Python

from MinhasFunçoes import *

#Este metodo é muito importante, pois ele importa de outro arquivo phiton, neste caso o * diz para importar tudo do arquivo MinhasFunçoes, assim conseguindo usar elementos presentes em outro arquivo, lembrando que o essencial é digitar corretamente o nome como está salvo o arquivo

A entrada vem em formato de String

```
inteiro1 = (input ("Digite o numero:"))

inteiro2 = (input ("Digite o numero:"))

print(inteiro1) print(inteiro2)
```

Transformando entrada em inteiro ou float

```
inteiro1 = int(input ("Digite o numero:"))

inteiro2 = int(input ("Digite o numero:"))

inteiro3 = float(input ("Digite o numero:"))

print(inteiro1) print(inteiro2) print(inteiro3)

soma = (inteiro1 + inteiro2) print(soma)
```

Adicionando valores em frases

Para adicionar valores em uma frase existem alguns métodos, suponhamos que:

dia = 31

nome = Marcos

print(”Hoje é dia 31, no dia 31 do mês passado o gato do Marcos morreu”)

Para imprimir esta frase usando os valores das variáveis temos que fazer:

print(”Hoje é dia %d, no dia %d do mês passado o gato do %s morreu”) # A letra após o símbolo % quer dizer o tipo da variável, d é de inteiro, s de string e temos o f que é de ponto flutuante

ou

print(”Hoje é dia {}, no dia {} do mês passado o gato do {} morreu”.format(dia, dia, nome)) #Observa-se que tivemos que informar a variável dia duas vezes, pois ela esta presente em dois momentos do codigo

ou

print(”Hoje é dia {0}, no dia {0} do mês passado o gato do {1} morreu”.format(dia, nome)) #Observa-se que a variável dia foi passada apenas uma vez, pois na frase dentro das colchetes informamos o index que esta declarado no format, no caso o primeiro index, que é o 0, é a melhor forma de se fazer, pois ocorre menos risco de você trocar o lugar das variáveis

ou

print(”Hoje é dia {dia}, no dia {dia} do mês passado o gato do {nome} morreu”.format(dia = dia, nome = nome)) #Este caso serve para ficar mais legível para quem ler o seu código, você coloca uma variável local ali dentro, e no final atribui o valor dele, pelos valores das suas variáveis já declaradas, também pode ser usado com dicionários, nesse caso, no final ao invez de atribuir valor por valor as variáveis, você apenas atribui o dicionário

Exemplo com dicionário

```
dados = {“dia” : 31, “nome” : “Marcos”}
print(”Hoje é dia {dia}, no dia {dia} do mês passado o gato do {nome} morreu”.format(**dados)) #Utiliza o dicionário para atribuir os valores
```

ou

```
print(f”Hoje é dia {dia}, no dia {dia} do mês passado o gato do {nome} morreu”) #Usando o f antes da sua frase, você não precisa lançar o format no final, e ela já substitui os nomes das variáveis colocadas dentro dos colchetes diretos, sem precisar ser atribuídos novamente
```

Funções em strings

nomestring.upper() Deixa a string inteira em maiúsculo

nomestring.lower() Deixa a string inteira em minúsculo

nomestring.title() Deixa a string apenas com a primeira letra em maiúsculo e o resto em minúsculo

nomestring.strip() Retira os espaços em brancos da esquerda e da direita

nomestring.lstrip() Retira os espaços em brancos da esquerda

nomestring.rstrip() Retira os espaços em brancos da direita

nomestring.center(10, “#”) Função utilizada para definir quantos caracteres você quer na sua String, como por exemplo você pega a string “Python”, ela tem 6 caracteres, utilizando essa função você transforma ela em uma string com 10 caracteres e coloca a sua string python no centro, nesse caso como definimos o “#” ficaria “##Python##”, mas se não houvéssemos definidos, iria ser completo por espaços em brancos como nesse caso “ Python “

“!”.join(nomestring) Neste caso você utiliza a função .join para adicionar algo a cada caractere da sua string, neste caso por exemplo esta sendo adicionado o “!” e a string “Python” ficaria “P!y!t!h!o!n”, pode ser usado em strings, listas e outros

str.zfill() Método para preencher com 0 a esquerda uma string, usaria como (numero_string = str(nome_string).zfill(2)) atribui zeros ate ela ficar com dois caractere

replace() Para remover um caractere específico de uma string em Python, você pode usar o método replace(). Este método permite substituir um caractere ou uma sequência de caracteres por outra, incluindo uma string vazia ("") para remover o caractere.

Fatiamento de Strings

Agora vamos ver opções de fatiar as Strings em seus caracteres utilizando como exemplo a variável:

Nome = “Marcos Augusto Lino de Oliveira”

nome[0] #Desta forma você diz o index do caractere que deseja saber, nesse caso o index 0 é a primeira letra o “M”, se fosse 4, seria o “o”

resp”M”

nome[:9] #Desta forma como você não atribuiu um valor antes dos dois pontos ele buscara do inicio ate parar no index desejado(como o index foi 9 o ultimo a ser mostrado sera o 8)

resp”Marcos Au”

nome[10:] #Colocando apenas o numero antes do dois pontos ele vai atribuir os caracteres do index 10 adiante

resp”usto Lino de Oliveira”

nome[10:16] #Desta forma ele pegara os caracteres do index 10 adiante ate o 16, então ele pegara os caracters do index 10, 11, 12, 13, 14, 15

resp”usto L”

nome[10:18:2] #Desta forma você imprimi os caracteres do 10 ao 18 mas com o intervalo de dois a cada caractere, ou seja pega o caractere do index 10, dai pega do index 12, 14 e assim por diante

resp”ut i”

nome[:] #Como não há valor atribuído imprimi ela inteira

resp” Marcos Augusto Lino de Oliveira”

nome[::-1] #Desta forma você consegue espelhar a sua String, como não há valores atribuídos ele imprimiria sua String inteira, mas como foi informado que o intervalo é -1, ele vai imprimir de tras para a frente

resp” arievilO ed oniL otsuguA socraM”

Strings de múltiplas linhas ou triplas

Elas são algumas strings utilizadas para falar uma coisa, como por exemplo:

```
nome = "Marcos Augusto Lino de Oliveira"

mensagem = f"""

  Olá meu nome é {nome},

Eu estou aprendendo Python.

        Essa mensagem tem diferentes recuos.
```

"""

print(mensagem)

Sua resposta vai ser:

Olá meu nome é Marcos Augusto Lino de Oliveira,

Eu estou aprendendo Python.

Essa mensagem tem diferentes recuos.

Observa se que utilizando esse método a formatação de espaçamentos e quebras de linhas, não mudam quando você a imprimila

Formatando o número de casas

inteiro1 = int(input ("Digite o numero a ser somado:"))

inteiro2 = int(input ("Digite o numero a ser somado:"))

inteiro3 = float(input ("Digite o numero a ser dobrado:"))

print(inteiro1) print(inteiro2) print(inteiro3) soma =

(inteiro1 + inteiro2) dobro = (inteiro3 * 2) print("Soma =

{}".format(soma)) print("Dobro = {}".format(dobro))

print("Dobro = {:.2f}".format(dobro))

#Tambem pode ser formatado colocando o f no começo, e ao inves do .format, você já informa o nome da variável dentro do colchetes junto com a formatação como por exemplo:

Print(f"Dobro = {dobro:.2f}")

Exercício transformar temperatura de graus Fahrenheit em graus celsius

```
temperaturaf = float(input ("Digite a temperatura em Fahrenheit:"))

temperaturac = ((temperaturaf - 32)/1.8) print ("A temperatura em

Celsius é de {:.2f}".format(temperaturac))
```

Exercício: receber 3 notas e calcular a média

```
n1 = int (input("Digite o 1º numero:")) n2 =

int (input("Digite o 2º numero:")) n3 = int

(input("Digite o 3º numero:")) media = ((n1

+ n2 + n3)/3) print ("A média é de

{:.2f}".format(media))
```

Exercício calculando média de dois números com o primeiro número com peso 2 e o segundo peso 3

```
n1 = int (input("Digite a 1º nota:")) n2 = int

(input("Digite a 2º nota:")) media = (((n1 * 2) +

(n2 * 3)) / 5) print ("A sua média é de

{:.2f}".format(media))
```

Operadores padrão

```
print(3 + 2)
```

#soma

```
print(3 - 2)
```

#subtração

```
print(3 / 2)
```

#divisão

```
print(3 * 2)
```

#multiplicação

```
print(3 // 2)
```

 # resultado inteiro da divisão

```
print(3 % 2)
```

#resto divisão

```
print(3 ** 2)
```

#potência

```
print(pow(3, 2))
```

#Potência

Operadores de comparação

```
print(3 == 2)
```

 # igual

```
print(3 != 2)
```

#não igual

```
print(3 > 2)
```

#maior

```
print(3 < 2)
```

#menor

```
print(3 >= 2)
```

#maior igual

```
print(3 <= 2)
```

#menor igual

Atribuição

```
x = 2
```

#atribuição simples

```
x += 2
```

#Atribuicão de soma x = x + 2

```
x -= 2
```

#Atribuicão de subtração x = x - 2

```
x /= 2
```

#Atribuicão de divisão x = x / 2

```
x *= 2
```

#Atribuicão de multiplicação x = x * 2

```
x //= 2
```

#Atribuicão de resultado inteiro da divisão x = x // 2

```
x %= 2
```

#Atribuicão de resto divisão x = x % 2

```
x **= 2
```

#Atribuicão de potência x = x ** 2

Operador logico

```
print(True and False)

#operador "e"

print(True or False)

#operador "ou"

print(not False)

#operador "nao"

print(not False or True)

#operador "nao ou"

print(not False and True)

#operador "nao e"
```

#Importante o método True e False, só funcionam se começarem com letra maiúscula

Estrutura if, elif e else

```
idade = int(input("Digite sua idade: ")) if

idade > 17:

    print("Você esta na maioridade") else:

    print("Você ainda é de menor")



idade = int(input("Digite sua idade: ")) if

idade > 17:
```

```
   print("Você esta na maioridade") elif
12 <= idade <= 17:
   print("Você esta na adolescencia") else:
   print("Você ainda é de menor")
```

Exercício aumento de salários na empresa

```
c = 1 salario = float(input("Digite o valor do seu salario:
R$")) print("Digite o numero correspondente ao seu
cargo:") print("[1] Programador") print("[2] Analista de
sistemas") print("[3] Analista de banco de dados") cargo
= int(input()) if cargo == 1:
   salario = (salario * 1.5) elif
cargo == 2:
   salario = (salario * 1.4) elif
cargo == 3:
   salario = (salario * 1.3) else:
   c = 0 if
c == 1:
   print ("O seu novo salario é de R${:.2f}".format(salario)) else:
   print ("Cargo invalido")
```

Exercício calcular peso ideal

```
c = 1 altura = float(input("Digite a sua
altura: ")) sexo = input("Digite o seu sexo:
[M/F] ") sexo = sexo.upper()
```

#Deixa a string inteira em maiuscula

sexo.lower() #Deixaria a string inteira em minusculo

sexo.title() #Deixaria apenas a primeira letra em maiúsculo e o resto em minusculo

```
if (sexo == "M"):
   pesoideal = ((72.7 * altura) - 58) elif
(sexo == "F"):
   pesoideal = ((62.1 * altura) - 44.7) else:
   c = 0 if
c == 1:
   print("O seu peso ideal é de {:.2f}".format(pesoideal)) else:
   print("Sexo invalido")
```

Estruturas for e while

```
for x in range(10 ,15):
   print("Oi")
print (x)
```

#Estrutura para, com o comando range, que significa no intervalo de 10 a 15, sendo os números de 10 a 14, ou usar range (5), seria do 0 ao 4, isso atribuindo o valor ao x

```
for x in range(1, 11, 2):
   print(x)
```

#Função range usando os números de 1 a 11 com intervalo de 2, percebe-se que o número 11, não se inclui

```
print("Covid-19") pcovid = 0 nump = int(input("Informe a
quantidade de pacientes:")) for x in range(nump):
   tosse = int(input("Tosse? \n 1.Sim \n 2.Não \n Resp.: "))     febre =
int(input("Febre? \n 1.Sim \n 2.Não \n Resp.: "))     resp =
int(input("Dificuldade em respirar? \n 1.Sim \n 2.Não \n Resp.: "))     if tosse
== 1 and febre == 1 and resp == 1:
       pcovid += 1
print("\nSuspeitos de COVID-19: {}".format(pcovid))
```

#O \n dentro das aspas tem a função de pular uma linha e pode ser usado em qualquer lugar dentro das aspas, mas se utilizado no meio, vai mandar a outra metade da frase para a linha abaixo

```
cont = 0 while
cont < 55:
   print("Palmeiras campeão!")
cont += 1
```

Criar uma estrutura enquanto utilizando contador

```
cont = 1 while
cont <= 5:     if
cont % 2 == 0:
       print("%d é par." %cont)
```

#Neste caso o %d esta transformando o %cont em decimal

```
   cont += 1
```

```
print("Covid-19") pcovid = 0 c = 1 nump =
int(input("Informe a quantidade de pacientes:")) while c <=
nump:
   print("%dº paciente:" %c)     tosse = int(input("Tosse? \n 1.Sim \n 2.Não \n
Resp.: "))     febre = int(input("Febre? \n 1.Sim \n 2.Não \n Resp.: "))     resp =
int(input("Dificuldade em respirar? \n 1.Sim \n 2.Não \n Resp.: "))     if tosse
== 1 and febre == 1 and resp == 1:
       pcovid += 1
c += 1
print("\nSuspeitos de COVID-19: {}".format(pcovid))



print("Covid-19")
pcovid = 0 while
True:
   tosse = int(input("Tosse? \n 1.Sim \n 2.Não \n Resp.: "))     febre =
int(input("Febre? \n 1.Sim \n 2.Não \n Resp.: "))     resp =
int(input("Dificuldade em respirar? \n 1.Sim \n 2.Não \n Resp.: "))     if tosse
== 1 and febre == 1 and resp == 1:
       pcovid += 1     resp = input("Ainda ha pacientes para serem
atendidos? [S/N]")     if resp.upper() == "N":
       break  #Usado para parar a estrutura de repetição print("\nSuspeitos de
COVID-19: {}".format(pcovid))
```

Calcular a média usando break

```
c = 1 nt = 0 while True:     n = int(input("Digite a
sua %dº nota: \n" %c))     nt += n
   resp = input("Deseja informar mais alguma nota? [S/N]\n")     if
resp.upper() == "N":
       break     c += 1 m = nt / c print("A
sua média é de {}".format(m))
```

Pedir para o usuário digitar 5 números inteiros para o programa definir qual é par

```
c = 1 p = 0
while True:
   n = int(input("Digite o %dº numero: \n" %c))
if n % 2 == 0:
       p += 1
if c >= 5:
break     c
+= 1
print("A quantidade de numeros pares digitados são de {}".format(p))
```

Construir um programa que leia sexo e idade

```
print("Digite uma idade negativa para encerrar")
idadeh = 0 moca = 0 idadem = 0 while True:
   idade = int(input("Digite a sua idade: \n"))
if idade < 0:
       break
```

```
   sexo = input("Digite o seu sexo: [M/F} \n")
if sexo.upper() == "M":         if idade >= 18:
idadeh += 1     elif sexo.upper() == "F":
       idadem += idade
moca += 1     else:
       print("Sexo invalido")
mediaidadem = idadem / moca print("A quantidade de homens maiores de 18
anos são de {}".format(idadeh)) print("A media de idade das mulheres são de
{:.2f}".format(mediaidadem))
```

Listas

#Criando uma lista vazia

```
notas = []
```

#Adicionando elementos a uma lista

#Index: 0 1 2 3

notas = [8.3, 7.1, 2.25, 4]

#Elem: 1 2 3 4

#Acessando elementos de uma lista

```
print(nota[1])  #Puxa a nota do index 1
```

#mudando o valor da nota do index desejado, observação o index é a localização do elemento na lista

```
notas[2] = 7.5
```

#imprimindo tudo que está na lista(vetor)

```
print(notas)
```

#imprimindo de uma em uma

```
for n in range(4):
print(notas[n])
```

Lista região sudeste

```
estados = ["São paulo", "Rio de Janeiro", "Minas Gerais", "Espirito Santo"]
print(estados) estados[0] = "São Paulo" estados[1] = "Rio de Janeiro"
estados[2] = "Belo Horizonte"
estados[3] = "Vitoria" for
n in range(4):
print(estados[n])
```

```
nomes = ["Marcos", "Samuel", "Antonio"] nomes.append("Jose")  #O metodo
append adiciona um valor ao final da lista nomes.insert(1, "Joao")
```

#O método insert adiciona o valor no index informado, observação ele não substitui o valor lá dentro ele apenas joga o elemento presente naquele index, um index para frente, assim como o resto da lista

```
print(nomes)
```

```
lindas = []                          #index
lindas.append("Paola Oliveira")      #0
lindas.append("Virginia")            #2
lindas.append("Mel maia")            #3
lindas.insert(1, "Iza")
```

```
#1 for elas in range(4):

print(lindas[elas])



lindas = [] while

True:

   nome = input("Digite o nome de uma linda: \n")     lindas.append(nome)

   resp = input("Deseja continuar? [S/N]\n")

if resp.upper() == "N":

      break for

nome in lindas:

          #Desta forma ele pega os nomes em lindas e leva para nome

   print(nome)



cidades = [] cidades =

["Taubaté"]

cidades.append("São luiz")

cidades.append("Lagoinha")

cidades.insert(0, "São Paulo")

cidades. insert(2, "São jose")

cidades.sort()

#Organiza em ordem decrescente, observa se que a letra minúscula vem
depois da maiúscula, desse jeito "São Paulo" e "São jose" ficaria como
São Paulo na frente, pois todas as letras maiúsculas vêm antes das
minúsculas #Organiza as informações em ordem crescente
cidades.sort(reverse = True)

for cidade in cidades:
```

```
    print (cidade)



import Random
                    #importando a classe random e a utilizando
random.shuffle()
                    #Modo utilizado para embaralhar os elementos
cidades = [] cidades =
["Taubaté"]
cidades.append("São luiz")
cidades.append("Lagoinha")
cidades.insert(0, "São Paulo")
cidades. insert(2, "São jose")
cidades.sort(reverse = True)
import random
random.shuffle(cidades) for
cidade in cidades:
    print (cidade)



nomes = [] while
True:
    nome = input("Digite o nome do aluno: \n")
nomes.append(nome)     resp = input("Deseja
continuar? [S/N]")     if resp.upper() == "N":
        break
nomes.sort() print
(nomes)
```

#Lista de nomes de alunos de uma classe

idade = [] idade.append(17)

idade.append(35)

idade.append(48)

idade.append(17)

idade.append(56)

idade.append(23)

idade.append(19)

print("A quantidade de elementos nesta lista é {}".format(len(idade)))

#A função len mostra a quantidade de coisas dentro da lista

print("A quantidade do numero 17 na lista idade é de {}".format(idade.count(17)))

#A função count, conta quantos argumentos iguais os da sua busca, estão presentes na lista, nesse caso, quantos 17 há lá

print("A menor idade na lista é {}".format(min(idade)))

#função min acha o menor valor presente na lista

print("A maior idade na lista é {}".format(max(idade)))

#Função max acha o maior valor presente na lista

print("A soma das idades da lista é {}".format(sum(idade)))

#Função sum soma todos os valores presentes na lista

print("A media das idades na lista é {:.2f}".format(sum(idade) / len(idade)))

#Dividindo a função sum (soma) pela função len (quantidade de elementos na lista) você consegue a média dos valores presentes na lista

Idade.extend([15, 16, 21])

#A função extend serve para você adicionar vários itens de uma vez em uma lista já existente, no caso você consegue adicionar uma lista a uma lista de uma vez

```
numero = [] while

True:

   n = int(input("Digite o numero:"))

if n > 0:

      numero.append(n)
   resp = input("Deseja continuar? [S/N]")

if resp.upper() == "N":

      break

print("A média geometrica entre o maior e menor numero digitado é de
{:.2f}".format((max(numero) + min(numero))/ 2))
```

```
idade = [] idade.append(17)

idade.append(35)

idade.append(48)

idade.append(17)

idade.append(56)

idade.append(23)

idade.pop()
```

#Apaga um elemento da lista, como não tem índice se referindo, apaga o último

```
print(idade) idade.pop(2)
```

#Apaga o elemento no index 2 da lista, nesse caso o 48

```
print(idade) idade.remove(35)
```

#Remove o elemento pedido na lista, nesse caso vai remover o 35 da lista, também pode ser usado com strings, elimina só o ultimo, ou seja, se houver mais de um 35, eliminara só o ultimo (com index mais alto)

```
for ida in idade:

if ida == 35:

    idade.remove(35)
```

#Desta forma apagamos todos os 35 da lista

```
print(idade) idade.clear()
```

#Apaga todos os elementos da lista

```
print(idade)



nadadoras = []

tempo = [] for i

in range(5):

  nome = input("Nome: ")

t = float(input("Tempo: "))

nadadoras.append(nome)

tempo.append(t)

indice_mt = tempo.index(min(tempo))
```

#a função tempo.index serve para achar o index de determinado elemento, neste programa está procurando o index do menor valor presente na lista tempo, por isso é utilizado a função min (tempo)

```
indice_pt = tempo.index(max(tempo))
print("A melhor nadadora é {}".format(nadadoras[indice_mt]))
```

#Neste caso está sendo pedido para imprimir o elemento presente na index achada anteriormente e salva na variável "indice_mt", assim conseguimos descobrir o nome da nadadora com o pior tempo

```
print("A pior nadadora é {}".format(nadadoras[indice_pt]))
```

```
produtos = [] while
True:     nome =
input("Nome: ")
produtos.append(n
ome)     resp =
input("Deseja
continuar? [S/N]")
if resp.upper() ==
"N":        break
qt_elem = len(produtos) for
ind in range(qt_elem):
    print("{} --> {}".format(ind, produtos[ind]))
```

#Usando esse método você consegue listar os produtos de uma lista de acordo com seu index, ele vai rodar todos os índices da lista, descobrindo o tamanho dela pela função len, e a cada teste mostra o elemento naquele index, começa do 0 até o número total da lista, e no final esse número é mostrado, e o elemento presente naquela posição

```
for indice, valor in enumerate(produtos):
print("{} --> {}".format(indice, valor))
```

#Usando a função enumerate, você consegue fazer o método anterior de forma mais fácil, pois ele vai buscar em todos os elementos da lista e retornar os valores do índice e os valores dos elementos sozinho, basta imprimir, neste caso seria o índice e o valor, ambos estão tendo algum valor atribuído no for

#Se tratando de listas, quando você quer retirar elementos pa fazer uma lista de outra lista, ao invés de você fazer um if já pode fazer tudo junto como por exemplo:

```
numeros = [2, 4, 5, 7, 12, 16, 21, 32]

pares = [numero for numero in numeros if numero % 2 == 0]
```

#Tambem conseguimos montar listas de formas diferentes como por exemplo

```
letras = list(“python”) #Faz uma lista contendo os caracteres da palavra python, então ficaria no index 0 ‘p’, no 1 “y” e assim por diante

números = list(range(10)) #Faz uma lista contendo os números de 0 a 10
```

QUESTÃO 1. Um professor de Matemática deseja construir um programa para gerar uma progressão Aritmética (PA). Para isso, devem ser informados 3 parâmetros de entrada: a) primeiro termo da PA, b) quantidade de termos da PA e c) razão dessa PA. Construa um programa para carregar e imprimir uma lista contendo os termos da PA, bem como a soma dos elementos da PA.

```
primeirot = int(input("Digite o primeiro termo da PA: \n")) quantidadet =

int(input("Digite a quantidade de termos na PA: ")) razao =

int(input("Digite a razão dessa PA: ")) pa = [] for i in range(quantidadet):

pa.append(primeirot)     primeirot = primeirot + razao for indice, valor in

enumerate(pa):     print("{} --> {}".format(indice, valor)) print("A soma

dos elementos dessa PA, são de {}".format(sum(pa)))
```

QUESTÃO 2. Os professores de Educação Física estão organizando uma seletiva para montar a equipe de natação. Para isso, eles convocaram os 7 melhores tempos da última competição e marcaram o tempo de cada um dos nadadores, na prova dos 25 metros, estilo nado livre.

```
nadadores = []
tempo = [] for i
in range(7):
   nome =input("Digite o nome do nadador: \n")     t = float(input("Digite o tempo do
nadador: \n"))     nadadores.append(nome)     tempo.append(t) indice_mt =
tempo.index(min(tempo)) indice_pt = tempo.index(max(tempo)) print("O melhor
nadador é {} com um tempo de {}".format(nadadores[indice_mt], tempo[indice_mt]))
print("O pior nadador é {} com um tempo de {}".format(nadadores[indice_pt],
tempo[indice_pt])) print("O tempo medio dos nadadores é de
{:.2f}".format(sum(tempo)/len(tempo)))
```

QUESTÃO 3. Uma turma de formandos está vendendo rifas para angariar recursos financeiros para sua cerimônia de formatura. Construa um programa para cadastrar os nomes das pessoas que compraram a rifa. Ao fim, o programa deve sortear o ganhador do prêmio e imprimir o seu nome.

```
nomes = [] while
True:
   nome = input("Digite o nome da pessoa: \n")
nomes.append(nome)     resp = input("Deseja
continuar? [S/N]")     if resp.upper() == "N":
      break
```

```
import random random.shuffle(nomes) sorteado
= random.choice(nomes) print("O ganhador da
rifa é {}".format(sorteado))
```

QUESTÃO 4. Crie um programa que solicite o usuário um número N ímpar, maior que 1. Em seguida, preencha uma lista com N números inteiros positivos. Selecione o elemento que está no centro da lista. Ao final, calcule e escreva o fatorial do elemento selecionado.

```
n = int(input("Digite um numero inteiro impar maior que 1: \n")) if
n % 2 == 0:
   print("O numero digitado é par, o programa sera finalizado") else:
   l = []    tot = 1    for i in range(n):        num =
int(input("Digite o numero inteiro positivo: \n"))
l.append(num)
   l.sort()    fat = l[int(n/2)]    print("A
fatorial de {} é: \n".format(fat))    while
fat > 0:        if fat > 1:
          print("{} * ".format(fat))
else:
          print("{} =
".format(fat))        tot *= fat
fat -= 1
print("{}".format(tot))
```

```
notas = [] n = int(input("Digite a quantidade de notas a serem
computadas: \n")) c = 1 while c <= n :
```

```
 nota = int(input("Digite a sua {}° nota: \n".format(c)))
notas.append(nota)   c+= 1
soma = sum(notas) print("A sua média é de
{:.2f}".format(sum(notas)/len(notas)))
```

Matrizes

Para criar uma matriz, você utiliza basicamente os mesmos métodos que para criar uma lista, lembrando que uma matriz, são como várias listas juntas, mais divididas

```
m = [[0,2],[1,3],[4,5]]
print(m) print(m[1][0])
```

#Dessa forma o programa vai imprimir o primeiro número da segunda lista, pois o [1] representa o index da lista, e o [0] o index dos números, nesse caso da 1

```
l1 = [10, 2]
l2 = [21, 13]
l3 = [7, 9] m
= [l1]
m.append(l2)
m.insert(2, l3) print(m)
```

#Outra forma de fazer uma matriz é criar várias listas e depois adicioná-las em uma matriz só, usando os mesmos métodos de adicionar números em uma lista

```
import random
while True:
   lin = int(input("informe a quantidade de linhas da matriz: \n"))
col = int(input("informe a quantidade de colunas da matriz: \n"))     if
lin > 0 and col > 0:        m = []        for i in range(lin):
          linha = []
for j in range(col):
             num = random.randint(1,21)
```

#Funçao utilizada para gerar números aleatórios inteiros, nesse caso entre 1 e 21 ou seja de 1 a 20

```
             linha.append(num)
m.append(linha)        break
print(m)
```

#Dessa forma montamos uma matriz aleatória

```
import random
```

#Importamos o método random

```
while True:
   lin = int(input("informe a quantidade de linhas da matriz: \n"))
col = int(input("informe a quantidade de colunas da matriz: \n"))     if
lin > 0 and col > 0:        m = []        for i in range(lin):
          linha = []
for j in range(col):
num =
random.randint(1,21)
linha.append(num)
```

```
m.append(linha)

break for i in range(lin):

for j in range(col):

    print(m[i][j], end = " ")
```

#Função end, neste caso manda imprimir o espaço em branco no final, após o número

```
print("")
```

#Dessa forma se formata uma matriz direitinho

x1 x2 x3

Em uma matriz m = [3, 4, 7] y1

[7, 8, 15] y2 [3,

12, 4] y3

#Para uma diagonal primaria em uma matriz, a condição é que o valor de x e y sejam iguais, como por exemplo x1, y1.

#Lembrando que para uma matriz ter diagonal ela tem que ser quadrada, ou seja, ter a mesma quantidade de linhas e colunas.

#Para uma diagonal secundaria na matemática a condição é que a soma de x e y seja o valor da ordem da matriz, (nesta seria 3, pois é uma matriz 3 x 3), mais 1, nesse caso 4, como por exemplo x1, y3, ou x2, y2, ou x3, y1 #Para elementos abaixo da diagonal principal, a condição é q o valor de x seja menor que o valor de y, como x1, y2, ou x2, y3 #Para elementos acima da diagonal principal, a condição é q o valor de x seja maior que o valor de y, como x2, y1, ou x3, y2

Questão criar uma matriz onde ache o maior número da diagonal principal, o menor da diagonal secundaria e tire a média entre eles import

```
random ordem = int(input("Digite a ordem da matriz: ")) if ordem >= 2:     while True:
m = []       for i in range(ordem):
        linha = []          for j in
range(ordem):               num =
random.randint(1,21)
linha.append(num)
m.append(linha)       break    dp = []
ds = []    for i in range(ordem):
for j in range(ordem):
print(m[i][j], end = " ")          if i == j:
            dp.append(m[i][j])
elif i + j == (ordem - 1):
```

#A diagonal secundaria em python, pôr os index começar com 0, será a ordem – 1 não igual na matemática que é a ordem + 1

```
            ds.append(m[i][j])
print("")
   maior_dp = max(dp)    menor_ds =
min(ds)    media = (maior_dp +
menor_ds) / 2    print("Média =
{:.2f}".format(media)) else:
   print("Informe uma ordem valida: ")
```

Tuplas

As tuplas são muito parecidas com as listas, mas ela é imutável, ao cria-la, ela vai permanecer com aqueles valores, ate o fim do programa, você não pode alterar aqueles valores

Voce a declara com parênteses, como por exemplo

Frutas = (“maça”, “laranja”, “pera”, “banana”,) #Observa se que ela sempre termina com uma virgula, isso é feito como boa pratica, pois existem tuplas de apenas um elemento e o compilador confundiria com uma string normal

Pais = (“Brasil”,)

Países = tuple([“Argentina”, “Peru”, “Venezuela”]) #Tambem pode se utilizar o método tuple para criar uma tuple

Matriz de tuplas, elas são praticamente idênticas as outras que são de lista como por exemplo

```
Matriz = (
(1, 3, “c”),
(5, “b”, “e”),
(7, 8, “a”),) #Uma matriz em tupla
```

Ela é utilizada quando os elementos não podem ser mudados, daí usa esse método ao invés de listas

Dicionários

#Criando um dicionário, com o dicionário, você consegue, associar elementos a outros deixando assim mais fácil a busca, como neste caso ao imprimir a busca 23 no dicionário vai aparecer o nome marcos

```
idade = {23: "Marcos", 22: "Samuel"} idade.update({22:
"Samuel Augusto"})
```

#O método update pode ou adicionar um novo elemento no dicionário ou atualizar um antigo ele checa se aquele primeiro elemento de busca já existe, se existir ele atualiza se não ele cria um

```
idade.update({25: "joao"})
print(idade) print(idade[23])
print(idade.get(23)) contato
= {"@camilaqueiroz":
"Camila Queiroz",
        "@paollaoliveira": "Paola de Oliveira"}
contato.update({"@bruna_ionica": "Bruna Marquezine"})
insta = input("Digite um instagram: ") if insta in contato:
   print("Este instagram é de {}".format(contato.get(insta)))
```

```
contato = {"@camilaqueiroz": "Camila Queiroz",
        "@paollaoliveira": "Paola de Oliveira",
        "@sheronmenezzes": "Sheron Menezes"} print("Antes
de incluir Marquezine: {}".format(len(contato)))
```

#Função len para ver quantos elementos tem em um dicionário

```
contato.update({"@bruna_ionica": "Bruna Marquezine"}) print("Apos
incluir a Marquezine: {}".format(len(contato)))
```

```
contato = {"@camilaqueiroz": "Camila Queiroz",
        "@paollaoliveira": "Paola de Oliveira",
        "@sheronmenezzes": "Sheron Menezes",
"@bruna_ionica": "Bruna Marquezine"} for insta
in contato.keys():
```

#Essa função keys, mostra todas as chaves de busca de determinado dicionário, nesse caso vai mostrar os Instagram cadastrados

#Tambem temos a função .fromkeys, com ela você informa as chaves que quer adicionar no seu dicionário sem atribuir valor ainda

```
   print(insta) for atriz in
contato.values():
```

#Essa função values, mostra todos os valores dentro do dicionário, é o contrário da keys que mostra apenas a chave

```
   print(atriz) for insta, atriz in
contato.items():
```

#Este método items, mostra todos os itens no dicionário, ou seja, vai mostrar tanto as chaves como os valores, lado a lado

```
   print("O instagram de {} é {}".format(atriz, insta)) for
insta, atriz in sorted(contato.items()):
```

#A função sorted é utilizada para ordenar os elementos do dicionário com base na chave

```
   print("O instagram de {} é {}".format(atriz, insta))



contato = {"Camila Queiroz": 1.77,
        "Paola de Oliveira": 1.70,
        "Sheron Menezes": 1.67,
```

```
        "Bruna Marquezine": 1.70} from
operator import itemgetter
```

#Importa o programa itemgetter para o programa #A função key = itemgetter(1) serve para ordenar os elementos com base nos valores #Neste caso em forma crescente

```
for atriz, estatura in sorted(contato.items(),key=itemgetter(1)):
   print("A altura de {} é {:.2f}m ".format(atriz, estatura)) for atriz, estatura in
sorted(contato.items(),key = itemgetter(1), reverse = True):
```

#Serve para ordenar em forma decrescente, por causa do reverse

```
   print("A altura de {} é {:.2f}m ".format(atriz, estatura))



contato = {"@camilaqueiroz": "Camila Queiroz",
        "@paollaoliveira": "Paola de Oliveira",
        "@sheronmenezzes": "Sheron Menezes",
"@bruna_ionica": "Bruna Marquezine"} backup =
contato.copy()
```

#Função copy é utilizado para fazer uma cópia de determinado dicionário

```
print(backup)
contato.pop("@paollaoliveira")
```

#A função pop é utilizada para excluir determinado elemento de um dicionário, ele exclui a chave e o seu valor

```
print(contato) contato.popitem()
```

#A função popitem é utilizada para excluir o último elemento de um dicionário

```
print(contato) contato.clear()
```

#A função clear é utilizada para esvaziar completamente um dicionário
print(contato)

#A função get serve para buscar itens, e diferente da maneira convencional, se ele não achar a chave ele retorna uma valor vazio ou um valor que você determinar, já no outro método que é apenas chamando, se não houver vai resultar em erro

#A função setdefault, pega uma chave que você definir e adiciona um valor que você informar, caso essa chave já exista ele retorna o valor presente nela, se não houver ele adiciona esse valor e se essa chave não existir ele adiciona a chave e o valor
#A função update serve para atualizar itens ou adicionar em um dicionário, se a chave já existir ele atualiza, se não ele cria uma nova, se na atualização a chave inserida tiver menos parâmetros ele vai excluir os demais e deixar apenas os novos
#Método in ele serve para ver de uma forma elegante se algo existe no dicionário, você põe o valor que deseja saber (valordesejado in contatos) se existir naquela lista vai voltar um True se não um False, com o metodo Count e index, você também consegue utilizar, uma volta a quantidade presente no dicionário e o outro o index de onde estiver esse valor
#A função del serve para eliminar o item especifico que você desejar, você pode informar a chave e logo depois o objeto, dessa forma você exclui apenas o objeto do dicionário (del dicionário[“chave”][“objeto”]), ou informar apenas a chave que excluiria tudo (del dicionário[“chave”]

Questão criar um programa onde o usuário digite código, nome, preço e quantidade de uma compra e no final liste tudo em ordem alfabética e de o valor total

```
produtos = {}

while True:

cod = int(input("Digite o codigo do produto: \n"))

 lista = []

    nome = input("Digite o nome do produto: \n")

lista.append(nome)
```

```
preço = float(input("Digite o valor do produto: \n"))
 lista.append(preço)
quant = int(input("Digite a quantidade a ser vendida: \n"))
 lista.append(quant)
produtos.update({cod: lista})
 resp = input("Deseja continuar? [S/N] \n")
if resp.upper() == "N":
        break total = 0
 for prod in sorted(produtos.values()):
    subtotal = prod[1] * prod[2]
print(prod[0] + ": R${:.2f}".format(subtotal))
 total += subtotal
print (10* "-")
print("Total : R${:.2f}".format(total))
```

Funções

Def é a palavra-chave para criar uma função

```
def saudar():
print("Ola")
```

#Cria uma função que vai imprimir ola

```
def saudar_aluno(nome):
   print("Olá, {}".format(nome))
```

```
def calcula_dobro(numero):
    return 2 * numero


def calcula_dobro_triplo(numero):
return 2 * numero, 3 * numero


def calcula_area_circulo():
    raio = 3     return valor_pi *
pow(raio, 2)


def calcula_comp_circulo(raio):     return
2 * valor_pi * raio
```

#Esses elementos estão comentados aqui, mas estão salvos no arquivo MinhasFunçoes

```
from MinhasFunçoes import *
```

#O comando from Minhas funções import é utilizado para importar as funções do arquivo que eu criei e deixei salvo como "MinhasFunçoes", (Que tem os elementos comentados acima)

```
saudar() saudar_aluno("Marcos")
print(calcula_dobro(3))
print(calcula_dobro_triplo(3)) dobro,
triplo = calcula_dobro_triplo(5) print("O
```

dobro de 5 é {}".format(dobro)) print("O

triplo de 5 é {}".format(triplo))

print(calcula_area_circulo())

print(calcula_comp_circulo(9))

#Variaveis criadas dentro das funções são variáveis locais e so existem dentro delas

```
def retorna_menor_maior_media(n):
return min(n), max(n), sum(n)/len(n)
l = [7, 2, 3, 4]
```

#Utilizando uma lista na função

menor, maior, media = retorna_menor_maior_media(l)

print("Menor elemento é {}".format(menor))

print("Maior elemento é {}".format(maior)) print("A

media dos elementos é {}".format(media))

```
def calcula_dobrox_triploy(x, y):
   return 2*x, 3*y varx = 5 vary = 7 dobrox, triploy =
```

calcula_dobrox_triploy(varx, vary)

#Se a sequência estiver incorreta, os valores retornados serão completamente errados

print("Dobro de {} = {}".format(varx, dobrox)) print("Triplo

de {} = {}".format(vary, triploy))

```
def calcula_imc(peso, altura, nome):
```

```
    imc = peso/pow(altura, 2)
    return "{}, seu IMC é {:.2f}".format(nome, imc) print(calcula_imc(84,
1.82, "Fabio"))
```

Neste caso, se faltar um elemento dos 3 pedidos, a função da erro

```
def calcula_imc(peso, altura, nome = "Prezado(a) "):
```

#Neste caso se o elemento nome não for informada, ele atribui o valor "Prezado(a)"

```
    imc = peso/pow(altura, 2)     return "{}, seu IMC
é {:.2f}".format(nome, imc) print(calcula_imc(84,
1.82, "Marcos"))
```

```
def calcula_imc(peso, altura, nome = "Prezado(a) "):
    "Esta função calcula o IMC"
```

#Este código em aspas duplas dentro da função serve como se fosse a documentação dela, ele não irá aparecer, a não ser que o usuário a importe da maneira escrita mais a frente, ela tem q ser chamada logo abaixo do início da função

```
    imc = peso/pow(altura, 2)     return "{}, seu IMC
é {:.2f}".format(nome, imc)
print(calcula_imc.__doc__)
```

#Forma de imprimir a documentação de determinada função, sem precisar ir ao arquivo dela

```
def calcula_area_losango(x, y):
```

```
   "Esta função serve para calcular a área de um losango"
return "A area desse losango é {}".format(x*y/2) dma =
int(input("Digite a diagonal maior do seu losango: \n")) dme =
int(input("Digite a diagonal menor do seu losango: \n"))
print(calcula_area_losango(dma, dme))


def calcula_distancia_pontos(x, y ):
   "Essa função serve para achar a distancia de dois pontos em um espaço R^2"     m =
[]
   m.append(x[0] - y [0])
   m.append(x[1] - y [1])     return m n1 = [] n2 = [] c = 1 while c <= 2:     x
= int(input("Digite a localização do primeiro ponto: (x, depois y)"))
n1.append(x)     c += 1 c = 1 while c <=2:
   x2 = int(input("Digite a localização do segundo ponto: (x, depois y)"))
n2.append(x2)     c += 1
print(calcula_distancia_pontos(n1, n2))


def gera_matriz_aleatoria(x, y):
   "Essa função serve para gerar uma matriz aleatoria"
import random     matriz = []     for i in range(x):
      linha = []
for j in range(y):
         num = random.randint(1, 11)          linha.append(num)
matriz.append(linha)     return matriz l = int(input("Digite o
numero de linhas de sua matriz: \n")) c = int(input("Digite o
```

```
numero de colunas de sua matriz: \n")) matriz = [] matriz =
gera_matriz_aleatoria(l, c) for i in range(l):     for j in range(c):
      print(matriz[i][j], end = " ")
print(" ")
```

Conjuntos

Utilizando a função set é possível fazer conjuntos, conjuntos são basicamente uma lista de elementos únicos, então ao contrario dos vetores e matrizes usando o comando set, os elementos duplicados são eliminados, como por exemplo:

```
set([1, 3, 4, 1, 6, 5, 3]) #Este conjunto vai ter salvo os números ([1, 3, 4, 6, 5])
set(“Abacaxi”) #Por usar diretamente esse comando em apenas uma String, ele vai dividir ela em vários elementos de um caractere, como eliminara os duplicatos, ficara [(‘a’, ‘b’, ‘c’, ‘x’, ‘i’)]
set((“palio”, “gol”, “uno”, “palio”)) #Usando o mesmo conceito ficara (“Palio, “gol”, “uno”)
números = {1, 2, 3, 4, 1} #Tambem podem ser declarados dessa forma
```

Conjuntos não podem ser fatiados, nem serem buscados por index, para realizar estas operações você deve transforma-lo em uma lista da seguinte forma:

```
números = {1, 2, 3, 4, 1}
números = list(números) #Depois dessa operação você consegue utilizar normalmente
```

Para imprimir um conjunto inteiro, utilizar o mesmo método for que as listas

Operação {}.union, serve para unir dois conjuntos como por exemplo:

```
conjunto_a = {1, 2}
conjunto_b = {3, 4}
conjunto_a.union(conjunto_b) #Vai juntar os dois conjuntos e transforma lo em (1, 2, 3, 4)
```

Operação {}.intersection, serve para pegar a intersecção de dois conjuntos, ou seja, somente os valores que são iguais dos conjuntos, como por exemplo:

```
conjunto_a = {1, 2, 3}
conjunto_b = {2, 3, 4}
conjunto_a.intersection(conjunto_b) #Vai pegar os números em comum dos dois conjuntos e transforma lo em (2, 3)
```

Operação {}.difference, serve para pegar diferença dos conjuntos, ou seja, somente os valores diferentes dos conjuntos, como por exemplo:

```
conjunto_a = {1, 2, 3}
conjunto_b = {2, 3, 4}
conjunto_a.difference(conjunto_b) #Vai pegar os números diferentes do conjunto a para o b e transforma lo em (1), pois diferente do conjunto a, o conjunto b so não tem o 1
conjunto_b.difference(conjunto_a) #Vai pegar os números diferentes do conjunto b para o a e transforma lo em (4), pois diferente do conjunto a, o conjunto b so não tem o 4
```

Operação {}.symmetric_difference, serve para pegar diferença dos conjuntos, ou seja, somente os valores diferentes dos conjuntos, como por exemplo:

```
conjunto_a = {1, 2, 3}
conjunto_b = {2, 3, 4}
conjunto_a.symmetric_difference(conjunto_b) #Vai pegar os números diferentes dos dois conjuntos e transforma lo em (1, 4)
```

Operação {}.issubset, serve para ver se um conjunto é subconjunto de outro, ou seja se um conjunto esta totalmente dentro do outro, como por exemplo:

```
conjunto_a = {1, 2, 3}
conjunto_b = {4, 2, 3, 5, 7, 6, 1}
conjunto_a.issubset(conjunto_b) #É verdadeiro, pois o conjunto a é um subconjunto de b, já que esta totalmente presente dentro dele
conjunto_b.issubset(conjunto_a) #É falso, pois o conjunto b não é um subconjunto de b, já que não esta totalmente presente dentro dele
```

Operação {}.issuperset, serve para ver se um conjunto não é subconjunto de outro, ou seja se um conjunto não esta totalmente dentro do outro, como por exemplo:

```
conjunto_a = {1, 2, 3}
conjunto_b = {4, 2, 3, 5, 7, 6, 1}
conjunto_a.issuperset(conjunto_b) #É falso, pois o conjunto a é um subconjunto de b, já que esta totalmente presente dentro dele
conjunto_b.issuperset(conjunto_a) #É verdadeiro, pois o conjunto b não é um subconjunto de b, já que não esta totalmente presente dentro dele
```

Operação {}.isdisjoint, serve para ver se um conjunto não é inteseccionado com outro, ou seja não tem nenhuma ligação com o outro, como por exemplo:

conjunto_a = {1, 2, 3, 4, 5}

conjunto_b = {6, 7, 8, 9}

conjunto_c = {0, 1}

conjunto_a.isdisjoin (conjunto_b) #É verdadeiro, pois o conjunto a não contem nenhum elemento igual ao conjunto b

conjunto_a.isdisjoin (conjunto_c) #É falso, pois o conjunto a contem elementos iguais ao conjunto c

Tambem temos o {}.add que adicionará um numero desde que já não tenha um igual la, temos o clear, o pop, o len, o in e copy que são iguais as funções dele nas listas
Temos a função {}.discard, que serve para descartar um numero especifico, ele é parecido com a função remove, que também pode ser utilizado, a diferença é que o remove da erro se o elemento pedido não existir, o discard não da erro

Trabalhando com Datas

Para trabalhar com data temos que importar o datetime

Dessa forma para trabalhar com data devemos fazer

Import datetime

D = datetime.date(2024, 01, 16)

Tambem podemos importar utilizando o comando from e informar a classe que você quer usar, assim não precisa escrever o datetime toda vez, dessa forma:

```
From datetime import date
D = date(2024, 01, 16)
```

Usando a função today do date, você consegue imprimir a data atual

```
From datetime import date
D = date.today()
```

Mas a classe principal que você vai utilizar e a classe datetime, que é basicamente igual a classe data, mas com a hora atual, também, a função today funciona da mesma forma

Observação a classe datetime, esta dentro do pacote datetime, entt se apenas importar o pacote ira ter que escrever datetime.datetime, ou importar a classe utilizando from como

```
From datetime import date, datetime
D = datetime(2024, 01, 16, 8, 30, 15) #A ordem desse números é ano, mês, dia. Horas, minutos, segundos, se não informar as horas, ele vai preencher em branco
```

Tambem temos a classe time que funciona da mesma forma, mas mostra apenas as horas

Usando a classe timedelta, ela representa a duração de algo, com ele você consegue adicionar ou subtrair, horas dias semanas do seu tempo

Usando o .time em uma data, você imprimi so o horário da sua data, utilizando o .date, você pega so a parte de data

Usando a função strftime(“modelodatadesejada”), você consegue formatar a sua data na qual você deseja, por exemplo, coloca la no padrão brasileiro, para realizar essa formação é bom que você procure a padrão da linguagem no goolge, mas a mais comum com dia, mês, ano, hora, minutos, segundos é

```
data_atual.strftime("%d/%m/%Y %H:%M")
```

Um código de exemplo é:

```
import datetime
import os
while True:
   data_atual = datetime.datetime.now()
   print(f"""
      Data atual {data_atual.strftime("%d/%m/%Y %H:%M:%S")}
       """)
   print("""
         ----------Modelo Carros----------
         |                                |
         |      1 - Carro Pequeno         |
         |      2 - Carro Médio           |
         |      3 - Carro Grande          |
         |                                |
         ---------------------------------
      """)
```

```
tipo_carro = int(input("Digite o codigo do carro: "))
tempo = 0
if tipo_carro == 1:
    modelo_carro = "Pequeno"
    tempo = 30
elif tipo_carro == 2:
    modelo_carro = "Médio"
    tempo = 45
else:
   modelo_carro = "Grande"
   tempo = 60
data_estimada = data_atual + datetime.timedelta(minutes=tempo)
os.system('cls') #Serve para limpar a tela
print(f"""
       ---------------------------------------------
       | Data de chegada: {data_atual.strftime("%H:%M:%S")} |
       |                                                    |
       | Tipo de Carro:                        {modelo_carro} |
       |                                                    |
       | Hora final     : {data_estimada.strftime("%H:%M:%S")} |
       ---------------------------------------------
     """)
resp = input("Novo carro para lavar? [S/N]")
if resp.upper() == "N":
    break
```

Para converter uma String em data, devemos utilizar o comando .strptime, como por exemplo

Date_string = “20/07/2023 15:30

D = datetime.datetime.strptime(date_string, “%d/”%m/%Y %H:%M”) #Observa se que os formatadores dentros das aspas colocados, estão ali para informar que numero é o que ali, e o programa o transformara em um programa com a data no formato padrão, que é ano, mês, dia, hora e minutos, para, deixar de outra forma, terá que formata la #Você também consegue formatar colocando os dados da formatação dentro de uma string e depois a invocando

Falando em datas é muito importante a parte de fuso horários, caso o seu programa seja utilizado internacionalmente, para isso em python usamos a biblioteca pytz, temos que importa la e depois conseguimos utiliza la com o metodo.timezone (“paisdesejado”), lembrando que esse pais tem que ser igual esta na biblioteca, no caso do brasil é “America/Sao_Paulo”

Import pytz

From datetim import datetime

Data = datetime.now(pytz.timezone(‘America/Sao_Paulo)’

Exemplos de algumas funções

```
def analise_vendas(vendas):
    total_vendas = 0
    # TODO: Calcule o total de vendas e realize a média mensal:
    for indice, valor in enumerate(vendas):
```

```
        total_vendas += valor
        media_vendas = total_vendas / len (vendas)
    return f"{total_vendas}, {media_vendas:.2f}"

def obter_entrada_vendas():
    # Solicita a entrada do usuário em uma única linha
    entrada = input()
    lista = entrada.split(",")
    vendas = list(map(int, lista))
    # Método de converter a entrada de strings em uma lista de inteiros:
    return vendas
vendas = obter_entrada_vendas()
print(analise_vendas(vendas))
```

Orientação a Objetos:

Criando Classes

A estrutura base de uma classe é :

```
class bicicleta:
#Para criar uma classe, você utiliza o comando class e o nome da classe que você
deseja criar, logo em seguida você informa as funções dessa classe
    def __init__(self, cor, modelo, ano, valor):
```

```
#Esta é a minha classe inicial onde informo a descrição dos objetos da classe, observa
se que o self é a referencia explicita ao objeto, como o this
        self.cor = cor
#Como por exemplo a cor da bicicleta
        self.modelo = modelo
#O modelo dela
        self.ano = ano
#O ano
        self.valor = valor
#E o valor


    def buzinar(self):
Aqui criamos o método buzinar para ela que pode ser chamado do programa principal
        print ("Biiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiii")


    def parar(self):
Aqui criamos o método parar para ela que pode ser chamado do programa principal
        print("Freiando, freiandoooo, parei")


    def correr(self):
Aqui criamos o método correr para ela que pode ser chamado do programa principal


b = []
b.append(bicicleta("Verde", "Caloi", 2024, 5000))
#Desta forma você atribui os valores pedidos na sua classe ao seu objeto
print(f"Sua bicicleta é {b[0].cor} do modelo {b[0].modelo} e ano {b[0].ano}, ela
atualmente tem o valor de mercado de R${b[0].valor}")
```

#Aqui podemos ver como usar os valores do objeto, no programa principal

b[0].buzinar()

#Aqui como utilizar os métodos

b[0].correr()

b[0].parar()

Adicionando uma função para ficar mais fácil visualizar dentro desse objeto

```
def __str__(self):
    return f"{self.__class__.__name__}: {', '.join([f'{chave} = {valor}' for chave, valor in self.__dict__.items()])}"
```

#Desta forma você consegue imprimir todos os elementos dentro de um dicionário. O método (self.__class__.__name__) busca a classe e informa o seu nome, neste caso bicicleta, mas o metodo class esta se referindo ao objeto, e seus dados, são salvos em uma lista de dicionários, o método seguinte, imprimi todas as variáveis e valores dentro desse objeto

print(b[1])

#Desta forma vai imprimir isto “bicicleta: cor = Verde, modelo = Caloi, ano = 2024, valor = 5000”