Python: Conhecendo a linguagem e suas possibilidades

Python é uma das linguagens de programação mais usadas atualmente e merece uma atenção especiais por algumas particularidades que aqui serão listadas e que poderão ser observadas ao longo desta aula. Segue a lista:

- Python é fácil de aprender
- Fácil de implementar
- É uma linguagem interpretada
- Possui vasta comunidade
- Muitas bibliotecas
- Multiplataforma

A partir deste momento focaremos nos princípios básicos da lingaguem e posteriormente apresentarei exemplos de aplicação.

Variáveis

Python trás a facilidade de não precisarmos definir os tipos das variáveis, pois o interpretador faz isso por nós. Definir variáveis em Python é bem simples, basta escolhermos um nome e atribuir um valor a ela.

In [189]:
```
a = 1
b = 1.0
c = "c é 1.0"
d = [1,2, "feijao", True, 1.2, {"um":"dicionario"}]
e = (1,2)
f = True
g = {"emojis são divertidos" : "😁"}

print(type(a))
print('------')
print(type(b))
print('------')
print(type(c))
print('------')
print(type(d))
print('------')
print(type(e))
print('------')
print(type(f))
print('------')
print(type(g))
```

```
<class 'int'>
------
<class 'float'>
------
<class 'str'>
------
<class 'list'>
------
<class 'tuple'>
------
<class 'bool'>
------
<class 'dict'>
```

Listas, Tuplas, Sets e Strings

Esses são três tipos do python que são agrupamentos de valores. A lista pode armazenar qualquer tipo de valor, a tupla não pode ser posteriormente modificada e os sets não podem possuir números repetidos. A seguir mostrarei em código como tudo funciona.

In [190]:
```
lista = [1,2,3,4.0,"feijao",{"oi":"io"}]
tupla = (1,2,3,"oi")

print("acessando conteúdo da lista: ", lista[0], lista[-1])
```

```
acessando conteúdo da lista:  1 {'oi': 'io'}
```

In [191]:
```
#modificando conteúdo da lista
lista[1] = "fui modificado"

print("nova lista: ", lista)
```

```
nova lista:  [1, 'fui modificado', 3, 4.0, 'feijao', {'oi': 'io'}]
```

In [192]:
```
#adicionando conteúdo
lista.append("Fui adicionado")

print("nova lista: ", lista)
```

```
nova lista:  [1, 'fui modificado', 3, 4.0, 'feijao', {'oi': 'io'}, 'Fui adici
onado']
```

In [193]:
```
#removendo conteúdo de um determinado ponto
lista.pop(0)

print("nova lista: ", lista)
```

```
nova lista:  ['fui modificado', 3, 4.0, 'feijao', {'oi': 'io'}, 'Fui adiciona
do']
```

In [194]:
```
outra_lista = ['eu','sou','outra','lista']
#concatenando listas
lista = outra_lista + lista
print(lista)
```

```
['eu', 'sou', 'outra', 'lista', 'fui modificado', 3, 4.0, 'feijao', {'oi': 'i
o'}, 'Fui adicionado']
```

Para mais possibilidades usando listas consultar: https://www.programiz.com/python-programming/methods/tuple (https://www.programiz.com/python-programming/methods/tuple)

In [195]:
```
# tenta modificar a tupla e mostra o erro
try:
    tupla[1] = "fui modificado"
except Exception as e:
    print('a tupla: ', tupla)
    print('o erro: ', e)
```

```
a tupla:  (1, 2, 3, 'oi')
o erro:  'tuple' object does not support item assignment
```

In [196]:
```
#tentando adicionar objetos à tupla
try:
    tupla.append('0')
except Exception as e:
    print(e)
```

```
'tuple' object has no attribute 'append'
```

In [197]:
```
#concatenando tuplas
tupla_a = (0,1,2)
tupla_b = (3,4,5)

resultado = tupla_a + tupla_b
print(type(resultado))
print(resultado)
```

```
<class 'tuple'>
(0, 1, 2, 3, 4, 5)
```

Para mais possibilidades usando tuplas consultar: https://www.programiz.com/python-programming/methods/tuple (https://www.programiz.com/python-programming/methods/tuple)

In [198]:
```
lista_nao_set = [2, 4, 1, 3, 5, 3, 4, 2, 5]
set_da_lista = set(lista_nao_set)
print(set_da_lista)
set_da_lista.add(26)
print(set_da_lista)
```

```
{1, 2, 3, 4, 5}
{1, 2, 3, 4, 5, 26}
```

Para mais possibilidades usando sets consultar: https://www.programiz.com/python-programming/methods/set (https://www.programiz.com/python-programming/methods/set)

In [199]:
```
# uma lista diferente: string
string = "eu sou uma string"
outra_string = ", ou será que não?"
#concatenando
string = string + outra_string
print(string)
print(string[0:-5])
print(string.upper())
print(string.split(','))
```

```
eu sou uma string, ou será que não?
eu sou uma string, ou será que
EU SOU UMA STRING, OU SERÁ QUE NÃO?
['eu sou uma string', ' ou será que não?']
```

Para mais possibilidades usando strings consultar: https://www.programiz.com/python-programming/methods/strings (https://www.programiz.com/python-programming/methods/strings)

Loops in Python

Loops são essenciais para interagir com uma quantidade massiva de coisas, realizando tarefas um determinado (ou não) número de vezes. Em python temos dois tipos de loop que veremos a seguir.

In [200]:
```
lista = [1,2,3,4,'5','6']

for element in lista:
    print(element)
```

```
1
2
3
4
5
6
```

In [201]:
```
string = 'eu sou uma string graaaaande'

for element in string:
    print(element + '                    fui concatenado em cada iteração')
```

```
e                    fui concatenado em cada iteração
u                    fui concatenado em cada iteração
                     fui concatenado em cada iteração
s                    fui concatenado em cada iteração
o                    fui concatenado em cada iteração
u                    fui concatenado em cada iteração
                     fui concatenado em cada iteração
u                    fui concatenado em cada iteração
m                    fui concatenado em cada iteração
a                    fui concatenado em cada iteração
                     fui concatenado em cada iteração
s                    fui concatenado em cada iteração
t                    fui concatenado em cada iteração
r                    fui concatenado em cada iteração
i                    fui concatenado em cada iteração
n                    fui concatenado em cada iteração
g                    fui concatenado em cada iteração
                     fui concatenado em cada iteração
g                    fui concatenado em cada iteração
r                    fui concatenado em cada iteração
a                    fui concatenado em cada iteração
a                    fui concatenado em cada iteração
a                    fui concatenado em cada iteração
a                    fui concatenado em cada iteração
a                    fui concatenado em cada iteração
n                    fui concatenado em cada iteração
d                    fui concatenado em cada iteração
e                    fui concatenado em cada iteração
```

In [202]:
```
dicionario = {
    "eu" : "eu",
    "sou": "sou",
    "um" : "o",
    "dicionario" : "valor"
}

for key, _ in dicionario.items():
    print (key)

print('----------------')


for _, value in dicionario.items():
    print (value)
```

```
eu
sou
um
dicionario
----------------
eu
sou
o
valor
```

In [203]:
```
a = 0
while a < 20:
    print(a)
    a += 1

# obs: para loop infinito é só falar que while True: (só faça isso se tiver c
erteza do que ta fazendo)
```

```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
```

Bonus: Gerando Listas de Listas

Em python é possível gerar listas a partir de listas como veremos a seguir:

In [204]:
```
lista_gerada_por_lista = [x**2 for x in [1,2,3,4,5,6,7,8,9]]
                          #[logica para variavel em lista]
print(lista_gerada_por_lista)
```

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

Condicionais

Em python existem 3 tipos de condicionais

- if: Padrão. Se satisfizer a condição executa o código
- elif: É o famoso se não. Se não executar o if vê se eu dou pro gasto
- else: Ninguém quer executar o código? Eu quero

Veremos exemplo a seguir

In [205]:
```
def condicionais(a):
    if a > 2:
        print('condicao a executada')
    elif 1 < a <= 2:
        print('condicao b executada')
    else:
        print('condicao c executada')
    return ""

print(
    condicionais(2),
    condicionais(3),
    condicionais(1)
)
```

```
condicao b executada
condicao a executada
condicao c executada
```

Dicionários: Outra maneira de lidar com dados

Os dicionários, assim como as listas, armazenam dados. Um ponto relevante sobre os dicionários é que eles se assemelham ao json, formato padrão de transmissão de dados na internet. Essa semelhança faz com que o python consiga fazer uma conversão direta entre eles. Segue exemplos de interações com dicionários

In [206]:
```
#criando dicionários. Os dicionários são criados por declarar charves
dicionario = {
    'um':'dicionario',
    'com':'itens',
    'variados':'diversos'
}
#acessando o dicionario
print(
    dicionario["um"],
    dicionario["com"],
    dicionario["variados"]
)
```

```
dicionario itens diversos
```

In [207]:
```
#adicionando itens ao dicionario
dicionario["novo"] = "item"
print(dicionario)
```

```
{'um': 'dicionario', 'com': 'itens', 'variados': 'diversos', 'novo': 'item'}
```

In [208]:
```
#removendo item
dicionario.pop("novo")
print(dicionario)
```

```
{'um': 'dicionario', 'com': 'itens', 'variados': 'diversos'}
```

In [209]:
```
#retornando uma lista com chaves e valores do dicionário em forma de tuplas
dicionario.items()
```

Out[209]:
```
dict_items([('um', 'dicionario'), ('com', 'itens'), ('variados', 'diversos')])
```

In [210]:
```
#retornando valores
dicionario.values()
```

Out[210]:
```
dict_values(['dicionario', 'itens', 'diversos'])
```

Para mais possibilidades usando dicionários consultar: https://www.programiz.com/python-programming/methods/dictionary (https://www.programiz.com/python-programming/methods/dictionary)

Funções em Python

Funções em Python têm o dever de definir um código reusável e um novo fluxo para os dados, além de prover uma forma de executar o mesmo código para parâmetros diferentes. A seguir teremos algumas possibilidades com funções:

In [211]:
```
def exemplo(a, b, *args, **kwargs):
    print(a)
    print(b)
    if args:
        print(args)
    if kwargs:
        print(kwargs)
    return "eu sou o produto da função"

retorno = exemplo(a, b, *[1, 2, 3, 4], **{"c":"d"})
print(retorno)
```

```
20
1.0
(1, 2, 3, 4)
{'c': 'd'}
eu sou o produto da função
```

In [212]:
```
def exemplob(**kwargs):
    return kwargs

func_dict = exemplob(carro="preto", moto="branca", caminhão="vermelho")
print(func_dict)
```

```
{'carro': 'preto', 'moto': 'branca', 'caminhão': 'vermelho'}
```

Objetos em Python

A programação orientada a objetos é o paradigma de programação mais popular atualmente, pois com ele o código fica altamente reusável, customisável, adaptável e é extremamente fácil de manipular e documentar. Aqui daremos uma leve olhada sobre o mundo dos objetos em Python para não passar em branco.

Um objeto é composto de características (propriedades) e funcionalidades atreladas a ele (métodos). O objeto é feito pro uma forma predefinida e pode herdar características de outros objetos. Vamos aos exemplos.

In [213]:
```
class Carro:
    possui_freio = True
    numero_rodas = 4
    possui_volante = True

    def __init__(self, cor, marca, ano, proprietario="loja"):
        self.cor = cor
        self.marca = marca
        self.ano = ano
        self.proprietario = proprietario
        self.velocidade = 0

    def acelerar(self):
        print('o veiculo acelerou')
        self.velocidade += 10

    def freiar(self):
        print('o veiculo desacelerou')
        self.velocidade -= 10

    def parar(self):
        print('o veiculo parou')
        self.velocidade = 0

vectra = Carro("vermelho", "chevrolet", "1999", proprietario="jv")
print('sobre o vectra: ---')
print(vectra.possui_freio, vectra.numero_rodas, vectra.possui_volante)
print(vectra.cor, vectra.marca, vectra.ano, vectra.proprietario)
print('sobre qualquer carro: ---')
print(Carro.possui_freio, Carro.numero_rodas, Carro.possui_volante)
```

```
sobre o vectra: ---
True 4 True
vermelho chevrolet 1999 jv
sobre qualquer carro: ---
True 4 True
```

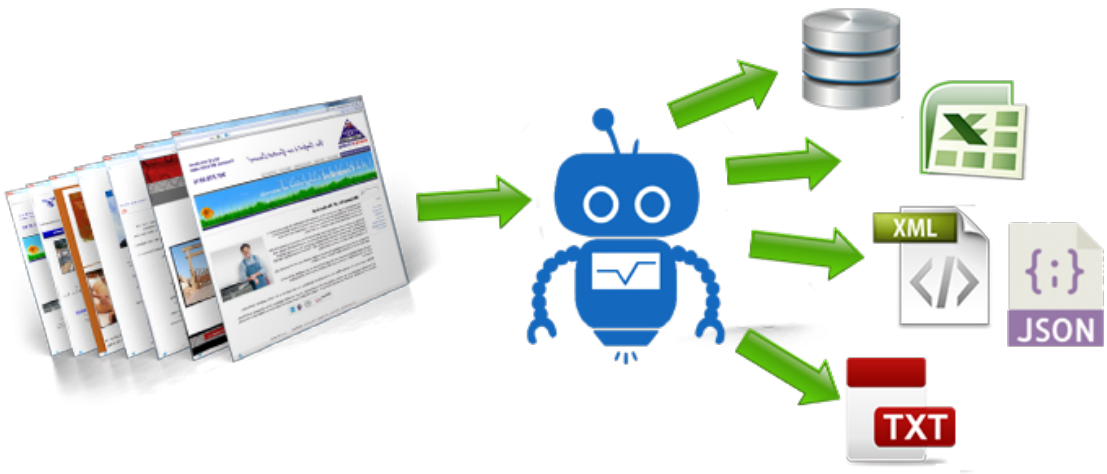
In [214]:
```
#acelerando o vectra
print(vectra.velocidade)
vectra.acelerar()
vectra.acelerar()
vectra.acelerar()
vectra.acelerar()
vectra.acelerar()
print(vectra.velocidade)
vectra.freiar()
vectra.freiar()
vectra.freiar()
print(vectra.velocidade)
vectra.parar()
print(vectra.velocidade)
```

```
0
o veiculo acelerou
o veiculo acelerou
o veiculo acelerou
o veiculo acelerou
o veiculo acelerou
50
o veiculo desacelerou
o veiculo desacelerou
o veiculo desacelerou
20
o veiculo parou
0
```

In [215]:
```
class Quadriciculo(Carro):
    def __init__(self):
        self.velocidade = 0

meu_quadriciculo = Quadriciculo()

try:
    print(meu_quadriciculo.cor)
except Exception as e:
    print(e)

print(meu_quadriciculo.velocidade)
meu_quadriciculo.acelerar()
meu_quadriciculo.acelerar()
meu_quadriciculo.acelerar()
meu_quadriciculo.acelerar()
print(meu_quadriciculo.velocidade)
meu_quadriciculo.freiar()
meu_quadriciculo.freiar()
print(meu_quadriciculo.velocidade)
meu_quadriciculo.parar()
print(meu_quadriciculo.velocidade)
```

```
'Quadriciculo' object has no attribute 'cor'
0
o veiculo acelerou
o veiculo acelerou
o veiculo acelerou
o veiculo acelerou
40
o veiculo desacelerou
o veiculo desacelerou
20
o veiculo parou
0
```

Web Scraping com Python

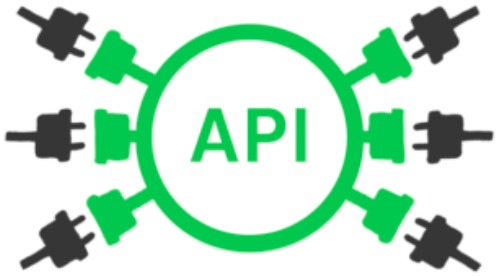


Web scraping consiste em utilizar bibliotecas combinadas de Python para obter dados de páginas web que são visíveis para o usuário. Um exemplo de aplicação é utilizar python para pegar automaticamente todas as notícias do dia na página principal de um portal e é o que faremos aqui. Para isso utilizaremos 2 bibliotecas do python: *requests* e *beautifulsoup*

In [224]:
```
import requests
from bs4 import BeautifulSoup
from IPython.display import display

page = requests.get('https://www.uol.com.br/')
soup = BeautifulSoup(page.text, 'html.parser')
lista = soup.find_all(class_="article_title")
lista = [x.text for x in lista]
display(lista)
```

```
[' Palmeiras goleia América-MG por 4 a 0 e fica mais perto de título; veja go
ls ',
 ' Flamengo bate Grêmio por 2 a 0 e ainda sonha com título ',
 ' Atlético-PR vence Corinthians e mantém chances de atingir G-6 ',
 ' Cuca realizará cirurgia no coração e deve deixar o Santos ',
 ' Indicado à Educação causa crise com bancada evangélica ',
 ' TCU aponta riscos de fraude e corrupção em 38 órgãos federais ',
 ' Ninguém acerta Mega-Sena, e prêmio deve chegar a R$ 70 mi ',
 ' Delegado conclui inquérito sobre morte de Daniel e indicia 7 ',
 ' Atlético-MG bate Inter por 2 a 1, e resultado ajuda o Palmeiras ',
 ' Justiça apreende carros e obra de arte em casa de irmão de Ronaldinho ',
 ' Vote e eleja os seus preferidos ao Troféu UOL TV e Famosos ',
 ' Sandra Bullock e elenco de Stranger Things estarão na CCXP de SP ',
 ' A gente se sentia explorado, diz cubano que saiu do Mais Médicos ',
 ' Iniciativa inédita: a prisão alemã onde cada detento tem o próprio tablet
',
 ' Pouco visitadas, cavernas do Peruaçu guardam patrimônio arqueológico ',
 ' Com 2 de Fred, Cruzeiro bate o Vitória por 3 a 0 no Mineirão; assista aos
gols ',
 ' Federação divulga tabela da 1ª fase do Paulistão ',
 ' Malcom busca o protagonismo no Barça ',
 ' Em um ano atuou apenas 14 vezes ',
 ' Torcedores protestam em treino e quebram carro de diretor do Vasco ',
 ' Chinês desbanca R. Augusto e Paulinho e leva prêmio na China ',
 ' No Placar UOL, você tem classificação em tempo real e 10 times favoritos '
,
 ' Plano para Estaduais para 2020 prevê fase final em dezembro ',
 ' Juventus quer decidir se Alex Sandro fica ou deixa o clube ',
 " Fora do UFC, Werdum fará luta de 'submission com tapas' no México ",
 ' Richarlison vira representante do torcedor na seleção ',
 ' Marido é preso por tentativa de homicídio contra lutadora do UFC ',
 ' Thaisa assume namoro com Mineiro, pivô do São José no NBB ',
 ' Por que assédio não é mimimi - nem no futebol, nem fora dele ',
 ' Leandro, do Choque: Imagina viver nessa realidade paralela? ',
 ' Christian Bale brilha como Dick Cheney em Vice; veja trailer ',
 ' Ilha diz se arrepender de envolvimento com drogas ',
 ' Delegado Machado surta com fetiche da mulher ',
 ' Exposição conta a vida do astro ',
 ' Barney: Sair da Record é mau negócio para Mion, Porchat e Bacci ',
 ' Pusha-T acusa Drake de pagar "capangas" para realizar ataque ',
 ' Bate-papo UOL mostra quem está perto e dá acesso rápido às salas ',
 ' Cantora Anna Setton lança álbum de estreia após 14 anos de estrada ',
 ' Bruna Marquezine usa casacão e tênis em dia frio em praia de Portugal ',
 ' O Tempo Não Para tem pior Ibope em dois anos no horário das 19h ',
 ' Otaviano fala sobre projetos: "Não volto mais para as novelas" ',
 ' Em ascensão, ator Gabriel Leone flerta com a música ',
 ' Até 70% de desconto em flores e presentes ',
 ' SP tem trânsito acima da média, mas sem recorde ',
 ' PT avalia entrar com ação nos EUA contra WhatsApp ',
 ' TRF-4 nega pedido de Lula para novo interrogatório ',
 ' Carrega cicatrizes de uma guerra ',
 ' Bombeiros localizam corpo de turista em trilha de Arraial do Cabo (RJ) ',
 ' Bolsonaro decide usar Granja do Torto durante transição de governos ',
 ' Viaje pelo mundo com vídeos 360° do app de realidade virtual do UOL ',
 ' Justiça de SP ordena ida de chefes do PCC para presídios federais ',
 ' MP pede prisão de acusados de matar moradora que criticou tráfico ',
 ' Justiça condena SP por enterrar desaparecidos como indigentes ',
 ' Relatos de LGBTs que adiaram voltar ao Brasil com medo de ataques ',
 ' Curte jogo de corrida? Fique longe desses 13 games para Android ',
 ' Novo Suzuki Jimny tunado ganha cara de Classe G ou Defender ',
 ' Carne na grelha e drink na mão: Jennifer Lopez posa de lingerie ',
 ' Anne Hathaway abandona os fios castanhos e surge ruiva ',
 ' Príncipe William diz que ficou mais "sensível" com a paternidade ',
 ' Maisa avisa: "Boy com masculinidade frágil não é para mim" ',
 ' Sapatos lace up: modelos de amarrar são tendência ',
 ' Veja 10 momentos de fofura de Henry, filho de Simone ',
 ' Lontras invadem pedido de casamento em Cingapura ',
 ' Dieta com pouco glúten pode beneficiar quem não tem alergia ',
```

APIs com Python



APIs (Application Programming Interface) é uma interface que soluções e serviços oferecem para interação com seu banco de dados, ou com alguma funcionalidade que oferece. APIs são disponibilizadas por uma infinidade de aplicações. Dentre elas estão: Bancos, redes sociais, portais da transparência, mercado financeiro e etc.

Aqui utilizaremos a API de um serviço que oferece dados de criptomoedas: https://coinmarketcap.com/ (https://coinmarketcap.com/). Segue o exemplo

In [217]:
```
import requests
import pandas as pd
import json
from IPython.display import display

data_text = requests.get('https://api.coinmarketcap.com/v2/ticker/?convert=BRL').text
data_json = json.loads(data_text)

'''o que eu quero: data_json['data'][x][name, symbol, rank]['quotes']['BRL']['price']'''

resp_list = []
for k,v in data_json['data'].items():
    resp = {}
    resp["nome"] = v["name"]
    resp["símbolo"] = v["symbol"]
    resp["rank"] = v["rank"]
    resp["preço R$"] = v["quotes"]["BRL"]["price"]
    resp_list.append(resp)

df = pd.DataFrame(resp_list)
display(df)
```

	nome	preço R$	rank	símbolo
0	Bitcoin	17322.564581	1	BTC
1	XRP	1.682465	2	XRP
2	Ethereum	512.849903	3	ETH
3	Bitcoin Cash	881.233551	4	BCH
4	Stellar	0.759552	5	XLM
5	EOS	14.404594	6	EOS
6	Litecoin	131.037868	7	LTC
7	Tether	3.750665	8	USDT
8	Cardano	0.178804	9	ADA
9	Monero	263.837707	10	XMR
10	TRON	0.055645	11	TRX
11	Dash	424.025000	12	DASH
12	IOTA	1.241027	13	MIOTA
13	Binance Coin	23.404836	14	BNB
14	NEM	0.320426	15	XEM
15	Ethereum Classic	21.752934	16	ETC
16	NEO	34.593559	17	NEO
17	Zcash	322.386738	18	ZEC
18	Tezos	2.779791	19	XTZ
19	Bitcoin Gold	78.872205	20	BTG
20	Maker	1711.445966	21	MKR
21	VeChain	0.021703	22	VET
22	Ontology	3.849988	23	ONT
23	Dogecoin	0.009156	24	DOGE
24	OmiseGO	7.607499	25	OMG
25	0x	1.678671	26	ZRX
26	Decred	99.130015	27	DCR
27	Qtum	9.305460	28	QTUM
28	Basic Attention Token	0.685000	29	BAT
29	TrueUSD	3.848407	30	TUSD
...	...	...	...	...
70	Revain	0.593737	71	R
71	MOAC	3.563646	72	MOAC
72	MonaCoin	3.466109	73	MONA
73	Nexo	0.393080	74	NEXO
74	Mithril	0.554856	75	MITH
75	ODEM	0.940599	76	ODE
76	Insight Chain	1.166631	77	INB
77	Wanchain	1.863020	78	WAN
78	GXChain	3.265000	79	GXS
79	Ark	1.823984	80	ARK
80	PIVX	3.220744	81	PIVX