

Criando Classe em Python

Mário Leite

...

A base de qualquer processamento computacional são, sem sombras de dúvidas, os **Dados**. Sem dados as informações não podem ser obtidas e, por conseguinte, não é possível tomar uma decisão! Portanto, saber gerenciar os dados (*entradas*) é fundamental em qualquer tipo de paradigma de programação. A **figura 1** mostra os principais tipos de dados que existem como entrada no processamento com os quais outros podem ser derivados (dependendo da linguagem de programação); por exemplo, a linguagem C estende essa relação, oferecendo mais recursos ao programador. Mas, de um modo geral esta figura mostra a base de todos os tipos de dados. Particularmente, um tipo de dado muito importante é representado pelos “Registros”, cujos congêneres são: **Type** (Basic), **record** (Pascal), **struct** (C, Swift), **Structure** (VB.Net), **data class** (Kotlin) e **class** (Java, Python, Ruby).

Um tipo de arranjo muito empregado para processar dados de maneira ordenada são as **tabelas**, que representam dados dispostos em linhas e colunas, como nos bancos de dados e que são a essência da manipulação e gerência de dados, na prática. Deste modo, em princípio, as matrizes poderiam ser utilizadas para conter os dados a serem processados. A **tabela 1** mostra um exemplo clássico com os dados cadastrais dos empregados de uma empresa. Mas, a questão é: *como armazenar todos esses dados numa só variável?! Utilizando matrizes (estruturas homogêneas)* só seria possível criando vetores para cada tipo de dado do empregado, pois os *arrays* não admitem tipos de dados diferentes. Assim o conceito de **registro** é perfeitamente aplicável nesta situação, para armazenar todos estes dados em apenas uma única variável, pois registro os registros são estruturas heterogêneas, que define um novo tipo de dado composto por uma coleção de elementos de um mesmo tipo ou de tipos diferentes. Esses elementos são chamados de *membros* ou *campos* e podem ser até mesmo *arrays*. Uma definição mais formal de registro: “*um conjunto de dados logicamente relacionados, de um mesmo tipo ou de tipos diferentes de dados*”. A definição de *registro* expande o conceito de matriz bidimensional, podendo tratar as colunas com dados de tipos diferentes. O Python cria registros usando o conceito de **classe** (*class*). O programa “**RegistroEmpregados**” mostra um código para criar um cadastro de empregados e a **figura 2** mostra um exemplo de saída desse programa. Onde **Empregado** representa uma variável do tipo **class**.

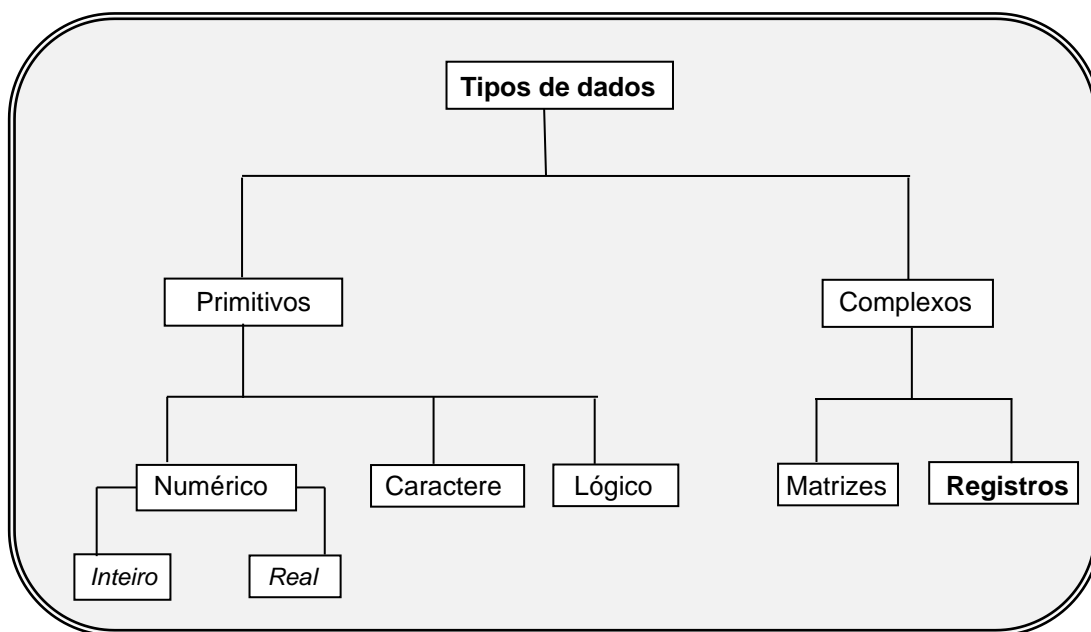
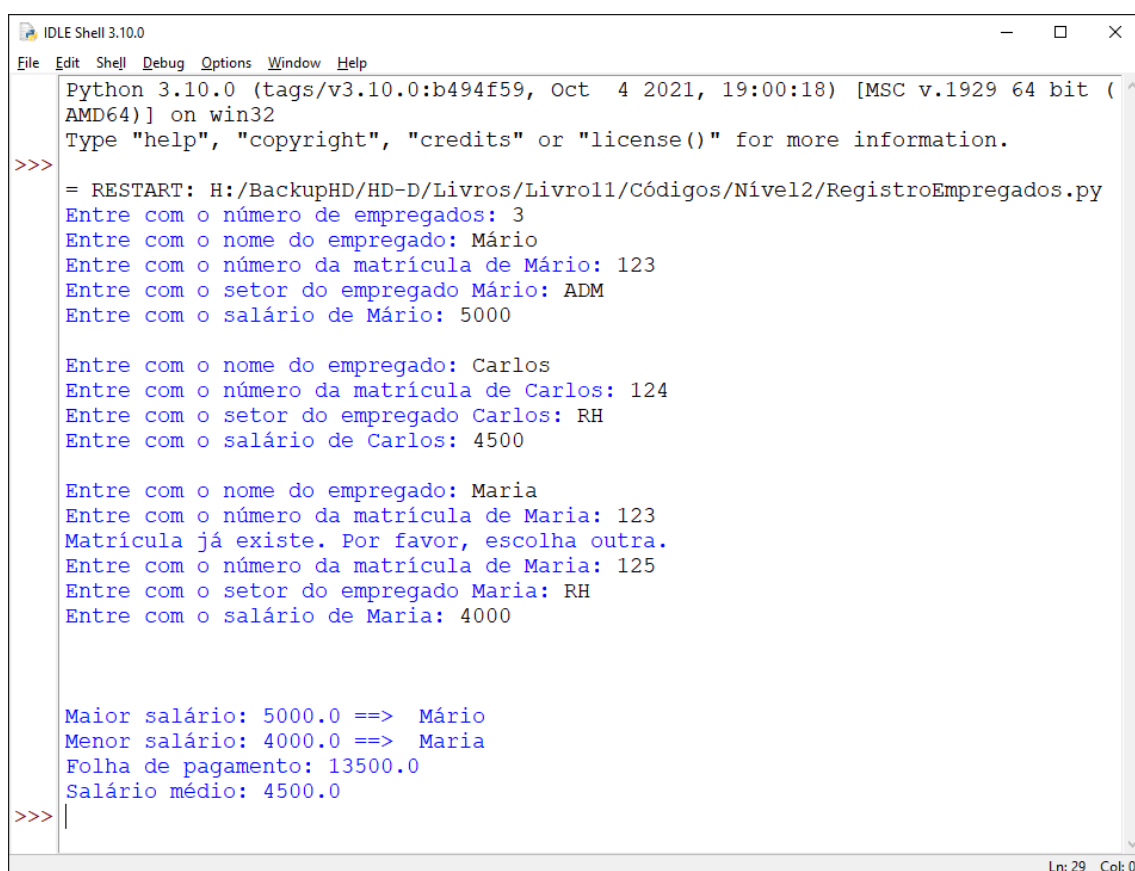


Figura 1 - Tipos básicos de dados

Matrícula	Nome	Sexo	Cargo	Admissão	Salário
1228	Gerônimo C. Boratto	M	Gerente Financeiro	01/07/2012	8.450,00
1229	Mirtes Acioli de Souza	F	Gerente de Pessoal	04/06/2010	6.800,00
1231	Mércio Lúcio da Silva	M	Chefe de Setor	03/08/2009	5.700,00
1233	Ângela Maria Saitama	F	Analista de Custos	01/09/2001	3.820,00
1234	Jandir F. Pereira	M	Programador Sênior	07/02/2008	2.750,00
1235	Jurandir P. Bastos	M	Eletricista I	06/08/2000	2.350,00
1237	Cassiano S. Burgos	M	Eletricista II	09/03/2002	2.950,00
1238	Laíz Santos Carrera	F	Auxiliar de Produção	02/01/2003	1.700,00
1241	Vanderley M. Araújo	M	Analista de Sistemas	06/06/2001	4.230,00
1243	Ana Clara F. R. Leite	F	Programador Júnior	04/05/2008	1.950,00
1244	Wandeilson R. Leite	M	Office boy	01/11/2011	1.000,00
1245	Elizabete S. Vangnoni	F	Secretária	05/10/2010	2.800,00
1247	Gentil Bastos Saito	M	Programador Júnior	04/05/2013	1.950,00
1252	---	-	---	---	---

Tabela 1 - Tabela de dados de empregados



```

Python 3.10.0 (tags/v3.10.0:b494f59, Oct 4 2021, 19:00:18) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: H:/BackupHD/HD-D/Livros/Livro11/Códigos/Nível2/RegistroEmpregados.py
Entre com o número de empregados: 3
Entre com o nome do empregado: Mário
Entre com o número da matrícula de Mário: 123
Entre com o setor do empregado Mário: ADM
Entre com o salário de Mário: 5000

Entre com o nome do empregado: Carlos
Entre com o número da matrícula de Carlos: 124
Entre com o setor do empregado Carlos: RH
Entre com o salário de Carlos: 4500

Entre com o nome do empregado: Maria
Entre com o número da matrícula de Maria: 123
Matrícula já existe. Por favor, escolha outra.
Entre com o número da matrícula de Maria: 125
Entre com o setor do empregado Maria: RH
Entre com o salário de Maria: 4000

Maior salário: 5000.0 ==> Mário
Menor salário: 4000.0 ==> Maria
Folha de pagamento: 13500.0
Salário médio: 4500.0
>>>

```

Figura 2 - Exemplo de saída do programa “RegistroEmpregados”

```
'''
```

RegistroEmpregados.py

Faz o registro de vários empregados e calcula: Valor da folha de pagamento, maior salário, menor salário e salário médio.

```
'''
```

```
class ClsEmpregado: #cria a classe de empregados
    def __init__(self, nome, matricula, setor, salario):
        self.nome = nome
        self.matricula = matricula
        self.setor = setor
        self.salario = salario

#-----
def VerificarMatricula(matricula, listaEmpregados):
    for Empregado in listaEmpregados:
        if(Empregado.matricula == matricula):
            return True #matrícula já existe
    return False

#=====
#Programa principal
numEmpregados = abs(int(input("Entre com o número de empregados: ")))
LstEmp = [] #cria lista em vazio para conter os empregados
somaSalarios = 0

#Registro dos empregados
for _ in range(numEmpregados):
    nome = input("Entre com o nome do empregado: ")
    while(True):
        matricula = int(input(f"Entre com o número da matrícula de
                               {nome}: "))
        if(not( VerificarMatricula(matricula, LstEmp))):
            break
        else:
            print("Matrícula já existe. Por favor, digite outra.")
    setor = input(f"Entre com o setor do empregado {nome}: ")
    salario = float(input(f"Entre com o salário de {nome}: "))
    #Cria a variável-registro "Empregado" para conter os dados de um empregado
    Empregado = ClsEmpregado(nome, matricula, setor, salario)
    LstEmp.append(Empregado)
    somaSalarios += salario
    print()

#Cálculos
maiorSal = max(Empregado.salario for emp in LstEmp)
menorSal = min(Empregado.salario for emp in LstEmp)
nomeMaiorSal = next(Empregado.nome for Empregado in LstEmp if
                    Empregado.salario == maiorSal)
nomeMenorSal = next(Empregado.nome for Empregado in LstEmp if
                    Empregado.salario == menorSal)
salMedio = somaSalarios / numEmpregados

#Resultados
print()
print("\nMaior salário:", maiorSal, "==> ", nomeMaiorSal)
print("Menor salário:", menorSal, "==> ", nomeMenorSal)
print("Folha de pagamento:", somaSalarios)
print("Salário médio:", salMedio)

#Fim do programa "RegistroEmpregados" -----
```