tuplas: as tuplas são imutáveis, se você quer fazer o mesmo papel de uma tupla, só que é possível mudar um valor, você vai precisar usar uma lista, as tuplas ficam entre () e as listas ficam entre []

o index serve para ver a posição
o sorted serve para ver uma tupla na ordem alfabética

número por extenso

Crie um programa que tenha uma dupla totalmente preenchida com uma contagem por extenso, de zero até vinte. Seu programa deverá ler um número pelo teclado (entre 0 e 20) e mostrá-lo por extenso.

```
num = int(input('Digite um número entre 0 e 20: '))
escolha = ('zero', 'um', 'dois', 'três', 'quatro', 'cinco', 'seis', 'sete', 'oito', 'nove', 'dez', 'onze', 'doze',
'treze', 'catorze', 'quinze', 'dezesseis', 'dezessete', 'dezoito', 'dezenove', 'vinte')
print(f'Você digitou o número {escolha[num]}')
```

```
Digite um número entre 0 e 20: 18
Você digitou o número dezoito
```

tuplas com times de futebol

Crie uma tupla preenchida com os 20 primeiros colocados da Tabela do Campeonato Brasileiro de Futebol, na ordem de colocação. Depois mostre:
a) A lista do time
b) Os 3 primeiros
c) Os 2 últimos colocados
d) Os times em ordem alfabética
e) A posição do Palmeiras

```
times = ('Corinthians', 'Palmeiras', 'Santos', 'Grêmio')
print(f'Lista de times: {times}')
print(f'Os três primeiros são: {times[0:3]}')
print(f'Os dois últimos colocados são: {times[2:]}')
print(f'Os times em ordem alfabética é {sorted(times)}') o sorted vai colocar na ordem alfabética
print(f'O Palmeiras está na posição: {times.index("Palmeiras")+1}º posição') 2 aspas =  posição
```

```
Lista de times: ('Corinthians', 'Palmeiras', 'Santos', 'Grêmio')
Os três primeiros são: ('Corinthians', 'Palmeiras', 'Santos')
Os dois últimos colocados são: ('Santos', 'Grêmio')
Os times em ordem alfabética é ['Corinthians', 'Grêmio', 'Palmeiras', 'Santos']
O Palmeiras está na posição: 2º posição
```

maior e menor valores em tupla

Crie um programa vai mostrar 10 números. Depois disso, mostre a listagem de números gerados e também indique o menor e o maior valor que estão na tupla.

```
num = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
print(f'Os números gerados foram {num}')
print(f'O maior número sorteado foi {max(num)}')
print(f'O menor número sorteado foi {min(num)}')
```

```
Os números gerados foram (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
O maior número sorteado foi 10
O menor número sorteado foi 1
```

maior e menor valores em tuplas aleatórios

Crie um programa que vai gerar cinco números aleatórios e colocar em uma tupla. Depois disso, mostre a listagem de números gerados e também indique o menor e o maior valor que estão na tupla.

```
from random import randint
n = (randint(1, 10), randint(1, 10), randint(1, 10), randint(1, 10), randint(1, 10)) se você só coloca um randint, vai sortear apenas um número, coloque 5 randint para sortear 5 números
print(f'Eu sorteei os valores {n}')
print(f'O maior valor é {max(n)}')
print(f'O menor valor é {min(n)}')
```

```
Eu sorteei os valores (7, 6, 8, 9, 2)
O maior valor é 9
O menor valor é 2
```

análise de dados de uma tupla

Desenvolva um programa que leia quatro valores pelo teclado e guarde-os em uma tupla. No final, mostre:
A) Quantas vezes apareceu o valor 9.
B) Em que posição foi digitado o primeiro valor 3.
C) Quais foram os números pares.

```
núm = (int(input('Digite um número: ')),
int(input('Digite outro número: ')),
int(input('Digite mais um número: ')),
int(input('Digite o último número: ')))
print(f'Você digitou os valores {núm}')
print(f'O valor 9 apareceu {núm.count(9)} vezes')
if 3 in núm:
    print(f'O valor 3 apareceu na posição {núm.index(3) + 1}')
else:
    print(f'O valor 3 não apareceu em nenhuma posição')
for n in núm:
    if n % 2 == 0:
        print(f'Os números pares digitados foram {n}')
```

```
Digite um número: 1
Digite outro número: 2
Digite mais um número: 3
Digite o último número: 4
Você digitou os valores (1, 2, 3, 4)
O valor 9 apareceu 0 vezes
O valor 3 apareceu na posição 3
Os números pares digitados foram  2 4
```

lista de preços com tupla

Crie um programa que tenha uma tupla única com nomes de produtos e seus respectivos preços, na sequência. No final, mostre uma listagem de preços, organizando os dados em forma tabular.

```
produtos = ('Lápis', 1.75, 'Borracha', 2.00, 'Caderno', 15.90, 'Estojo', 25.00, 'Transferidor', 4.20,
            'Compasso', 9.99, 'Mochila', 120.32, 'Canetas', 22.30, 'Livro', 34.90)
print('='*50)
print('{:^50}'.format('LISTAGEM DE PREÇOS')) :^ significa centralizado
print('='*50)
```

for c in range(0, len(produtos), 2): vai do 0 até o último produto de 2 em 2
    print(f'{produtos[c]:.<40}', f' R$ {produtos[c+1]:>7.2f}') :< espaço para a direita, :> espaço para a esquerda, .2f para ter valor com 2 casas após a vírgula, :.< espaço para a direita com pontos
print("="*50)

```
==================================================
                LISTAGEM DE PREÇOS
==================================================
Lápis..................................  R$    1.75
Borracha...............................  R$    2.00
Caderno................................  R$   15.90
Estojo.................................  R$   25.00
Transferidor...........................  R$    4.20
Compasso...............................  R$    9.99
Mochila................................  R$  120.32
Canetas................................  R$   22.30
Livro..................................  R$   34.90
==================================================
```

contando vogais em tupla

Crie um programa que tenha uma tupla com várias palavras (não usar acentos). Depois disso, você deve mostrar, para cada palavra, quais são as suas vogais.

```
palavras = ('aprender', 'programar', 'linguagem', 'python', 'curso', 'gratis', 'estudar', 'praticar', 'trabalhar', 'mercado', 'programador', 'futuro')
for p in palavras:
    print(f'\n Na palavra {p.upper()} temos ', end=' ')
    for letra in p:
        if letra.lower() in 'aeiou':
            print(letra, end=' ')
```

```
Na palavra APRENDER temos  a e e
Na palavra PROGRAMAR temos  o a a
Na palavra LINGUAGEM temos  i u a e
Na palavra PYTHON temos  o
Na palavra CURSO temos  u o
Na palavra GRATIS temos  a i
Na palavra ESTUDAR temos  e u a
Na palavra PRATICAR temos  a i a
Na palavra TRABALHAR temos  a a a
Na palavra MERCADO temos  e a o
Na palavra PROGRAMADOR temos  o a a o
Na palavra FUTURO temos  u u o
```

listas

para adicionar um valor no final da lista, exemplo:
lanche.append('cookie')

para adicionar um valor em uma posição específica, exemplo:
lanche.insert(0, 'hotdog')

para eliminar um valor pela chave:
del lanche[3]

ou:
lanche.pop(3)

para eliminar o último valor:
lanche.pop()

para eliminar um valor pelo nome:
lanche.remove('pizza')

para colocar um valor na ordem crescente:
valores = [8, 7, 6, 5, 4, 3, 2, 1]
valores.sort()

para colocar um valor na ordem decrescente:
valores = [8, 7, 6, 5, 4, 3, 2, 1]
valores.sort(reverse=True)

para pegar o total:
len(valor)

mandar algo receber uma cópia de outra lista: (é necessário fazer uma cópia, porque se você for mudar um valor, vai mudar nas duas li)
b = a[:]
b[2] = 8

maior e menor valores na lista
Faça um programa que leia 5 valores numéricos e guarde-os em uma lista. No final, mostre qual foi o maior e o menor valor digitado e as suas respectivas posições na lista.

```
maior = 0 maior número
menor = 0 menor número
listanum = [] lista criada
for c in range(0, 5): vai ler 5 valores
    listanum.append(int(input(f'Digite um valor na posição {c}: '))) vai pedir um valor
    if c == 0: se o valor for o primeiro valor
        maior = menor = listanum[c] o maior, o menor e a lista vão ter o mesmo valor
    else: se não
        if listanum[c] > maior: se os valores da lista serem maiores que o maior número
            maior = listanum[c] o maior valor se torna os valores da lista
        if listanum[c] < menor: se os valores da lista serem menores que o maior número
            menor = listanum[c] o menor valor se torna os valores da lista
print('-=' * 30) linha decorativa
print(f'Os valores digitados foram {listanum}') mostrando os valores digitados
print(f'O maior valor digitado foi {maior} na posição ', end='') mostrando a posição do maior valor
for i, v in enumerate(listanum): vai mostrar a posição do valor v
    if v == maior: se o valor v for igual ao maior
        print(v) vai mostrar o valor v
        print(f'O menor valor digitado foi {menor} na posição ', end='') mostrando o menor valor
for i, v in enumerate(listanum): vai mostrar a posição do valor v
    if v == menor: se o valor v for igual ao menor
        print(v) vai mostrar o valor v
```

```
Digite um valor na posição 0: 0
Digite um valor na posição 1: 1
Digite um valor na posição 2: 2
Digite um valor na posição 3: 3
Digite um valor na posição 4: 4
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
Os valores digitados foram [0, 1, 2, 3, 4]
O maior valor digitado foi 4 na posição 4
O menor valor digitado foi 0 na posição 0
```

valores únicos em uma lista

Crie um programa onde o usuário possa digitar vários valores numéricos e cadastre-os em uma lista. Caso o número já exista lá dentro, ele não será adicionado. No final, serão exibidos todos os valores únicos digitados, em ordem crescente.

```
valores = [] a lista de valores
while True: enquanto for verdadeiro
    num = int(input('Digite um valor: ')) vai pedir um valor
    if num in valores: se esse valor estiver nos valores
        print('Valor duplicado! Não será adicionado.') é um valor duplicado e não pode ser adicionado
    else: se não
        valores.append(num) o valor vai ser adicionado nos valores
        print('Valor adicionado com sucesso!') mostrando na tela que foi possível adicionar
    resposta = str(input('Deseja continuar? [S/N]: ')).strip().upper()[0] deseja continuar?
    while resposta not in 'SN': se a resposta não estiver entre sim ou não
        resposta = str(input('Tente novamente. Deseja continuar? [S/N]: ')).strip().upper()[0] não é possível continuar
    if resposta == 'N': se a resposta for não
        break vai parar
print('=' * 30) linha decorativa
valores.sort() vai mostrar os valores na ordem
print(f'Você digitou os valores {valores}') os valores digitados
```

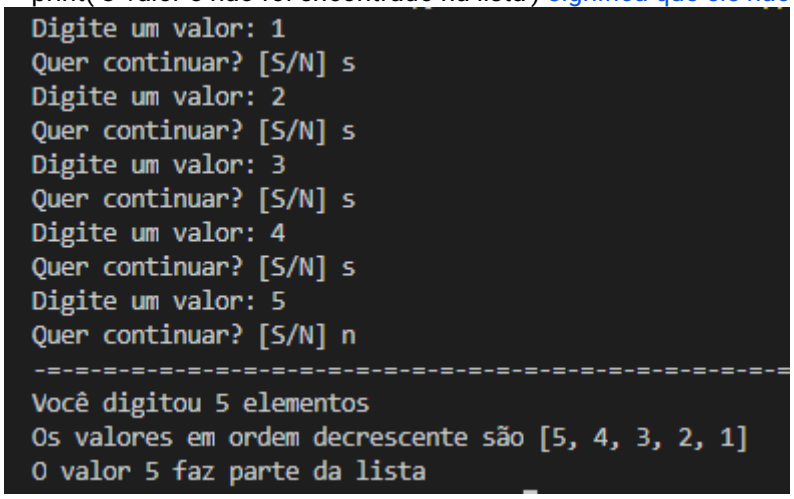
```
Digite um valor: 2
Valor adicionado com sucesso!
Deseja continuar? [S/N]: s
Digite um valor: 3
Valor adicionado com sucesso!
Deseja continuar? [S/N]: s
Digite um valor: 3
Valor duplicado! Não será adicionado.
Deseja continuar? [S/N]: n
==============================
Você digitou os valores [2, 3]
```

extraindo dados de uma lista

Crie um programa que vai ler vários números e colocar em uma lista. Depois disso, mostre: A) Quantos números foram digitados. B) A lista de valores, ordenada de forma decrescente. C) Se o valor 5 foi digitado e está ou não na lista.

```
valores = [] os valores vão ficar nessa lista
while True: loop infinito
    valores.append(int(input('Digite um valor: '))) vai pedir para digitar um número
```

```
    resp = str(input('Quer continuar? [S/N] ')) vai perguntar se quer continuar
    if resp in 'Nn': se a resposta for não
        break vai parar
print('-=' * 30) lista decorativa
print(f'Você digitou {len(valores)} elementos') o total dos valores
valores.sort(reverse=True) vai colocar na ordem decrescente
print(f'Os valores em ordem decrescente são {valores}') mostrando os valores na ordem decrescente
if 5 in valores: se o 5 tiver nos valores
    print('O valor 5 faz parte da lista') significa que ele faz parte da lista
else: se o 5 não tiver nos valores
    print('O valor 5 não foi encontrado na lista') significa que ele não faz parte da lista
```



dividindo valores em várias listas

Crie um programa que vai ler vários números e colocar em uma lista. Depois disso, crie duas listas extras que vão conter apenas os valores pares e os valores ímpares digitados, respectivamente. Ao final, mostre o conteúdo das três listas geradas.

```
num = [] crie uma lista para receber todos os números
pares = [] crie uma lista para receber os números pares
ímpares = [] crie uma lista para receber os números ímpares
while True: loop infinito
    num.append(int(input('Digite um número: '))) vai pedir para digitar um número, esse número vai ser guardado na lista num
    resp = str(input('Quer continuar? [S/N] ')).upper().strip()[0] vai perguntar se quer continuar
    if resp not in 'SN': se a resposta não for entre sim ou não
        print('Resposta inválida. Quer continuar? [S/N] ').upper().strip()[0] vai dizer que a resposta é inválida e vai perguntar se quer continuar
    if resp == 'N': se a resposta for não
        break o programa vai parar
for i, v in enumerate(num): vai pegar a posição
    if v % 2 == 0: se for par
        pares.append(v) vai adicionar na lista dos pares
for i, v in enumerate(num): vai pegar a posição
    if v % 2 == 1: se for ímpar
        ímpares.append(v) vai adicionar na lista dos ímpares
print(f'A lista completa é {num}') a lista completa
print(f'A lista de pares é {pares}') a lista dos pares
print(f'A lista de ímpares é {ímpares}') a lista dos ímpares
```

```
Digite um número: 2
Quer continuar? [S/N] s
Digite um número: 3
Quer continuar? [S/N] s
Digite um número: 4
Quer continuar? [S/N] s
Digite um número: 5
Quer continuar? [S/N] s
Digite um número: 6
Quer continuar? [S/N] n
A lista completa é [2, 3, 4, 5, 6]
A lista de pares é [2, 4, 6]
A lista de ímpares é [3, 5]
```

validando expressões matemáticas

Crie um programa onde o usuário digite uma expressão qualquer que use parênteses. Seu aplicativo deverá analisar se a expressão passada está com os parênteses abertos e fechados na ordem correta.

```
expr = str(input('Digite a expressão: ')) pedindo uma expressão
pilha = [] a lista da pilha
for símb in expr: para cada símbolo na expressão
    if símb == '(': se o símbolo for de abrir
        pilha.append('(') vai adicionar esse símbolo na lista
    elif símb == ')': se o símbolo for de fechar temos 2 opções
        if len(pilha) > 0: se tiver mais de um '('
            pilha.pop() vai excluir o último item
        else: se não tiver mais de um símbolo de fechamento
            pilha.append(')') vai adicionar o símbolo de fechar
            break vai parar
if len(pilha) == 0: se o total da pilha estiver com as quantidades certas para cada símbolo
    print('Sua expressão está válida') a expressão é válida
else: se não
    print('Sua expressão está errada') a expressão está errada
```

```
Digite a expressão: (a+b)
Sua expressão está valida
```

```
Digite a expressão: (a+b))
Sua expressão está errada
```

listas - parte 2

Faça um programa que leia nome e peso de várias pessoas, guardando tudo em uma lista. No final, mostre:
A) Quantas pessoas foram cadastradas. B) Uma listagem com as pessoas mais pesadas. C) Uma listagem com as pessoas mais leves.

```
temp = [] lista temporários, guarda os dados temporários antes de ir para a lista principal
princ = [] lista principal
while True: loop infinito
    temp.append(str(input('Nome: '))) pedindo o nome
    temp.append(float(input('Peso: '))) pedindo o peso
    if len(princ) == 0: se não tem ninguém cadastrado
        mai = men = temp[1] o maior é = ao menor que é = ao peso dos dados temporários
    else: se não
        if temp[1] > mai: se o peso for maior que o maior
            mai = temp[1] o maior passa a ser o temp[1]
        if temp[1] < men: se o temp[1] é menor que o menor
```

```
            men = temp[1] o menor é igual ao temp[1]
        princ.append(temp[:]) vai adicionar na lista principal a cópia dos dados temporários
        temp.clear() vai limpar os dados temporários
        resp = str(input('Quer continuar? ')) perguntando se quer continuar
        if resp in 'Nn': se a resposta for não
            break vai parar
print('-=' * 30) linha decorativa
print(f'Ao todo você cadastrou {len(princ)} pessoas.') vai mostrar o total de cadastros
print(f'O peso maior foi {mai} KG. Peso de ',end='') vai mostrar o peso maior
for p in princ: para cada pessoa nos dados principais
    if p[1] == mai: se o peso for o maior
        print(f'[{p[0]}] ', end='') vai mostrar o dado deles
print() vai ser um espaço
print(f'O peso menor foi {men} KG. Peso de ', end='') vai mostrar o peso menor
for p in princ: para cada pessoa nos dados principais
    if p[1] == men: se o peso for o menor
        print(f'[{p[0]}] ', end='') vai mostrar o dado deles
print() vai ser um espaço
```

```
Nome: Fernanda
Peso: 80
Quer continuar? sim
Nome: Cláudia
Peso: 100
Quer continuar? sim
Nome: Raíssa
Peso: 80
Quer continuar? sim
Nome: Robinson
Peso: 80
Quer continuar? n
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
Ao todo você cadastrou 4 pessoas.
O peso maior foi 100.0KG. Peso de [Cláudia]
O peso menor foi 80.0 KG. Peso de [Fernanda ] [Raíssa] [Robinson]
```

lista com pares e ímpares

Crie um programa onde o usuário possa digitar sete valores numéricos e cadastre-os em uma lista única que mantenha separados os valores pares e ímpares. No final, mostre os valores pares e ímpares em ordem crescente.

```
núm = [[], []] uma lista com 2 listas dentro dela
valor = 0 os valores vão ser inseridos dentro dessa lista
for c in range(1, 8): vai contar de 1 até 7
    valor = int(input(f'Digite o valor {c}: ')) pedir para digitar o valor
    if valor % 2 == 0: se esse valor for par
        núm[0].append(valor) esse valor vai ser colocado na primeira lista
    else: se não
        núm[1].append(valor) se for ímpar, vai ser colocado na segunda lista
print('-=' * 30) linha decorativa
núm[0].sort() vai colocar os valores pares na ordem crescente
núm[1].sort() vai colocar os valores ímpares na ordem crescente
print(f'Os valores pares digitados foram {núm[0]}') mostrando os valores pares
print(f'Os valores ímpares digitados foram {núm[1]}') mostrando os valores ímpares
```

```
Digite o valor 1: 2
Digite o valor 2: 3
Digite o valor 3: 4
Digite o valor 4: 5
Digite o valor 5: 6
Digite o valor 6: 7
Digite o valor 7: 8
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
Os valores pares digitados foram [2, 4, 6, 8]
Os valores ímpares digitados foram [3, 5, 7]
```

matriz em python

Crie um programa que declare uma matriz de dimensão 3×3 e preencha com valores lidos pelo teclado. No final, mostre a matriz na tela, com a formatação correta.

matriz = [[0, 0, 0], [0, 0, 0], [0, 0, 0]] 3 listas
for l in range(0, 3): para as linhas vai de 0 a 3
    for c in range(0, 3): para as colunas vai de 0 a 3
        matriz[l][c] = int(input(f'Digite o valor para [{l}, {c}]: ')) para essas matrizes digite um valor
print('-=' * 30) linha decorativa
for l in range(0, 3): para as linhas vai de 0 a 3
    for c in range(0, 3): para as colunas vai de 0 a 3
        print(f'[{matriz[l][c]:^5}]', end='') vai mostrar a matriz, o :^5 centralizou, o end='' fez pular uma linha
    print() um espaço em branco para poder pular a linha

```
Digite o valor para [0, 0]: 091
Digite o valor para [0, 1]: 129
Digite o valor para [0, 2]: 375
Digite o valor para [1, 0]: 981
Digite o valor para [1, 1]: 378
Digite o valor para [1, 2]: 382
Digite o valor para [2, 0]: 281
Digite o valor para [2, 1]: 182
Digite o valor para [2, 2]: 382
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
[ 91  ][ 129 ][ 375 ]
[ 981 ][ 378 ][ 382 ]
[ 281 ][ 182 ][ 382 ]
```

mais sobre matriz em python

Aprimore o desafio anterior, mostrando no final:
A) A soma de todos os valores pares digitados.
B) A soma dos valores da terceira coluna.
C) O maior valor da segunda linha.

matriz = [[0, 0, 0], [0, 0, 0], [0, 0, 0]] definindo a matriz
somapar = li = maior = 0 a soma do par = a soma dos valores da terceira linha = maior valor da segunda coluna
for l in range(0, 3): linha
    for c in range(0, 3): coluna
        matriz[l][c] = int(input(f'Digite o valor para [{l}, {c}]: ')) pegando os valores
print('-=' * 30) separador
for l in range(0, 3): linha

```
    for c in range(0, 3): coluna
        print(f'[{matriz[l][c]:^5}]', end='') montando visualmente a matriz
        if matriz[l][c] % 2 == 0: se for divisível por 2 (aproveitou o for de 'l' e 'c')
            somapar += matriz[l][c] vai adicionar na variável somapar
    print() espaço
print('-=' * 30) separador
print(f'A soma dos valores pares é {somapar}') soma dos valores pares
for l in range(0, 3): linha
    col += matriz[l][2]  vai adicionar na variável linha
print(f'A soma dos valores da terceira linha é {li}') soma dos valores da terceira linha
for c in range(0, 3): coluna
    if c == 0: se for a primeira coluna
        mai = matriz[1][c] vai pegar a segunda linha de todas as colunas
    elif matriz[1][c] > mai: se o valor for maior que o maior
        mai = matriz[1][c] o maior é o valor localizado como maior de todos
print(f'O maior valor da segunda coluna é {mai}') o valor da segunda coluna
```

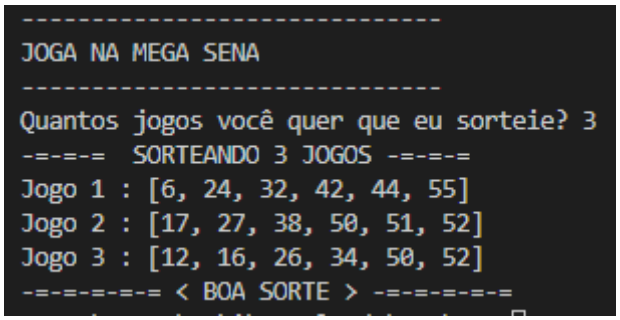
```
Digite o valor para [0, 0]: 1
Digite o valor para [0, 1]: 2
Digite o valor para [0, 2]: 3
Digite o valor para [1, 0]: 4
Digite o valor para [1, 1]: 5
Digite o valor para [1, 2]: 6
Digite o valor para [2, 0]: 7
Digite o valor para [2, 1]: 8
Digite o valor para [2, 2]: 9
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
[  1  ][  2  ][  3  ]
[  4  ][  5  ][  6  ]
[  7  ][  8  ][  9  ]
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
A soma dos valores pares é 20
A soma dos valores da terceira linha é 18
O maior valor da segunda coluna é 6
```

palpites para a mega sena

Faça um programa que ajude um jogador da MEGA SENA a criar palpites. O programa vai perguntar quantos jogos serão gerados e vai sortear 6 números entre 1 e 60 para cada jogo, cadastrando tudo em uma lista composta.

```
from random import randint vai importar o randint
from time import sleep
lista = [] lista
jogos = [] lista de jogos
print('-' * 30) linha decorativa
print('JOGA NA MEGA SENA') título
print('-' * 30) linha decorativa
quant = int(input('Quantos jogos você quer que eu sorteie? ')) perguntando a quantidade
tot = 1 total de jogos
while tot <= quant: enquanto o total for igual ou menor que a quantidade
    cont = 0 criou essa variável para ver a quantidade de números dentro de uma lista
    while True: loop infinito
        num = randint(1, 60) vai sortear um número de 1 a 60
        if num not in lista: vai verificar se não está na lista
            lista.append(num) se não estiver na lista vai adicionar esse número na lista
            cont += 1 e o contador vai adicionar mais 1
        if cont >= 6: se o contador for maior ou igual a 6
```

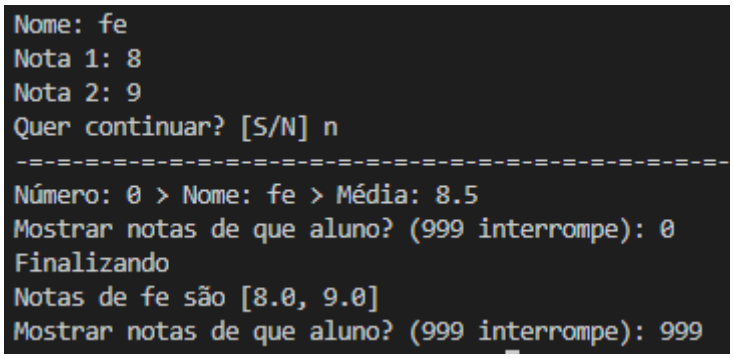
```
            break vai parar
    lista.sort() vai mostrar na ordem crescente
    jogos.append(lista[:]) vai adicionar uma cópia da lista
    lista.clear() vai apagar a lista
    tot += 1 vai adicionar no total
print('-=' * 3, f' SORTEANDO {quant} JOGOS', '-=' * 3)
for i, l in enumerate(jogos): para cada índice com a lista de jogos
    print(f'Jogo {i+1} : {l}') mostrando o jogo
    sleep(1) vai mostrar o jogo de 1 em 1 segundo
print('-=' * 5, '< BOA SORTE >', '-=' * 5) mensagem de boa sorte
```



boletim com listas compostas

Crie um programa que leia nome e duas notas de vários alunos e guarde tudo em uma lista composta. No final, mostre um boletim contendo a média de cada um e permita que o usuário possa mostrar as notas de cada aluno individualmente.

```
ficha = [] criando uma lista para guardar os valores
while True: loop infinito
    nome = str(input('Nome: ')) pegando o nome
    nota1 = float(input('Nota 1: ')) pegando a nota 1
    nota2 = float(input('Nota 2: ')) pegando a nota 2
    média = (nota1 + nota2) / 2 pegando a média
    resp = str(input('Quer continuar? [S/N] ')) perguntando se quer continuar
    ficha.append([nome, [nota1, nota2], média]) adicionando na ficha
    if resp in 'Nn': se a resposta for não
        break o programa vai acabar
print('-=' * 30) separador
for i, l in enumerate(ficha): enumerate
    print(f'Número: {i} > Nome: {l[0]} > Média: {l[2]}') vai mostrar o índice, e os números em suas
posições
while True: loop infinito
    mostrar = int(input('Mostrar notas de que aluno? (999 interrompe): ')) mostrando as notas
    if mostrar <= len(ficha) -1: o valor digitado precisa ser menor ou igual ao total -1
        print(f'Notas de {ficha[mostrar][0]} são {ficha[mostrar][1]}') vai mostrar os valores
    if mostrar == 999: se for 999
        break o programa vai parar
```



dicionários:
um dicionário fica entre chaves. exemplo: dados = {}

fazer print
dados = {'nome':'Pedro', 'idade': 25}
print(dados['nome'])

adicionar elementos
dados['sexo'] = 'M'

remover elementos
del dados['idade']

vai mostrar as chaves
.keys()

vai mostrar os valores
.values()

vai mostrar as chaves e os valores
.items()

copiar um conteúdo
.copy()

somar algo
sum()

dicionário em python
Faça um programa que leia nome e média de um aluno, guardando também a situação em um dicionário. No final, mostre o conteúdo da estrutura na tela. se a média for maior ou igual a 7 (aprovado), se a média for menor que 7 (recuperação), se a média for menor que 5 (reprovado)

```
aluno = {} dicionário do aluno
aluno['nome'] = str(input('Nome: ')) vai pedir o nome
aluno['média'] = float(input(f'A média de {aluno["nome"]} ')) vai pedir a média
if aluno['média'] >= 7: se a média for maior ou igual a 7
   aluno['situação'] = 'Aprovado' o aluno está aprovado
elif 5 <= aluno['média'] < 7: se a média for igual ou maior que 5 e menor que 7
   aluno['situação'] = 'Recuperação' o aluno está de recuperação
print('-=' * 30) separador
for k, v in aluno.items(): vai mostrar a chave e o valor
   print(f'{k} é igual a {v}') mostrando os valores
```

```
Nome: Fernanda
A média de Fernanda 6
-----------------------------
nome é igual a Fernanda
média é igual a 6.0
situação é igual a Recuperação
```

jogo de dados com python

Crie um programa onde 4 jogadores joguem um dado e tenham resultados aleatórios. Guarde esses resultados em um dicionário em Python. No final, coloque esse dicionário em ordem, sabendo que o vencedor tirou o maior número no dado.

```
from random import randint importando o randint
from time import sleep importando o time
from operator import itemgetter importando o itemgetter
jogo = {'jogador1':  randint(1, 6), sorteando valores
        'jogador2': randint(1, 6),
        'jogador3': randint(1, 6),
        'jogador4': randint(1, 6)} sorteando valores até o 4
ranking = [] lista do ranking
print('Valores sorteados:') mostrando os valores sorteados
for k, v in jogo.items(): fazendo o for dos dicionários
    print(f'{k} tirou o valor {v} no dado') mostrando o valor tirado
    sleep(1) pausa
ranking = sorted(jogo.items(), key=itemgetter(1), reverse=True) vai mostrar em ordem
decrescente os valores (o itemgetter precisa ser 1, pois vai mostrar o valor do randint, se for 0
vai mostrar a posição do jogador)
print('-=' * 30) separador
for i, v in enumerate(ranking): vai mostrar o ranking
    print(f'{i + 1}º lugar: {v[0]} com {v[1]}.') vai mostrar o índice, vai mostra o jogador e o valor que
o jogador tirou (foi o i + 1 para não contar a partir da posição 0 e sim do 1)
    sleep(1) pausa
```

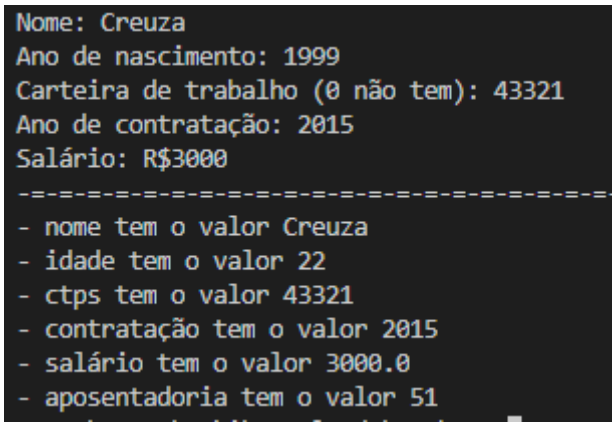
```
Valores sorteados:
jogador1 tirou o valor 4 no dado
jogador2 tirou o valor 4 no dado
jogador3 tirou o valor 3 no dado
jogador4 tirou o valor 6 no dado
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
1º lugar: jogador4 com 6.
2º lugar: jogador1 com 4.
3º lugar: jogador2 com 4.
4º lugar: jogador3 com 3.
```

cadastro de trabalhador em python

crie um programa que leia nome, ano de nascimento e carteira de trabalho e cadastre-o (com idade) em um dicionário. Se por acaso a CTPS for diferente de ZERO, o dicionário receberá também o ano de contratação e o salário. Calcule e acrescente, além da idade, com quantos anos a pessoa vai se aposentar.

```
from datetime import datetime importando a data e a hora
dados = {} dicionário dos dados
dados['nome'] = str(input('Nome: ')) pedindo o nome
nasc = int(input('Ano de nascimento: ')) pedindo o ano de nascimento
dados['idade'] = datetime.now().year - nasc cadastrando a idade
```

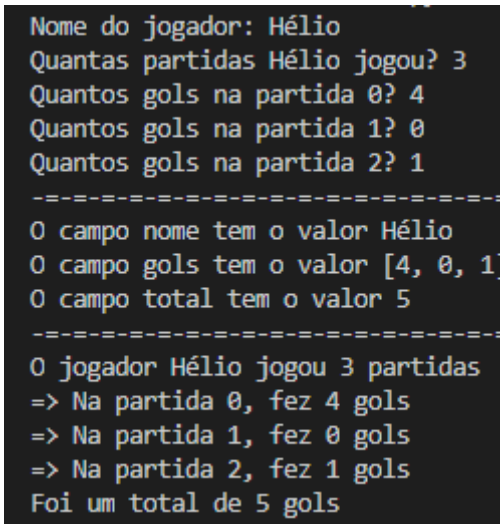
```
dados['ctps'] = int(input('Carteira de trabalho (0 não tem): ')) pedindo a carteira de trabalho
if dados['ctps'] != 0: se a carteira de trabalho não for igual a 0
    dados['contratação'] = int(input('Ano de contratação: ')) vai pedir o ano de contratação
    dados['salário'] = float(input('Salário: R$')) pedindo o salário
    dados['aposentadoria'] = dados['idade'] + (dados['contratação'] + 35) - datetime.now().year
calculou com quantos anos a pessoa vai se aposentar, os 35 anos variam de sexo para sexo,
fizemos um geral
    print('-=' * 30) separador
    for k, v in dados.items():
        print(f'- {k} tem o valor {v}') vão aparecer os valores
```



cadastro de jogador de futebol

Crie um programa que gerencie o aproveitamento de um jogador de futebol. O programa vai ler o nome do jogador e quantas partidas ele jogou. Depois vai ler a quantidade de gols feitos em cada partida. No final, tudo isso será guardado em um dicionário, incluindo o total de gols feitos durante o campeonato.

```
jogador = {} dicionário de jogador
partidas = [] lista de partidas
jogador['nome'] = str(input('Nome do jogador: ')) pedindo o nome do jogador
tot = int(input(f'Quantas partidas {jogador["nome"]} jogou? ')) perguntando quantas partidas o
jogador jogou
for c in range(0, tot):
    partidas.append(int(input(f'Quantos gols na partida {c}? '))) pedindo a quantidade de gols de
todas as partidas
jogador['gols'] = partidas[:] como é um dicionário e uma lista, é preciso colocar [:]
jogador['total'] = sum(partidas) o total é a soma das partidas
print('-=' * 30) separador
for k, v in jogador.items(): for para chave e valores
    print(f'O campo {k} tem o valor {v}') mostrando os valores do campo
print('-=' * 30) separador
print(f'O jogador {jogador["nome"]} jogou {len(jogador["gols"])} partidas') mostrando a
quantidade de gols que o jogador jogou
for i, v in enumerate(jogador['gols']): pegando a posição dos gols
    print(f'=> Na partida {i}, fez {v} gols') mostrando a quantidade de gols em cada partida
print(f'Foi um total de {jogador["total"]} gols') mostrando o total de gols
```



unindo dicionários e listas

Crie um programa que leia nome, sexo e idade de várias pessoas, guardando os dados de cada pessoa em um dicionário e todos os dicionários em uma lista. No final, mostre: A) Quantas pessoas foram cadastradas B) A média de idade C) Uma lista com as mulheres D) Uma lista de pessoas com idade acima da média.

```
pessoa = {} foi criado um dicionário
galera = [] foi criada uma lista
soma = média = idade = 0 a soma, a média e a idade tem valor inicial 0

while True: loop infinito
    pessoa.clear() vai limpar a cópia
    pessoa['nome'] = str(input('Nome: ')) vai pedir o nome

    while True: loop infinito
        pessoa['sexo'] = str(input('Sexo (M/F): ')).upper()[0] pedindo o sexo da pessoa
        if pessoa['sexo'] in 'MF': se aparecer M ou F
            break o programa vai parar
        print('ERRO! Por favor digite apenas M ou F.') se não for M ou F vai aparecer uma
mensagem de erro

    pessoa['idade'] = int(input('Idade: '))  pedindo a idade
    soma += pessoa['idade'] a soma é a soma + idade da pessoa
    galera.append(pessoa.copy()) vai mandar essas informações para a lista 'galera', por ser
uma lista é feito desse modo a transferência

    while True: loop infinito
        resp = str(input('Quer continuar? (S/N): ')).upper()[0] perguntando se quer continuar
        if resp in 'SN': se a resposta for S ou N
            break o programa vai parar
        print('ERRO! Responda apenas S ou N.') se a resposta não for S ou N vai aparecer uma
mensagem de erro

    if resp == 'N': se a resposta for N
        break o programa vai parar
print('-=' * 30) linha decorativa

print(f'Ao todo temos {len(galera)} pessoas cadastradas.') mostrando o total de pessoas
cadastradas
```

média = soma / len(galera) calculando a média
print(f'A média de idade é de {média:5.2f} anos.') mostrando a média

print(f'As mulheres cadastradas foram ', end='') mostrando as mulheres cadastradas
for p in galera: calculando o for
  if p['sexo'] in 'Ff': se o sexo for feminino
    print(f'{p["nome"]} ', end='') vai mostrar os nomes das mulheres cadastradas
print() colocou o end e o print para cada informação de cada pessoa ficar em uma linha

print(f'Lista de pessoas que estão com a nota acima da média: ', end='') mostrando lista de pessoas com nota acima da média
for p in galera: calculando o for
  if p['idade'] >= média: se a idade for maior que a média
    print(' ')
    for k, v in p.items(): vai calcular os valores da lista das pessoas
      print(f'{k} = {v}, ', end='') mostrando os valores
    print() colocou o end e o print para cada informação de cada pessoa ficar em uma linha

print('<<<ENCERRADO>>>')

```
Nome: Fernanda
Sexo (M/F): F
Idade: 28
Quer continuar? (S/N): S
Nome: Maria
Sexo (M/F): F
Idade: 21
Quer continuar? (S/N): S
Nome: José
Sexo (M/F): ç
ERRO! Por favor digite apenas M ou F.
Sexo (M/F): M
Idade: 23
Quer continuar? (S/N): K
ERRO! Responda apenas S ou N.
Quer continuar? (S/N): N
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Ao todo temos 3 pessoas cadastradas.
A média de idade é de 24.00 anos.
As mulheres cadastradas foram Fernanda Maria
Lista de pessoas que estão com a nota acima da média:
nome = Fernanda, sexo = F, idade = 28,
<<<ENCERRADO>>>
```

aprimorando os dicionários - não entendi

Aprimore o desafio 93 para que ele funcione com vários jogadores, incluindo um sistema de visualização de detalhes do aproveitamento de cada jogador.

time = [] lista de time
jogador = {} dicionário de jogador
partidas = [] lista de partidas
while True: loop infinito
  jogador.clear() vai limpar, porque esse loop vai ser executado mais de 1 vez
  jogador['nome'] = str(input('Nome do jogador: ')) pedindo o nome do jogador

```
    tot = int(input(f'Quantas partidas {jogador["nome"]} jogou? '))  pedindo a quantidade de partidas

    partidas.clear() vai limpar as partidas, porque esse loop vai ser executado mais de 1 vez
    for c in range(0, tot): fazendo um for
        partidas.append(int(input(f'Quantos gols na partida {c+1}? '))) perguntando a quantidade de gols
    jogador['gols'] = partidas[:]  passando as partidas para os gols
    jogador['total'] = sum(partidas)  calculando o total
    time.append(jogador.copy()) adicionando na lista time o dicionário jogador

    while True: loop infinito
        resp = str(input('Quer continuar? (S/N): ')).upper()[0] perguntando se quer continuar
        if resp in 'SN': se a resposta for S ou N
            break o programa vai encerrar
        print('ERRO! Responda apenas S ou N.') se a resposta não for S ou N vai aparecer uma mensagem de erro

    if resp == 'N': se a resposta for N
        break o programa vai parar

print('-=' * 30) separador
print('cod ', end='') vai mostrar o cod (ele não está em jogador.keys)
for i in jogador.keys(): vai mostrar os títulos para a tabela
    print(f'{i:<15}', end='') vai centralizar
print() o 'end' e o 'print()' para continuar na mesma linha e depois pular uma linha
print('-' * 40) separador

for k, v in enumerate(time): para keys e values não entendi porque usou esses valores e o enumerate do nada
    print(f'{k:>4} ', end='') mostrando as keys
    for d in v.values(): pegando os values
        print(f'{str(d):<15}', end='') mostrando o 'd' como string e centralizando não entendi porque virou uma string
    print() o 'end' e o 'print()' para continuar na mesma linha e depois pular uma linha
print('-=' * 40) separador

while True: loop infinito
    busca = int(input('Mostrar dados de que jogador? (999 para parar) ')) mostrar dados
    if busca == 999: se for 999
        break o programa vai parar
    if busca >= len(time): se for maior ou igual ao total de time não entendi como chegou nesse cálculo do len(time)
        print(f'ERRO! Não existe jogador com código {busca}!') vai aparecer um erro
    else: caso contrário
        print(f'- LEVANTAMENTO DO JOGADOR {time[busca]["nome"]}: ') vai aparecer o levantamento do jogador
        for i, g in enumerate(time[busca]['gols']): vai buscar os gols
            print(f' No jogo {i} fez {g} gols') mostrando os gols
    print('-=' * 40) separador
print('<<<VOLTE SEMPRE>>>') fim do programa
```

funções

exemplos de funções que usamos com frequência:
- print()
- len()
- input()

- int()
- float()

exemplo de função:
```
def mostrarLinha():
print('-------')
```

(criamos uma função que guarda como valor os traços)

para usarmos essa função podemos fazer assim:
```
mostrarLinha()
print('Esse é um exemplo')
```

outro exemplo de função:
```
def mensagem(msg):
 print('----')
 print(msg)
 print('----')

mensagem('SISTEMA DE ALUNOS')
```

(foi criada função que se chama 'mensagem', ela recebe o valor 'msg'
a mensagem recebeu o valor 'SISTEMA DE ALUNOS', e essa mensagem será substituída dentro da função)

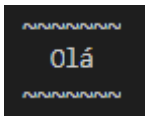
como passar vários parâmetros para uma função:
```
def contador(* num):
contador(1, 2, 3, 4)
contador(4, 5, 6, 7)
```

função que calcula a área
Faça um programa que tenha uma função chamada área(), que receba as dimensões de um terreno retangular (largura e comprimento) e mostre a área do terreno.

```
def área(larg, comp): função da largura e comprimento
    a = larg * comp cálculo da área
    print(f'A área de um terreno {larg}x{comp} é de {a}m².') mostrando a área do terreno

print('CONTROLE DE TERRENOS') controle de terrenos
print('-' * 20) separador
l = float(input('Largura (m): ')) largura
c = float(input('Comprimento (m): ')) comprimento
área(l, c) mostrando a área
```

um print especial
Faça um programa que tenha uma função chamada escreva(), que receba um texto qualquer como parâmetro e mostre uma mensagem com tamanho adaptável.

```
def escreva(msg): criando a função 'escreva' que recebe o valor 'msg'
    tam = len(msg) + 4 o tamanho recebe o total da mensagem + 4 (+ 4 para se ter 2 bordas na direita e na esquerda)
    print('~' * tam) '~' vai aparecer na quantidade de vezes que for o tamanho da mensagem
    print(f'  {msg}') colocou a mensagem, antes deu 2 espaços, para ficar centralizado
    print('~' * tam)  '~' vai aparecer na quantidade de vezes que for o tamanho da mensagem

escreva('Olá') o valor que vai aparecer
```



função de contador - não entendi
Faça um programa que tenha uma função chamada contador(), que receba três parâmetros: início, fim e passo. Seu programa tem que realizar três contagens através da função criada:
a) de 1 até 10, de 1 em 1
b) de 10 até 0, de 2 em 2
c) uma contagem personalizada

```
from time import sleep importando a biblioteca do tempo

def contador(i, f, p): não entendi
    if p < 0: não entendi
        p *= -1 não entendi
    if p == 0: não entendi
        p = 1 não entendi

    print('-=' * 20) separador
    print(f'Contagem de {i} até {f} de {p} em {p}') mostrando a contagem
    sleep(2.5) tempo de pausa

    if i < f:  não entendi
        cont = i não entendi
        while cont <= f: não entendi
            print(f'{cont} ', end='', flush=True) mostrando o valor, o 'flush=True' serve para quando o computador dá um bug ele fica demorando para carregar, com ele não vai demorar para carregar
            sleep(0.5) tempo de pausa
            cont += p não entendi
        print('FIM!') fim
    else: se não
        cont = i não entendi
        while cont >= f: não entendi
            print(f'{cont} ', end='', flush=True) mostrando o resultado
            sleep(0.5) pausa
            cont -= p
        print('FIM!') fim

contador(1, 10, 1) cálculo que vai se realizar
contador(10, 0, 2) cálculo que vai se realizar
print('-=' * 20) separador
print('Agora é a sua vez de personalizar a contagem.') personalizando a contagem
ini = int(input('Início: ')) início
fim = int(input('Fim: ')) fim
pas = int(input('Passo: ')) passo
contador(ini, fim, pas) passando para o contador, para entrar na função
```

```
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
Contagem de 1 até 10 de 1 em 1
1 2 3 4 5 6 7 8 9 10 FIM!
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
Contagem de 10 até 0 de 2 em 2
10 8 6 4 2 0 FIM!
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
Agora é a sua vez de personalizar a contagem.
Início: 10
Fim: 0
Passo: -2
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
Contagem de 10 até 0 de 2 em 2
10 8 6 4 2 0 FIM!
```

função que descobre o maior

Faça um programa que tenha uma função chamada maior(), que receba vários parâmetros com valores inteiros. Seu programa tem que analisar todos os valores e dizer qual deles é o maior.

```
def maior(* num): como são vários valores, acrescentou o *
    if len(num) > 0: se o total de num for maior que 0
        print(f'O maior de {num} é o {max(num)}.') vai mostrar o maior valor
    else: se o valor for menor que 0, significa que não tem nenhum valor
        print('Não foi informado nenhum valor!') mostrando que não foi informado nenhum valor
```

a seguir são os valores que serão passados para a função:

```
maior(9, 2, 11, 5, 3)
maior(4, 7, 1, 2)
maior(1, 2, 3, 7, 3, 9, 5, 12)
maior(6)
maior(1, 2)
maior()
```

```
O maior de (9, 2, 11, 5, 3) é 11
O maior de (4, 7, 1, 2) é 7
O maior de (1, 2, 3, 7, 3, 9, 5, 12) é 12
O maior de (1, 2) é 2
Não foi informado nenhum valor!
```

funções para sortear e somar

Faça um programa que tenha uma lista chamada números e duas funções chamadas sorteia() e somaPar(). A primeira função vai sortear 5 números de 1 a 10 e vai colocá-los dentro da lista e a segunda função vai mostrar a soma entre todos os valores pares sorteados pela função anterior.

```
from random import randint vai importar o random
from time import sleep vai importar o sleep

def sorteia(lista): função sorteia
    print(f'Sorteando 5 valores da lista: ', end='') sorteando 5 valores
    for cont in range(0, 5): vai pegar de 1 a 5
        n = randint(1, 10) dentro dos números de 1 a 10
        lista.append(n) vai adicionar na lista
        print(f'{n} ', end='', flush=True) mostrou os valores sorteados
        sleep(0.3) vai mostrar os valores de 0.3 em 0.3
```

```
    print('PRONTO!') finalizando

def somaPar(lista): função somaPar
    soma = 0 a soma é 0
    for valor in lista: para cada valor na lista
        if valor % 2 == 0: se o valor for par
            soma += valor vai somar esses valores
    print(f'Somando os valores pares da {lista}, temos {soma}') mostrando os valores

números = [] lista dos números
sorteia(números) função sorteia
somaPar(números) função somaPar
```

```
Sorteando 5 valores da lista: 10 8 6 7 2 PRONTO!
Somando os valores pares da [10, 8, 6, 7, 2], temos 26
```

funções - parte 2

para achar informações do input:
help(input)

docstrings
embaixo da função abra aspas 3 vezes + enter
dica: as docstrings só funcionam no pycharm

```
def contador(i, f, p):
    """
        -> Faz uma contagem e mostra na tela.
        :param i: início da contagem
        :param f: fim da contagem
        :param p: passo da contagem
        :return: sem retorno
    """
    c = i
    while c <= f:
        print(f'{c}', end='.. ')
        c += p
    print('FIM!')
```

parâmetros opcionais:
são parâmetros opcionais, caso a pessoa não preencha um valor, o programa não vai parar de funcionar

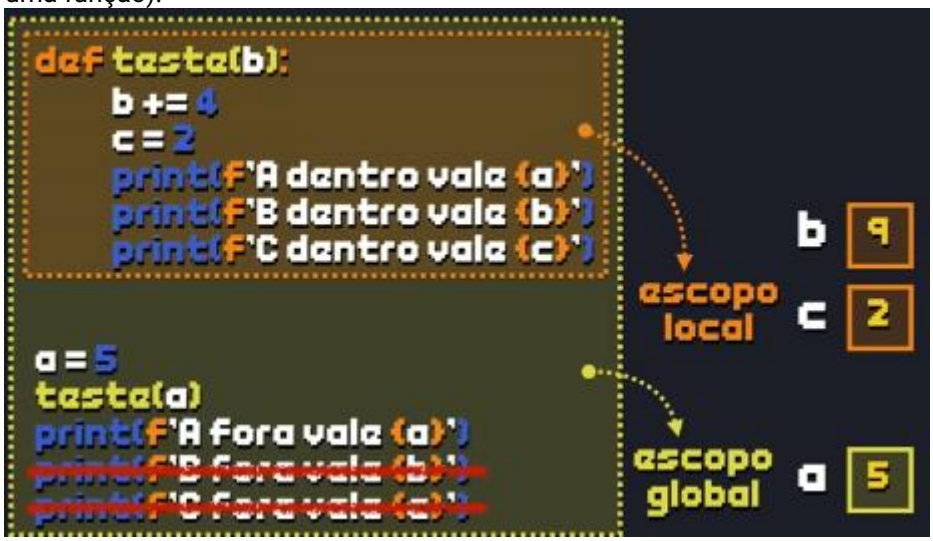
no exemplo abaixo, ‘a’, ‘b’, ‘c’ recebem o valor ‘0’, ou seja, se o usuário não digitar nenhum número, o programa ainda funcionaria

```
def somar(a=0, b=0, c=0):
    s = a + b + c
    print(f'A soma vale {s}')



somar( 3, 2, 5 )
somar( 8, 4 )
somar()
```

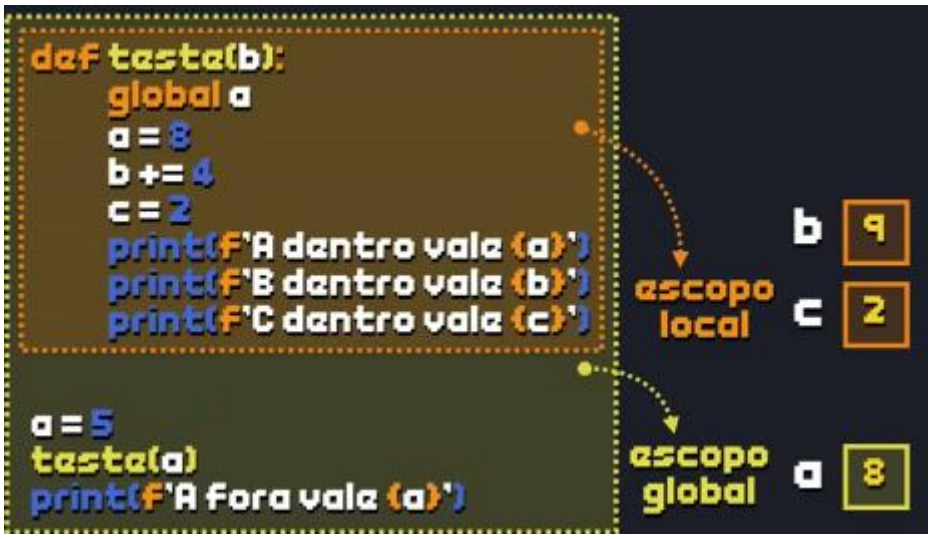
escopo de variáveis

escopo local: só funciona dentro de uma função. (no exemplo abaixo o ‘b’ e o ‘c’ só funcionam dentro da primeira função, não funcionam fora dela).

escopo global: funciona em mais de uma função. (no exemplo abaixo o ‘a’ funciona em mais de uma função).



se você quer que algo seja global, em vez de escrever de novo e criar uma nova variável, escreva: global + variável



retorno de valores

usando o return podemos retornar valores, são muito úteis quando você quer personalizar resultados

```
def somar(a=0, b=0, c=0):
    s = a + b + c
    return s


r1 = somar( 3, 2, 5 )
r2 = somar( 1, 7 )
r3 = somar( 4 )
print(f'Meus cálculos deram {r1}, {r2} e {r3}.')
```

funções para votação

Crie um programa que tenha uma função chamada voto() que vai receber como parâmetro o ano de nascimento de uma pessoa, retornando um valor literal indicando se uma pessoa tem voto NEGADO, OPCIONAL e OBRIGATÓRIO nas eleições.

```
def voto(idade): início da função
    from datetime import date importando a data atual

    atual = date.today().year cálculo para pegar a data atual
    idade = atual - nasc calculando a idade

    if idade < 16: se a idade for maior que 16
        return f'Com {idade} anos. Não vota.' a pessoa não vai votar
    elif 16 <= idade < 18 or idade > 65: se a idade for igual ou maior que 16 e menor que 18 ou a idade maior que 65 anos
        return f'Com {idade} anos. É opcional.' a pessoa pode votar opcionalmente
    else: se não
        return f'Com {idade} anos. Voto obrigatório.' a pessoa vai obrigatoriamente votar

nasc = int(input('Em que ano você nasceu? ')) perguntando o ano que a pessoa nasceu
print(voto(nasc)) ligando com a função
```

```
Em que ano você nasceu? 2003
Com 18 anos. Voto obrigatório.
```

função para fatorial

Crie um programa que tenha uma função fatorial() que receba dois parâmetros: o primeiro que indique o número a calcular e outro chamado show, que será um valor lógico (opcional) indicando se será mostrado ou não na tela o processo de cálculo do fatorial.

def fatorial(n, show=False): 1- função do fatorial. 2- o show é o parâmetro opcional e ele começa com false porque não quer mostrar o cálculo.

```
    """ usando docstrings
    -> Calcule o fatorial de um número.
    :param n: O número a ser calculado.
    :param show: (opcional) Mostrar ou não a conta.
    :return: O valor do Fatorial de um número n.
    """ fim da docstring

    f = 1 criou uma variável f, que começa com 1
    for c in range(n, 0, -1): vai começar do número até o 0 e vai de maneira decrescente
        if show: se o show
            print(c, end='') vai mostrar o número
            if c > 1: se for maior que 1
                print(' x ', end='') vai mostrar o 'x'
            else: se não
                print(' = ', end='') se for igual a 1, significa que é o último número
        f *= c vai adicionar o fatorial no c
    return f vai mostrar o valor de f


print(fatorial(5, show=True)) vai mostrar o fatorial do 5, o show=True vai mostrar o valor mostrado na tela
help(fatorial) vai mostrar o conteúdo das docstrings
```

```
5 x 4 x 3 x 2 x 1 = 120
Help on function fatorial in module __main__:

fatorial(n, show=False)
    -> Calcule o fatorial de um número.
    :param n: O número a ser calculado.
    :param show: (opcional) Mostrar ou não a conta.
    :return: O valor do Fatorial de um número n.
```

ficha do jogador

Faça um programa que tenha uma função chamada ficha(), que receba dois parâmetros opcionais: o nome de um jogador e quantos gols ele marcou. O programa deverá ser capaz de mostrar a ficha do jogador, mesmo que algum dado não tenha sido informado corretamente.

def analise(nome, gols): função criada

```
  if not nome: se não tiver nome
      nome = '<desconhecido>' o nome vai ser desconhecido
```

```
    if not gols or not gols.isnumeric(): se não tiver gols ou gols numéricos
        gols = 0 o gol vai ser igual a 0
    print('-' * 50) separador
    print(f'O jogador {nome} fez {gols} gol(s) no campeonato.') mostrando o valor na tela

j = str(input('Nome do jogador: ')).strip() pedindo o nome do jogador
g = str(input('Número de gols: ')).strip() pedindo o número de gols, precisa ser string para ser possível digitar um valor vazio
analise(j, g) passando os valores para a função
```

```
Nome do jogador:
Número de gols:
--------------------------------------------------
O jogador <desconhecido> fez 0 gol(s) no campeonato.
```

```
Nome do jogador: Fernanda
Número de gols: 8
--------------------------------------------------
O jogador Fernanda fez 8 gol(s) no campeonato.
```

validando a entrada de dados em python

Crie um programa que tenha a função leiaInt(), que vai funcionar de forma semelhante 'a função input() do Python, só que fazendo a validação para aceitar apenas um valor numérico. Ex: n = leiaInt('Digite um n: ')

```
def leiaInt(num): iniciando a função
    while True: loop infinito
        valor = str(input((num))) o valor é 'n'
        if valor.isnumeric(): se esse valor for numérico
            return valor vai retornar o valor de 'n'
        else: se não
            print('\033[31mERRO! Digite um número inteiro válido\033[m') vai retornar uma mensagem de erro com códigos que deixam o texto vermelho
        if valor.isnumeric(): se o valor for numérico
            break o programa vai parar

n = leiaInt('Digite um número: ') pedindo o número
print(f'Você digitou o número {n}') mostrando o número digitado
```

```
Digite um número:
ERRO! Digite um número inteiro válido
Digite um número: 5
Você digitou o número 5
```

analisando e gerando dicionários

faça um programa que tenha uma função notas() que pode receber várias notas de alunos e vai retornar um dicionário com as seguintes informações:
- quantidade de notas
- a maior nota
- a menor nota
- a média da turma
- a situação (opcional)

```
def notas(*notas, sit=True): início da função
    adSituacao = dict() criando um dicionário para guardar as informações

    total = len(notas) calculando o total de notas
    adSituacao['Total'] = total adicionando no dicionário

    maior = max(notas) calculando a nota máxima
    adSituacao['Maior'] = maior adicionando no dicionário

    menor = min(notas) calculando a nota mínima
    adSituacao['Menor'] = menor adicionando no dicionário

    soma = sum(notas) calculando a soma das notas
    media = soma / len(notas) calculando a média
    adSituacao['Média'] = f'{media :.1f}' adicionando no dicionário

    if sit: se a situação
        if media >= 7: se a média for maior ou igual a 7
            adSituacao['Situação'] = 'Aprovado' aprovado
        elif media >= 5: se a média é maior ou igual a 5
            adSituacao['Situação'] = 'Razoável' razoável
        else: se não
            adSituacao['Situação'] = 'Reprovado' reprovado

    return adSituacao vai retornar os valores da situação

# Programa principal...
resposta = notas(5.6, 10, 8.7, 3.3, sit=True) notas selecionadas para esse programa
print(resposta) mostrando as notas
```

```
{'Total': 4, 'Maior': 10, 'Menor': 3.3, 'Média': '6.9', 'Situação': 'Razoável'}
```

interactive helping system in python

Faça um mini-sistema que utilize o Interactive Help do Python. O usuário vai digitar o comando e o manual vai aparecer. Quando o usuário digitar a palavra ‘FIM’, o programa se encerrará. Importante: use cores.

```
from time import sleep importanto o sleep

a seguir são as cores definidas
c = ('\033[m', # 0 - sem cores
    '\033[0;30;41m', # 1 - vermelho
    '\033[0;30;42m', # 2 - verde
    '\033[0;30;43m', # 3 - amarelo
    '\033[0;30;44m', # 4 - azul
    '\033[0;30;45m', # 5 - roxo
    '\033[7;30m' # 6 - branco
    );

def ajuda(com): função ajuda
    título(f'Acessando o manual do comando \'{com}\'', 4) chamando a função título, vai mostrar essa mensagem, no local do ‘com’ vai aparecer o nome do método que você quer ver o manual, o 4 apareceu com a cor azul
    print(c[5], end='') vai mostrar a cor roxa
    help(com) com esse comando vai mostrar o manual do que você digitou
    print(c[5], end='') vai mostrar a cor roxa
```

sleep(2) vai pausar por 2 segundos

def título(msg, cor=0): função título, recebe uma mensagem e uma cor
tam = len(msg) + 4 vai ser criado as cobrinhas, o tamanho das cobrinhas vai ser igual ao tamanho da mensagem + 4, para sobrar cobrinhas
print(c[cor], end='') vai mostrar a cor sem pular linha
print('~' * tam) mostrando a cobrinha
print(f' {msg}') vai mostrar a mensagem, vai ficar pro lado por dois espaços, para sobrar 2 cobrinhas em cada lado
print('~' * tam) mostrando a cobrinha
print(c[0], end='') vai mostrar a cor na posição 0, para limpar a cor e sem pular linha
sleep(1)

# Programa principal
comando = '' uma variável chamada 'comando', ela começa vazia
while True: loop infinito
título('SISTEMA DE AJUDA PyHELP', 2) vai mostrar o título, com o número 2, que é o verde
comando = str(input("Função ou biblioteca > ")) vai pedir para você digitar algo
if comando.upper() == 'FIM': se o comando for 'fim'
break vai parar
else: se for outro comando
ajuda(comando) vai pedir a ajuda do comando
título('ATÉ LOGO', 1) vai dizer até logo com o número 1, que é a cor vermelha

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~
  SISTEMA DE AJUDA PyHELP
~~~~~~~~~~~~~~~~~~~~~~~~~~~
Função ou biblioteca > print
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Acessando o manual do comando 'print'
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.

~~~~~~~~~~~~~~~~~~~~~~~~~~~
  SISTEMA DE AJUDA PyHELP
~~~~~~~~~~~~~~~~~~~~~~~~~~~
Função ou biblioteca > fim
~~~~~~~~~~~~
  ATÉ LOGO
~~~~~~~~~~~~
```

módulos e pacotes

podemos separar o programa principal de uma função, podemos deixar o programa principal em um arquivo e a função em outro. para fazer a ligação entre eles:

```
from úteis import fatorial, dobro
```

úteis: exemplo de nome de arquivo
fatorial e dobro: exemplo do nome das funções a serem importadas

ou fazer:

```
from úteis
```

e adicionar o fatorial e o dobro nos locais que precisam ser adicionados, exemplo:

```
print(f'O dobro de {num} é {uteis.dobro(num)}')
```

para criar um pacote > new > python package > dê um nome para o pacote

exercitando módulos em python

Crie um módulo chamado moeda.py que tenha as funções incorporadas aumentar(), diminuir(), dobro() e metade(). Faça também um programa que importe esse módulo e use algumas dessas funções.

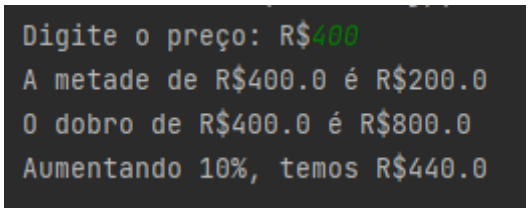
vai na raiz > new > directory > dê o nome de 'ex117'
clique com o botão direito em 'ex117' > new > python file > dê o nome de 'moeda.py'
clique com o botão direito em 'ex117' > new > python file > dê o nome de 'teste.py'

no moeda.py:

```
def aumentar(preço, taxa): função aumentar
    res = preço + (preço * taxa/100) cálculo
    return res vai retornar o cálculo

def diminuir(preço, taxa): função diminuir
    res = preço - (preço * taxa/100)
    return res vai retornar o cálculo

def dobro(preço): função dobro
    res = preço * 2 cálculo
    return res vai retornar o cálculo

def metade(preço): função metade
    res = preço / 2 cálculo
    return res vai retornar o cálculo
```

no teste.py:

```
from ex116 import moeda importando a moeda

p = float(input('Digite o preço: R$')) pedindo o preço
print(f'A metade de R${p} é R${moeda.metade(p)}') mostrando a metade
print(f'O dobro de R${p} é R${moeda.dobro(p)}') mostrando o dobro
print(f'Aumentando 10%, temos R${moeda.aumentar(p, 10)}') mostrando o aumento
```



formatando moedas em python

Adapte o código do desafio acima, criando uma função adicional chamada moeda() que consiga mostrar os números como um valor monetário formatado.

faça ctrl + v e ctrl + c no diretório do 'ex116.py'
vai duplicar esse exercício e dê o nome de 'ex117.py'

def aumentar(preço=0, taxa=0): função aumentar, ela recebe parâmetros, caso não tenha um valor para eles, eles irão receber 0
    res = preço + (preço * taxa/100) cálculo
    return res vai retornar o cálculo

def diminuir(preço=0, taxa=0): função diminuir, ela recebe parâmetros, caso não tenha um valor para eles, eles irão receber 0
    res = preço - (preço * taxa/100) cálculo
    return res vai retornar o cálculo

def dobro(preço=0): função preço, ela recebe um parâmetro, caso não tenha um valor para ela, ela vai receber 0
    res = preço * 2 cálculo
    return res vai retornar o cálculo

def metade(preço=0): função metade, ela recebe um parâmetro, caso não tenha um valor para ela, ela vai receber 0
    res = preço / 2 cálculo
    return res vai retornar o cálculo

def moeda(preço = 0, moeda='R$'): função moeda, ela recebe parâmetros, caso não tenha um valor para eles, eles irão receber 0 e R$
    return f'{moeda}{preço:>.2f}'.replace('.', ',') vai retornar o R$ e o preço com 2 casas depois da vírgula, os pontos vão ser substituídos por vírgulas

no teste.py:

from ex117 import moeda importando a moeda

p = float(input('Digite o preço: R$')) vai pedir o preço
print(f'A metade de R${moeda.moeda(p)} é R${moeda.moeda(moeda.metade(p))}') mostrando a metade
print(f'O dobro de R${moeda.moeda(p)} é R${moeda.moeda(moeda.dobro(p))}') mostrando o dobro
print(f'Aumentando 10%, temos R${moeda.moeda(moeda.aumentar(p, 10))}') mostrando o aumento

```
Digite o preço: R$800
A metade de R$R$800,00 é R$R$400,00
O dobro de R$R$800,00 é R$R$1600,00
Aumentando 10%, temos R$R$880,00
```

formatando moedas em python

Modifique as funções que foram criadas no desafio antes do anterior para que elas aceitem um parâmetro a mais, informando se o valor retornado por elas vai ser ou não formatado pela função moeda(), desenvolvida no desafio anterior.

faça ctrl + v e ctrl + c no diretório do 'ex117.py'
vai duplicar esse exercício e dê o nome de 'ex119.py'

no moeda.py:

```
def aumentar(preço=0, taxa=0, formato=False): foi adicionado o parâmetro formato que é falso
    res = preço + (preço * taxa/100) cálculo
    return res if formato is False else moeda(res) vai retornar o cálculo se for falso, se não vai passar pela função da moeda

def diminuir(preço=0, taxa=0, formato=False): foi adicionado o parâmetro formato que é falso
    res = preço - (preço * taxa/100) cálculo
    return res if formato is False else moeda(res) vai retornar o cálculo se for falso, se não vai passar pela função da moeda

def dobro(preço=0, formato=False): foi adicionado o parâmetro formato que é falso
    res = preço * 2 cálculo
    return res if not formato else moeda(res) vai retornar o cálculo se for falso, se não vai passar pela função da moeda

def metade(preço=0, formato=False): foi adicionado o parâmetro formato que é falso
    res = preço / 2 cálculo
    return res if not formato else moeda(res) vai retornar o cálculo se for falso, se não vai passar pela função da moeda

def moeda(preço = 0, moeda='R$'): função moeda, ela recebe parâmetros, caso não tenha um valor para eles, eles irão receber 0 e R$
    return f'{moeda}{preço:>.2f}'.replace('.', ',') vai retornar o R$ e o preço com 2 casas depois da vírgula, os pontos vão ser substituídos por vírgulas
```

no teste.py:

```
from ex119 import moeda importando a moeda

p = float(input('Digite o preço: R$')) pedindo o preço
print(f'A metade de R${moeda.moeda(p)} é R${moeda.metade(p, True)}') o True foi adicionado
print(f'O dobro de R${moeda.moeda(p)} é R${moeda.dobro(p, True)}') o True foi adicionado
print(f'Aumentando 10%, temos R${moeda.aumentar(p, 10, True)}') o True foi adicionado
print(f'Reduzindo 13%, temos {moeda.diminuir(p, 13, True)}') o True foi adicionado
```

```
Digite o preço: R$100
A metade de R$R$100,00 é R$R$50,00
O dobro de R$R$100,00 é R$R$200,00
Aumentando 10%, temos R$R$110,00
Reduzindo 13%, temos R$87,00
```

reduzindo ainda mais o seu programa

Adicione o módulo moeda.py criado nos desafios anteriores, uma função chamada resumo(), que mostre na tela algumas informações geradas pelas funções que já temos no módulo criado até aqui.

no moeda.py do ex119:

```
def aumentar(preço=0, taxa=0, formato=False): foi adicionado o parâmetro formato que é falso
    res = preço + (preço * taxa/100) cálculo
    return res if formato is False else moeda(res) vai retornar o cálculo se for falso, se não vai passar pela função da moeda

def diminuir(preço=0, taxa=0, formato=False): foi adicionado o parâmetro formato que é falso
    res = preço - (preço * taxa/100) cálculo
    return res if formato is False else moeda(res) vai retornar o cálculo se for falso, se não vai passar pela função da moeda

def dobro(preço=0, formato=False): foi adicionado o parâmetro formato que é falso
    res = preço * 2 cálculo
    return res if not formato else moeda(res) vai retornar o cálculo se for falso, se não vai passar pela função da moeda

def metade(preço=0, formato=False): foi adicionado o parâmetro formato que é falso
    res = preço / 2 cálculo
    return res if not formato else moeda(res) vai retornar o cálculo se for falso, se não vai passar pela função da moeda

def moeda(preço = 0, moeda='R$'): função moeda, ela recebe parâmetros, caso não tenha um valor para eles, eles irão receber 0 e R$
    return f'{moeda}{preço:>.2f}'.replace('.', ',') vai retornar o R$ e o preço com 2 casas depois da vírgula, os pontos vão ser substituídos por vírgulas

def resumo(preço=0, taxaa=10, taxar=5): função resumo, recebeu taxa de aumento e taxa de redução
    print('-' * 30) separador
    print('RESUMO DO VALOR'.center(30)) título centralizado no meio
    print('-' * 30) separador
    print(f'Preço analisado: \t{moeda(preço)}') preço analisado, usou-se o \t para deixar os preços centralizados
    print(f'Dobro do preço: \t{dobro(preço, True)}') dobro do preço
    print(f'Metade do preço: \t{metade(preço, True)}') metade do preço
    print(f'{taxaa}% de aumento: \t{aumentar(preço, taxaa, True)}') taxa de aumento
    print(f'{taxar}% de redução: \t{diminuir(preço, taxar, True)}') taxa de redução
    print('-' * 30) separador
```

no teste.py do ex119:

```
from ex120 import moeda importando a moeda
```

p = float(input('Digite o preço: R$')) pedindo o preço
moeda.resumo(p, 20, 12) mostrando um valor para calcular

```
Digite o preço: R$100
------------------------------
       RESUMO DO VALOR
------------------------------
Preço analisado:    R$100,00
Dobro do preço:     R$200,00
Metade do preço:    R$50,00
20% de aumento:     R$120,00
12% de redução:     R$88,00
------------------------------
```

transformando módulos em pacotes

Crie um pacote chamado utilidadesCeV que tenha dois módulos internos chamados moeda e dado. Transfira todas as funções utilizadas nos desafios 116, 117 e 119 para o primeiro pacote e mantenha tudo funcionando.

faça ctrl + v e ctrl + c no diretório do 'ex119.py'
vai duplicar esse exercício e dê o nome de 'ex120.py'
clique com o botão direito em cima de 'ex120.py' > python package > dê o nome de 'utilidadescev'
clique com o botão direito em cima de 'utilidadescev' > python package > dê o nome de 'moeda'
clique com o botão direito em cima de 'utilidadescev' > python package > dê o nome de 'dado'

no '__init__.py' do dado: não escreva nada
no '__init__.py' da moeda: cole o código da moeda anterior
no '__init__.py' da 'utilidadecev': from ex120.utilidadescev import moeda, dado
exclua a moeda.py do 'utilidadescev'
no teste.py: from ex120.utilidadescev import moeda

```
Digite o preço: R$20
------------------------------
       RESUMO DO VALOR
------------------------------
Preço analisado:    R$20,00
Dobro do preço:     R$40,00
Metade do preço:    R$10,00
20% de aumento:     R$24,00
12% de redução:     R$17,60
------------------------------
```

,

entrada de dados monetários

faça ctrl + v e ctrl + c no diretório do 'ex120.py'
vai duplicar esse exercício e dê o nome de 'ex121.py'

vá no pacote dado > __init__.py

no dado > __init__.py:

```
def leiaDinheiro(msg): função leiaDinheiro
    ok = False o ok, começa falso
    valor = 0 o valor é 0
    while True: loop infinito
        try: operação
            n = input(msg).strip() vai pegar o valor e vai tirar os espaços em branco
            valor = float(n.replace(',', '.')) vai pegar o valor e vai substituir as vírgulas por ponto
        except ValueError: erro de valor
            print(f"\033[1;33m{n} \033[1;31mé um preço inválido!\033[m") vai dizer que é inválido
            continue e vai fazer esse loop toda vez
        if valor: se o valor
            ok = True ok se torna verdadeiro
        if ok: se o ok
            break vai parar
    return float(valor) vai retornar o valor em float
```

moeda > __init__.py (continuou igual, não mudou nada)

```
def aumentar(preço=0, taxa=0, formato=False): foi adicionado o parâmetro formato que é falso
    res = preço + (preço * taxa/100) cálculo
    return res if formato is False else moeda(res) vai retornar o cálculo se for falso, se não vai
passar pela função da moeda

def diminuir(preço=0, taxa=0, formato=False): foi adicionado o parâmetro formato que é falso
    res = preço - (preço * taxa/100) cálculo
    return res if formato is False else moeda(res) vai retornar o cálculo se for falso, se não vai
passar pela função da moeda

def dobro(preço=0, formato=False): foi adicionado o parâmetro formato que é falso
    res = preço * 2 cálculo
    return res if not formato else moeda(res) vai retornar o cálculo se for falso, se não vai passar
pela função da moeda

def metade(preço=0, formato=False): foi adicionado o parâmetro formato que é falso
    res = preço / 2 cálculo
    return res if not formato else moeda(res) vai retornar o cálculo se for falso, se não vai passar
pela função da moeda

def moeda(preço = 0, moeda='R$'): função moeda, ela recebe parâmetros, caso não tenha um
valor para eles, eles irão receber 0 e R$
    return f'{moeda}{preço:>.2f}'.replace('.', ',') vai retornar o R$ e o preço com 2 casas depois da
vírgula, os pontos vão ser substituídos por vírgulas

def resumo(preço=0, taxaa=10, taxar=5): função resumo, recebeu taxa de aumento e taxa de
redução
    print('-' * 30) separador
    print('RESUMO DO VALOR'.center(30)) título centralizado no meio
    print('-' * 30) separador
    print(f'Preço analisado: \t{moeda(preço)}') preço analisado, usou-se o \t para deixar os
preços centralizados
    print(f'Dobro do preço: \t{dobro(preço, True)}') dobro do preço
    print(f'Metade do preço: \t{metade(preço, True)}') metade do preço
```

```
    print(f'{taxaa}% de aumento: \t{aumentar(preço, taxaa, True)}') taxa de aumento
    print(f'{taxar}% de redução: \t{diminuir(preço, taxar, True)}') taxa de redução
    print('-' * 30) separador
```

unidadescev > __init__.py

```
from ex121.utilidadescev import moeda, dado apenas mudou para '121'
```

unidadescev > teste.py

```
from ex121.utilidadescev import moeda importando a moeda
from ex121.utilidadescev import dado adicionou esse, importou o dado

p = dado.leiaDinheiro('Digite o preço: R$') mostrou a função leiaDinheiro
moeda.resumo(p, 20, 12) moeda resumo
```

```
Digite o preço: R$
 é um preço inválido!
Digite o preço: R$ko
ko é um preço inválido!
Digite o preço: R$20,00
------------------------------
       RESUMO DO VALOR
------------------------------
Preço analisado:    R$20,00
Dobro do preço:     R$40,00
Metade do preço:    R$10,00
20% de aumento:     R$24,00
12% de redução:     R$17,60
------------------------------
```

tratamentos de erros
try (operação)
except (falhou)
finally (vai aparecer de qualquer jeito)

exemplo:

```
try: início da operação
a = int(input('Numerador: ')) vai pedir o numerador
b = int(input('Denominador: ')) vai pedir o denominador
r = a / b cálculo da divisão
except (ValueError, TypeError): quando tem mais de 1 exceção colocar em parênteses
print('Tivemos um problema com os tipos de dados que você digitou.') mensagem
except ZeroDivisionError: quando só tem 1 exceção não precisa de parênteses
print('Não é possível dividir um número por zero.') mensagem
except KeyboardInterrupt: outra exceção
print('O usuário preferiu não informar os dados.') mensagem
else: se não
print(f'O resultado é {r:.1f}')  mostrando o resultado
finally: o finally mostra uma mensagem independente do que acontecer
```

print('Volte sempre! Muito obrigado!') mensagem que vai aparecer sempre

funções aprofundadas

Reescreva a função leiaInt(), incluindo agora a possibilidade da digitação de um número de tipo inválido. Aproveite e crie também uma função leiaFloat() com a mesma funcionalidade.

```
def leiaInt(msg): função do int
   while True:  loop infinito
      try: operação
         n = int(input(msg)) a variável n recebe a 'msg'
      except (ValueError, TypeError): se derem esses valores
         print('ERRO! Por favor digite um número inteiro válido.') vai aparecer uma mensagem
         continue vai reiniciar o loop
      except KeyboardInterrupt: se der esse erro
         print('O usuário preferiu não digitar esse número.') vai aparecer uma mensagem
         return 0 vai retornar 0
      else: se não
         return n vai retornar o valor n

def leiaFloat(msg):  função do float
   while True: loop infinito
      try: operação
         n = float(input(msg)) a variável n recebe a 'msg'
      except (ValueError, TypeError): se derem esses valores
         print('ERRO! Por favor digite um número inteiro válido.') vai aparecer uma mensagem
         continue vai reiniciar o loop
      except KeyboardInterrupt: se der esse erro
         print('O usuário preferiu não digitar esse número.') vai aparecer uma mensagem
         return 0 vai retornar 0
      else: se não
         return n vai retornar o valor n

n1 = leiaInt('Digite um valor inteiro: ') pedindo para digitar um valor inteiro
n2 = leiaFloat('Digite um número real: ') pedindo um número real
print(f'O valor inteiro digitado foi {n1} e o valor real digitado foi {n2}') pedindo um valor inteiro e
um valor real
```

site está acessível?

Crie um código em Python que teste se o site pudim está acessível pelo computador usado.

```
import urllib é uma biblioteca padrão do python
import urllib.request importando o request da urllib

try: operação
   site = urllib.request.urlopen('http://www.pudim.com.br') vai tentar abrir essa url
except urllib.error.URLError: se der erro para abrir a url
   print('O site pudim não está acessível no momento.') vai mostrar uma mensagem
else: se não
   print('Consegui acessar o site pudim com sucesso!') vai aparecer outra mensagem
   print(site.read()) é um comando que mostra todo o código fonte do site
```

criando menu em python - arquivos em python - finalizando o projeto

Vamos criar um menu em Python, usando modularização.
Vamos ver como fazer acesso a arquivos usando o Python.
Vamos finalizar o projeto de acesso a arquivos em Python.

1- criou-se um diretório chamado ‘ex118’

2- clique nesse arquivo com o botão direito > new > directory > dê o nome de ‘lib’

3- clique na ‘lib’ com o botão direito > new > python package > dê o nome de ‘interface’

4- dentro de ‘interface’ tem um arquivo chamado ‘__init__.py’ coloque o código:

```
def leiaInt(msg): função inteira explicada anteriormente do leiaInt
    while True:
        try:
            n = int(input(msg))
        except (ValueError, TypeError):
            print('ERRO! Por favor digite um número inteiro válido.')
            continue
        except KeyboardInterrupt:
            print('O usuário preferiu não digitar esse número.')
            return 0
        else:
            return n

def linha(tam=42): função linha, vai receber um tamanho, se não informar o tamanho, vai ser de
42
    return '-' * tam vai retornar um tracinho, na quantidade do tamanho


def cabeçalho(txt): função cabeçalho, vai receber o parâmetro ‘txt’
    print(linha()) vai mostrar a linha
    print(txt.center(42)) vai mostrar o txt centralizado no tamanho de 21
    print(linha()) vai mostrar a linha


def menu(lista): função menu, que recebe a lista
    cabeçalho('MENU PRINCIPAL') vai mostrar ‘menu principal’ da função do cabeçalho
    c = 1 contador criado
    for item in lista: para cada item da lista
        print(f'{c} - {item}') vai mostrar o contador e o item
        c += 1 vai somar o contador
    print(linha()) vai mostrar a linha
    opc = leiaInt('Sua opção: ') vai pedir sua opção da função leiaInt
    return opc vai retornar o resultado da sua opção
```

6- clique com o botão direito em ‘ex118.py’ > new > python file > dê o nome de ‘sistema.py’

```
from ex118.lib.interface import * com o * vai importar tudo dos locais informados
from ex118.lib.arquivo import * com o * vai importar tudo dos locais informados
from time import sleep importando a biblioteca sleep

arq = 'cursoemvideo.txt' uma variável criada que vai guardar um arquivo
if not arquivoExiste(arq): se o arquivo não existir
    criarArquivo(arq) vai criar o arquivo


while True: loop infinito
```

```
   resposta = menu(['Ver pessoas cadastradas', 'Cadastrar nova pessoa', 'Sair do sistema']) vai mostrar as opções do menu
   if resposta == 1: se a resposta for igual a 1
      lerArquivo(arq) vai ler o arquivo
   elif resposta == 2: se a resposta for igual a 2
      cabeçalho('NOVO CADASTRO') título
      nome = str(input('Nome: ')) pedindo o nome
      idade = leiaInt('Idade: ') pedindo a idade
     cadastrar(arq, nome, idade) vai passar esses dados para a função cadastrar. vai passar o nome do arquivo, o nome e a idade
   elif resposta == 3: se a resposta for igual a 3
      cabeçalho('Saindo do sistema... Até logo!') mensagem de sair do sistema
   else: se não for nenhuma dessas opções
      print('Erro! Digite uma opção válida.') foi uma opção inválida
   sleep(2) vai pausar por 2 segundos
```

7- clique com o botão direito em 'lib' > python package > dê o nome de 'arquivo' > entre no arquivo '__init__.py'

```
from ex118.lib.interface import * com o * vai importar tudo dos locais informados


def arquivoExiste(nome): função criada para ver se o arquivo existe, vai receber o 'nome'
   try: início da operação
      a = open(nome, 'rt') com o 'open' vai tentar abrir um arquivo, vai ser o arquivo 'nome' e vai abrir para 'rt', read e text
      a.close() vai tentar abrir e fechar
   except FileNotFoundError: se o arquivo não for encontrado
      return False vai retornar falso
   else: se ele abriu
      return True vai retornar true

def criarArquivo(nome): função que cria o arquivo
   try: início da operação
      a = open(nome, 'wt+') com o 'open' vai tentar abrir o arquivo, com o 'wt+' vai escrever um arquivo de txt, se ele não existir ele vai ser criado com o +
      a.close() vai fechar o arquivo
   except: se houve algo fora do comum
      print('Houve um erro na criação do arquivo!') houve um erro na criação do arquivo
   else: se não
      print(f'Arquivo {nome} criado com sucesso!') o arquivo foi criado com sucesso


def lerArquivo(nome): função de ler o arquivo
   try: início da operação
      a = open(nome, 'rt') vai tentar abrir o arquivo, read e text
   except: se der erro
      print('Erro ao ler o arquivo!') mensagem de erro ao ler o arquivo
   else: se deu tudo certo
      cabeçalho('PESSOAS CADASTRADAS') vai cadastrar pessoas
      for linha in a: para cada linha dentro do arquivo
      dado = linha.split(';') vai separar a linha e vai adicionar dentro do dado
      dado[1] = dado[1].replace('\n', '') a idade vai receber a idade tirando o \n
      print(f'{dado[0]:<30}, {dado[1]:>3} anos') dado 0 é o nome (vai ser alinhado à esquerda), dado 1 é a idade (vai ser alinhado à direita)
```

finally: vai aparecer independente do que acontecer
a.close() vai fechar o arquivo

def cadastrar(arq, nome='desconhecido', idade=0): função cadastrar
try: operação
a = open(arq, 'at') vai tentar abrir o arquivo chamado ‘arq’, vai tentar abrir ele para colocar mais dados, ‘a’ é append, que é colocar coisas dentro do arquivo, ‘t’ é texto
except: se ocorrer um erro
print('Houve um erro na abertura do arquivo!') mensagem que deu erro
else: se não
try: vai tentar
a.write(f'{nome};{idade}\n') vai tentar escrever dentro do arquivo, o nome, a idade e uma quebra de linha
except: se der erro
print('Houve um ERRO na hora de escrever os dados!') mensagem de erro
else: se não der erro
print(f'Novo registro de {nome} adicionado.') mensagem de novo registro adicionado
a.close() vai fechar

```
--------------------------------------------
              MENU PRINCIPAL
--------------------------------------------
1 - Ver pessoas cadastradas
2 - Cadastrar nova pessoa
3 - Sair do sistema
--------------------------------------------
Sua opção: 2
--------------------------------------------
              NOVO CADASTRO
--------------------------------------------
Nome: Fernanda Maki Hirose
Idade: 17
Novo registro de Fernanda Maki Hirose adicionado.
```

```
-----------------------------------------
              MENU PRINCIPAL
-----------------------------------------
1 - Ver pessoas cadastradas
2 - Cadastrar nova pessoa
3 - Sair do sistema
-----------------------------------------
Sua opção: 1
-----------------------------------------
           PESSOAS CADASTRADAS
-----------------------------------------
Fernanda Maki Hirose          ,  17 anos
-----------------------------------------
              MENU PRINCIPAL
-----------------------------------------
1 - Ver pessoas cadastradas
2 - Cadastrar nova pessoa
3 - Sair do sistema
-----------------------------------------
Sua opção: 3
-----------------------------------------
      Saindo do sistema... Até logo!
-----------------------------------------
```