

    def __init__(self):
        print("Animal criado")

    def Identif(self):
        print("Animal")

    def comer(self):
        print("Comendo")
```

```
# Criando a classe Cachorro - Sub-classe
class Cachorro(Animal):

    def __init__(self):
        Animal.__init__(self)
        print("Objeto Cachorro criado")

    def Identif(self):
        print("Cachorro")

    def latir(self):
        print("Au Au!")
```

```
# Criando um objeto (Instanciando a classe)
rex = Cachorro()
```

```
Animal criado
Objeto Cachorro criado
```

```
# Executando o método da classe Cachorro (sub-classe)
rex.Identif()
```

```
Cachorro
```

```
# Executando o método da classe Animal (super-classe)
rex.comer()
```

```
Comendo
```

```
# Executando o método da classe Cachorro (sub-classe)
rex.latir()
```

```
Au Au!
```

Métodos especiais

__abs__	__delattr__	__getattribute__	__index__
__add__	__delete__	__getitem__	__init__
__and__	__delitem__	__getslice__	__instancecheck_
__call__	__delslice__	__gt__	__int__
__class__	__dict__	__hash__	__invert__
__cmp__	__div__	__hex__	__ior__
__coerce__	__divmod__	__iadd__	__ipow__
__complex__	__eq__	__iand__	__irshift__
__contains__	__float__	__idiv__	__isub__
__del__	__floordiv__	__ifloordiv__	__long__
__itruediv__	__ge__	__ilshift__	__lshift__
__ixor__	__get__	__imod__	__mod__
__len__	__getattr__	__imul__	__new__

Exemplo 1

```
# Criando a classe Livro
class Livro():
    def __init__(self, titulo, autor, paginas):
        print ("Livro criado")
        self.titulo = titulo
        self.autor = autor
        self.paginas = paginas

    def __str__(self):
        return "Título: %s , autor: %s, páginas: %s " \
    %(self.titulo, self.autor, self.paginas)

    def __len__(self):
        return self.paginas

    def len(self):
        return print("Páginas do livro com método comum: ", self.paginas)
```

```
livro1 = Livro("Os Lusíadas", "Luis de Camões", 8816)
```

```
Livro criado
```

Quando eu pedir pra dar um print em em um objeto, vai rodar o __str__
Posso escrever print(Objeto) ou str(Objeto), as duas formas vão rodar o __str__

```
livro1 = Livro("Os Lusíadas", "Luis de Camões", 8816)
```

```
Livro criado
```

```
# Métodos especiais
print(livro1)
```

```
Título: Os Lusíadas , autor: Luis de Camões, páginas: 8816
```

```
str(livro1)
```

```
'Título: Os Lusíadas , autor: Luis de Camões, páginas: 8816 '
```

Quando eu escrevo len(Objeto), estou chamando o método especial __len__
Para chamar o método que criei chamado len eu uso objeto.len( )

```
len(livro1)
```

```
8816
```

```
livro1.len()
```

```
   Páginas do livro com método comum:  8816
```

Capítulo 6

sexta-feira, 26 de fevereiro de 2021 19:32

Banco de dados

Bancos de Dados Relacionais

- São gerenciados por RDBMS (Relational Database Management System)
- Um conceito importante em um banco de dados realcional é o conceito de atributo chave , que permite identificar e diferenciar uma tupla de outra.