Tópicos fora da aula:

Atraso de tempo de resposta do python

Randomizar um resultado numérico

I) Operadores matemáticos

OBS: Para arredondar para 1 casa depois da vírgula **print("A media foi {:.1f}".format(m))**

II) Manipulação de Texto

a) Fatiamento

`frase[9]` resultado é V

`frase[9:13]` caracteres do 9 ao **12** resultado é Vide

`frase[9:21:2]` caracteres do 9 ao **20 pulando de 2 em 2** = VdoPto

b) Análise

`len(frase)` é o comprimento da frase = 21 (do 0 ao 20)

`frase.count('o')` quantas vezes aparece o "o" = 3

quantas vezes aparece o “o”, entre as letras de numero 0 e 12 = 1

A partir de qual posição que a frase “deo” está aparecendo. Caso não apareça, o retorno será -1

caso exista a palavra “curso”, o retorno será True

substitui o Python pelo Android

ou


colocar a frase em maiúscula ou minúscula

Coloca os caracteres para minúscula e somente a 1 letra fica maiúscula

Coloca os caracteres para minúscula e somente as 1 letras ficam maiúsculas

Remove os espaços inúteis antes e depois da frase

Remove apenas os últimos espaços, da direita

c) Divisão

cada palavra tem sua array própria. Como exemplo abaixo:

junta todos os elementos da frase e os separa por um “-”

III) Condições

Estrutura simples condicional

Estrutura condicional aninhada

IV) Ciclo de Repetição

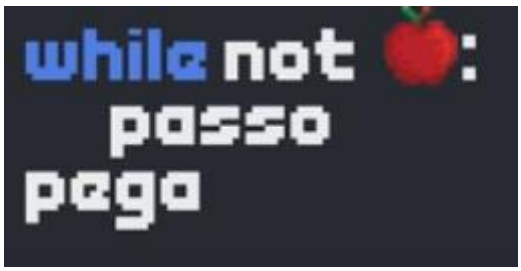
OBS: De 1 até 6 ele considera os números 1, 2, 3, 4 e 5. Ou seja, desconsidera o último número.

OBS: Se for de 0 até 6, desta forma consideram as 6 repetições! Pois as posições seriam 0, 1, 2, 3, 4, 5. Ou seja, 6 valores.

O ultimo argumento representa de quantos em quantos a contagem será pulada

Ex: Contar para trás = for c in range(6, 0, **-1**):

Ex2: Contar de 2 em 2 = for c in range(6, 0, **2**):



V) Tuplas (As tuplas são imutáveis)

No caso abaixo, tanto faz utilizar qualquer uma das estruturas for.

```
lanche = ('Hambúrguer', 'Suco', 'Pizza', 'Pudim', 'Batata Frita')

for comida in lanche:
    print(f'Eu vou comer {comida} ')

for cont in range(0, len(lanche)):
    print(f'Eu vou comer {lanche[cont]} na posição {cont}')

for pos, comida in enumerate(lanche):
    print(f'Eu vou comer {comida} na posição {pos}')

print('Comi pra caramba!')
```

OBS: A função **del(**lanche**)** apaga uma tupla inteira.

OBS2: Uma Tupla1 + outra Tupla2 é a concatenação entre os valores dela e não o somatório.

OBS3: para identificar a posição do Suco na tupla => tupla.**index(**“Suco”**)**

VI) Listas (Podem ser mutáveis)

Para criar uma lista com um range:

Para **adicionar** elementos novos na lista, usa-se o comando:

Para **adicionar** elementos em uma posição específica:

Para **remover elementos de uma lista**:

Para remover o **último elemento**:

Verificação para remoção:

Para alinhar valores de Listas:

Para alinhar ao contrário:

Exemplo de for em lista. Onde d é a posição que representa o valor dentro de uma lista.

OBS: O código abaixo pode ser lido da seguinte forma: Para cada elemento em lista faça um if

```
#numeros pares
for d in lista:
    if d%2 == 0:
        e = e + 1
print(f"A quantidade de números pares é: {e}")
```

Outro exemplo: Onde c é a posição e v é o valor

```
valores = []
valores.append(5)
valores.append(9)
valores.append(4)

for c, v in enumerate(valores):
    print(f'Na posição {c} encontrei o valor {v}!')
print('Cheguei ao final da lista.')
```

Para que uma variável receba a **cópia dos valores de uma Lista**, para que os valores de apenas esta variável sejam alterados:

```
a = [2, 3, 4, 7]
b = a[:]
b[2] = 8
print(f'Lista A: {a}')
print(f'Lista B: {b}')
```

```
Lista A: [2, 3, 4, 7]
Lista B: [2, 3, 8, 7]
```

OBS: Para identificar a quantidade de vezes que um valor específico apareceu:
Lista**.count(**valor buscado**)**

Outra forma de criar uma lista

Criando uma lista dentro de outra lista

Ficando da seguinte forma:

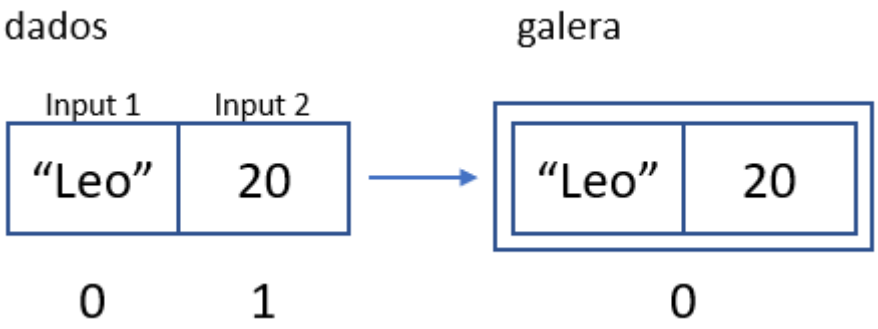
Exemplificado:

Dentro da pessoa 0, eu quero o item 0. No caso, é igual a “Pedro”.

Percorrendo uma lista com critérios. Trazer os nomes de quem tem igual ou mais de 21 anos

```
galera = list()
dados = list()

for c in range(0, 2):
    dados.append(input("Digite o nome: "))
    dados.append(int(input("Digite a idade: ")))
    galera.append(dados[:])
    dados.clear()

for p in galera:
    if p[1] >=21:
        print(p[0])
```

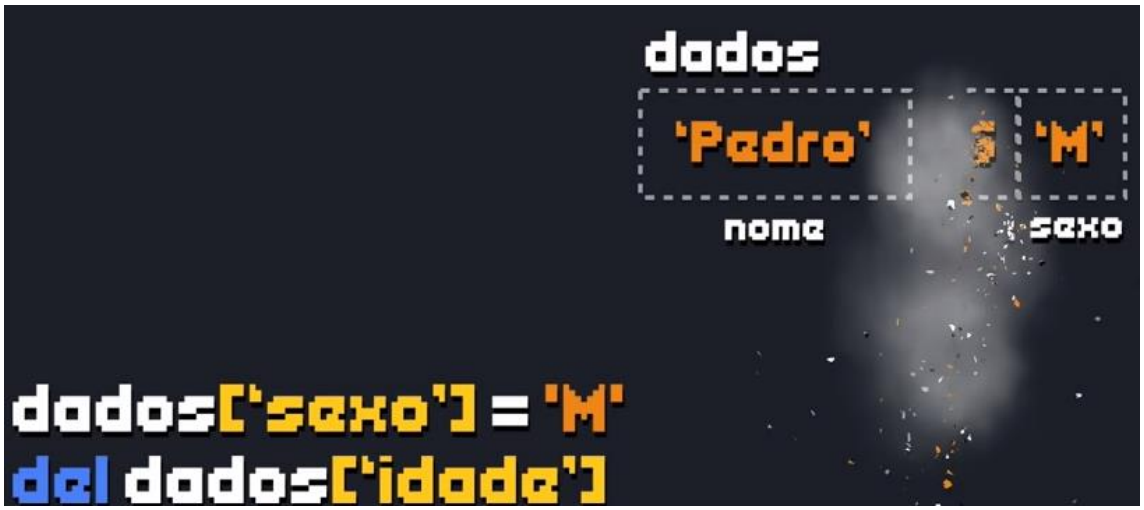
Neste caso acima, o algoritmo está realizando o input do nome, em seguida o input da idade. Depois transforma os dois em um valor único na lista (comando **galera.append(dados[:])**)



VII) Dicionários (índices literais para as listas)

Para adicionar dados, utilizar:

Para remover dados:



Exemplo de criação de um dicionário:

Para buscar os valores:

Para buscar as chaves:


```
print(filme.keys())
```

Para buscar ambos os valores:

```
print(filme.items())
```

Estrutura de repetição com dicionário:

É possível colocar um dicionário dentro de uma lista:

Onde as listas são identificadas por números e os dicionários são identificados por textos.

Exemplos práticos:

```
pessoas = {"Nome":"Daniel","Idade":28, "Sexo":"M"}

print(f"O {pessoas['Nome']} tem {pessoas['Idade']} anos")

# O Daniel tem 28 anos
# OBS: Se estiver usando aspas duplas para escrever o print, deve ser usado aspas simples para o dicionário
```

Outro exemplo:

```
pessoas = {"Nome":"Daniel","Idade":28, "Sexo":"M"}

for k in pessoas.keys():
    print(k)

# resultado:
#Nome
#Sexo
#Idade
```

Incluindo dicionário dentro de lista:

```
pessoa1 = {"Nome":"Daniel","Idade":28, "Sexo":"M"}
pessoa2 = {"Nome":"Carol","Idade":19, "Sexo":"F"}
lista = []

lista.append(pessoa1)
lista.append(pessoa2)

print(lista)
```

```
[{'Nome': 'Daniel', 'Idade': 28, 'Sexo': 'M'}, {'Nome': 'Carol', 'Idade': 19, 'Sexo': 'F'}]
```

Identificando os elementos:

```
print(lista[0]["Nome"])
```

```
Daniel
```

Para que os valores sejam “copiados” de um dicionário para que sejam “empacotados” em uma lista, usa-se o método **.copy()**

Obs: O .copy() é similar ao “Fatiamento” das listas [:]

```
aluno = {}
chamada = []

for c in range(0, 3):
    aluno["Nome"] = input("Digite um nome: ")
    aluno["Idade"] = int(input("Digite a idade: "))
    chamada.append(aluno.copy())

print(chamada)
```

```
Digite um nome: Daniel
Digite a idade: 28
Digite um nome: Carol
Digite a idade: 19
Digite um nome: Junior
Digite a idade: 30
[{'Nome': 'Daniel', 'Idade': 28}, {'Nome': 'Carol', 'Idade': 19}, {'Nome': 'Junior', 'Idade': 30}]
```

Percorrendo dicionários dentro de listas:

```
aluno = {}
chamada = []

for c in range(0, 2):
    aluno["Nome"] = input("Digite um nome: ") #armazena nome no dicionário
    aluno["Idade"] = int(input("Digite a idade: ")) #armazena idade no dicionário
    chamada.append(aluno.copy()) #os dois valores do dicionário irão compor uma posição da lista

for d in chamada: #percorre as listas inteiras inputadas em chamada
    for k, v in d.items():#percorre as keys e os valores dentro das listas
        print(f"{k} = {v}")
```

Segunda forma de percorrer:

```
aluno = {}
turma = []
j = 0

for c in range(0, 2):
    aluno["Nome"] = input("Digite um nome: ")
    aluno["Idade"] = int(input("Digite uma idade: "))
    turma.append(aluno.copy())

for d in turma:
    print(f"O nome é {turma[j]['Nome']} e a idade é {turma[j]['Idade']}")
    j += 1
```

```
O nome é daniel e a idade é 19
O nome é carol e a idade é 20
```

VIII) Funções

Utilizado em rotinas

Função com passagem de parâmetros:

Onde a frase “Sistema de Alunos” irá substituir o msg (parâmetro interno da função)

OBS: O msg serve como uma “variável” qualquer que irá funcionar apenas dentro da função

Exemplo:

```
def linha(txt):
    print("-"*30)
    print(txt)


linha("Primeiro Titulo")
linha("Segundo Titulo")
linha("Terceiro Titulo")
```

```
------------------------------
Primeiro Titulo
------------------------------
Segundo Titulo
------------------------------
Terceiro Titulo
```

```
def soma(a,b):
    s = a+b
    print(s)

soma(4,5)
soma(8,9)
soma(2,1)
```

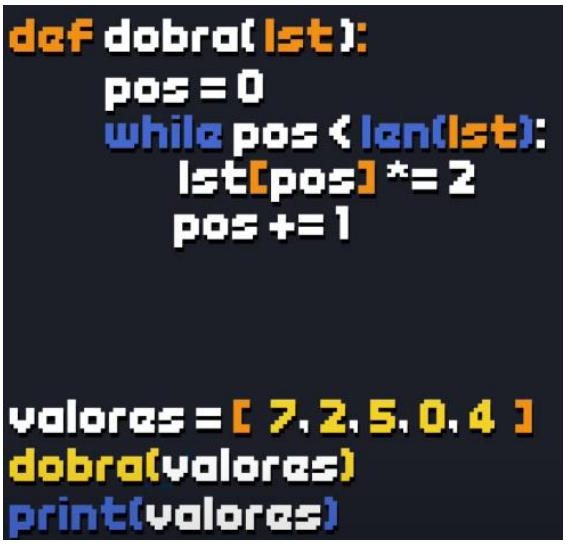
Conceito de empacotar parâmetros:

Quando não se sabe a quantidade exata de parâmetros, coloca-se * conforme abaixo. No caso abaixo, o **núm vai virar uma Tupla**


```
def media(*num):
    s = 0
    for c in num:
        s += c
    m = s/len(num)
    print(f"A média foi {m}")


media(10,9)
media(10, 8 ,6)
media(5, 11, 30, 18, 53)
```

Além disso, é possível que uma **função receba uma Lista como parâmetro** conforme exemplo abaixo:

Função para dobrar os valores de uma lista



O mesmo resultado pode ser obtido desta outra forma:

```python
def dobra(lista):
    c=0
    while c < len(lista):
        lista[c] *= 2
        c +=1
    print(lista)

lista = [7, 8 ,6 ,2, 6]

dobra(lista)
```

OBS: Para o **Python toda passagem de parâmetro é por referência**, diferente de outras linguagens como o Java que é por valor.

Função help() serve para verificar a documentação de uma outra função. Ex: help(print) irá mostrar as funcionalidades da função print

DOCSTRINGs são as documentações/descrições de uma função. Ex: Docstring da função criada chamada contador

```python
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!')
```

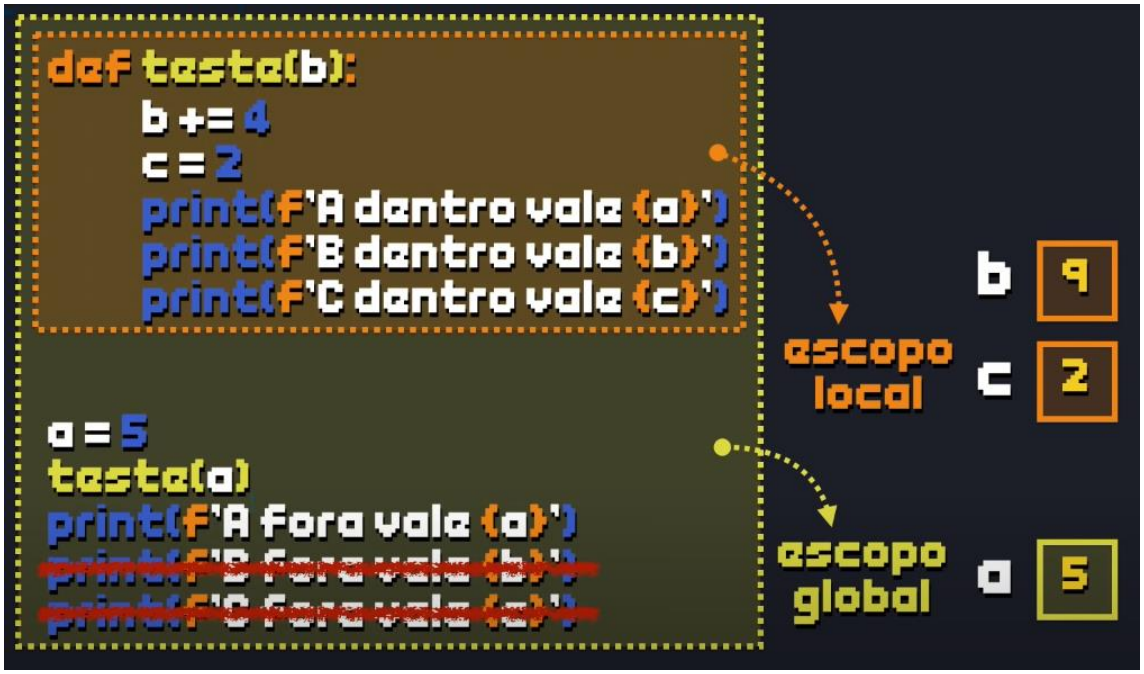
Funções com parâmetros opcionais

No exemplo abaixo, como c não está recebendo parâmetro, como default o valor do mesmo será 0

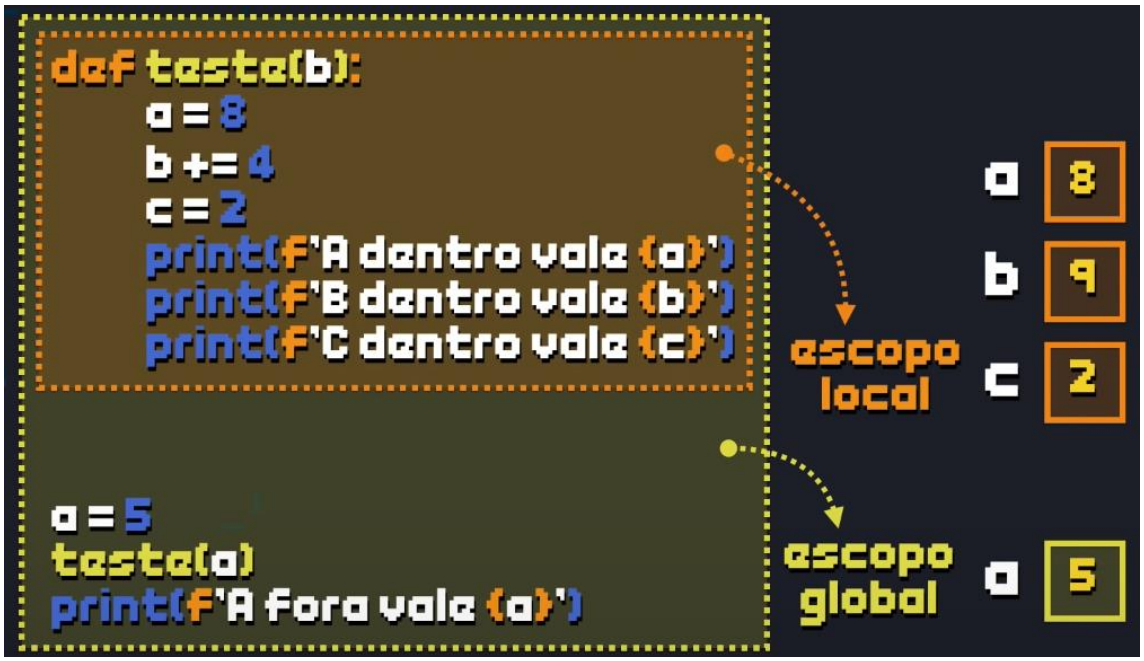
Escopo de variáveis

A variável a existe dentro e fora da função. Este evento é chamado de variável global.

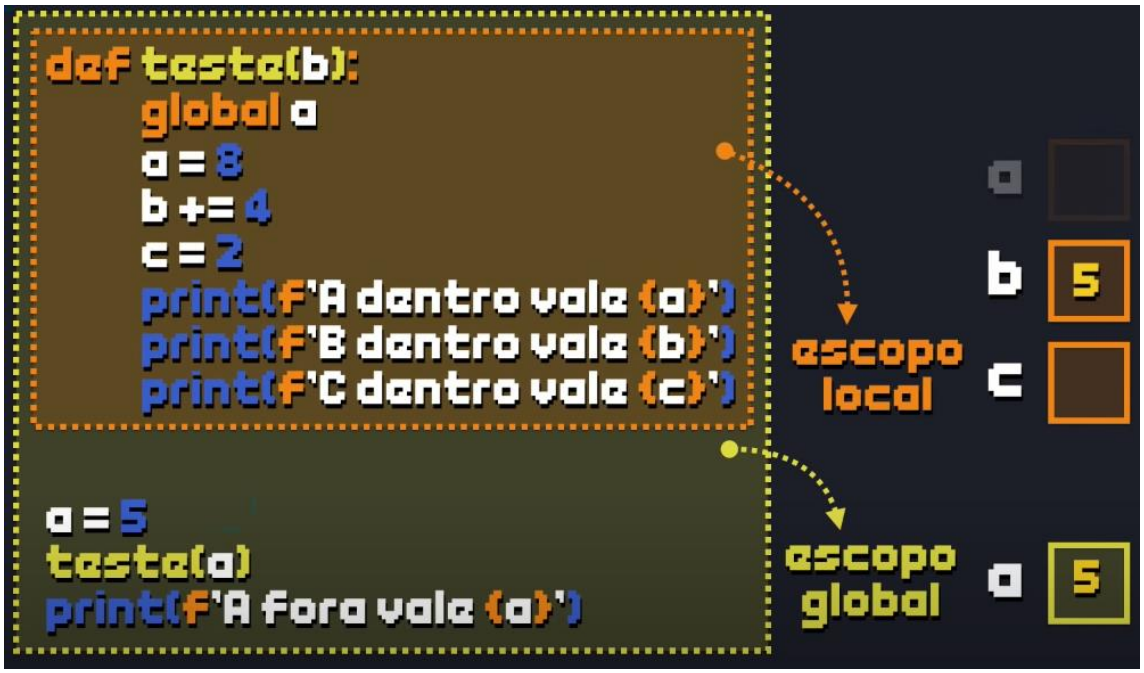
Já as variáveis b e c existem apenas na função, ou seja, variáveis locais.



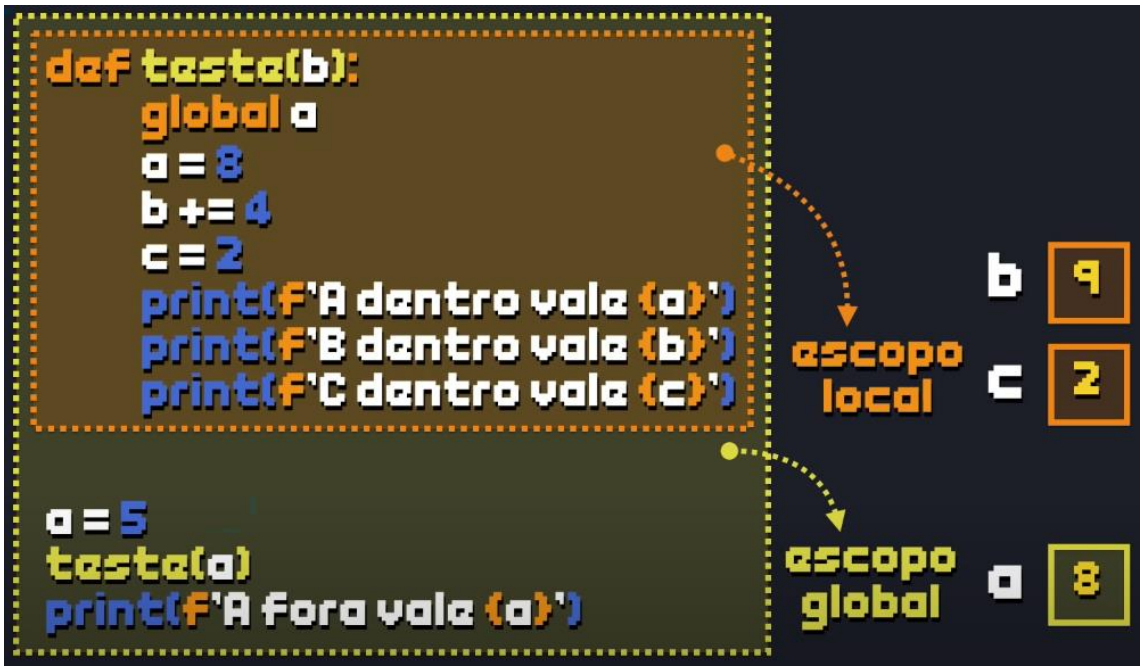
Por padrão, nos casos em que existirem mesmos nome de variáveis, uma local e outra global. As variáveis terão valores diferentes dependendo se ela estiver posicionada como local ou global. Como no exemplo da variável a no quadro abaixo.



Caso necessário, é possível importar a variável global para uma função utilizando a frase **global** + variável. Conforme quadro abaixo:



Desta forma a variável a global passará a valer 8 conforme abaixo:



IX) Funções com retorno

Serve para armazenar um valor resultante de uma função em uma variável. No exemplo abaixo, a variável **resp** recebeu o valor do somatório das variáveis a + b + c

Atribuindo o resultado de uma função à uma variável será possível realizar algo como o exemplo abaixo:

```
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}.')
```

Caso não deseja atribuir a uma variável, pode-se atribuir o valor a uma outra função, como no exemplo abaixo foi utilizado o print

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


print(somar( 3, 2, 5 ))
```

X) Módulos

OBS: Qualquer arquivo .py pode ser importado como módulo.Abaixo está um exemplo de importação de um módulo:

Código anterior sem modularização:

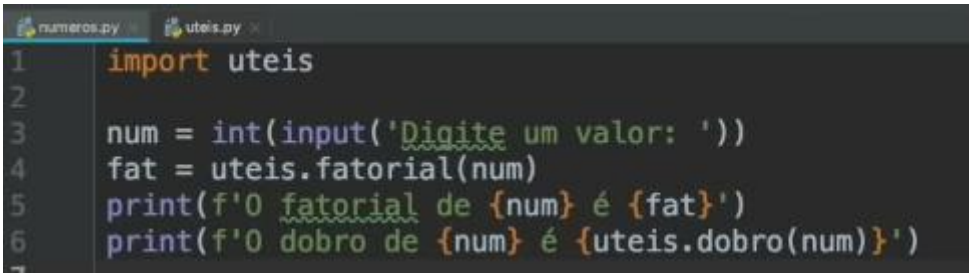
```
def fatorial(n):
    f = 1
    for c in range(1, n+1):
        f *= c
    return f


def dobro(n):
    return n*2


def triplo(n):
    return n*3


num=int(input("Digite um valor"))
fat=fatorial(num)
print(f"O fatorial de {num} é {fat}")
```

O código acima foi desmembrado em dois módulos:

Módulo **“uteis”**, que contém todas as funções.

E módulo do Código principal, que importa “úteis”

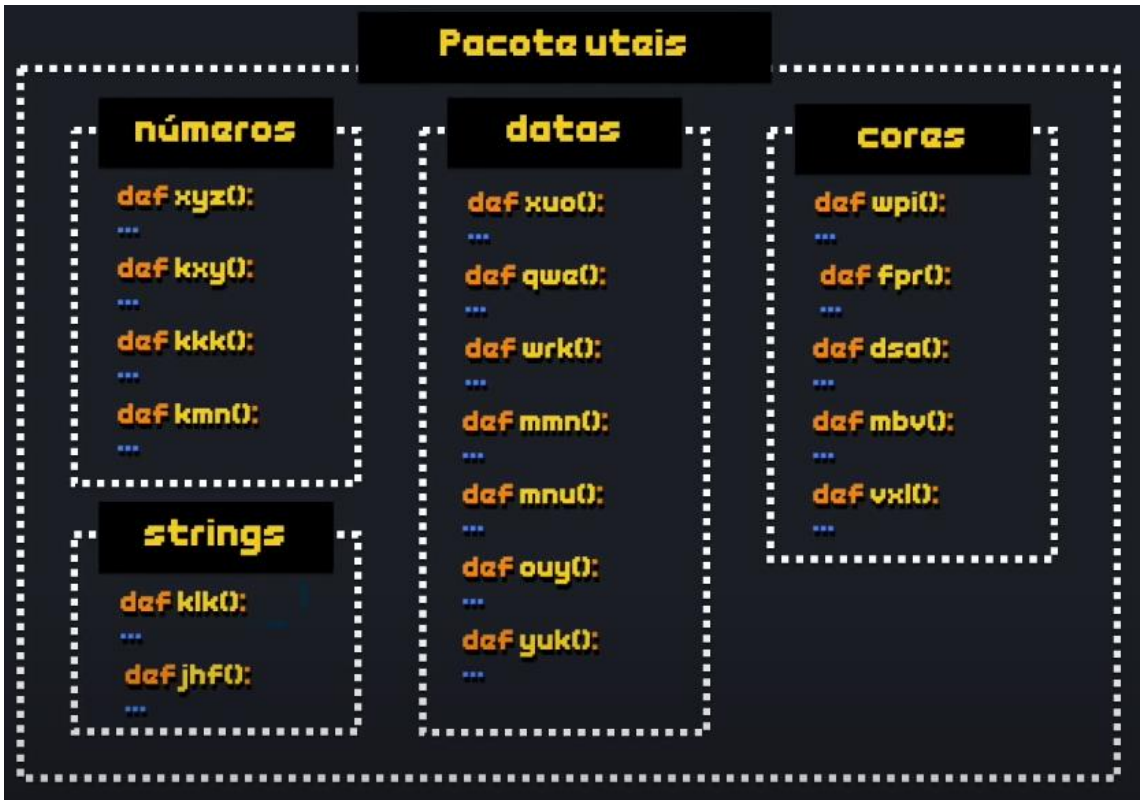
OBS: Além disso, para rodar o código, será necessário que as funções no módulo principal tenham o código “uteis.” Ex uteis.fatorial(num)



XI) Pacotes / Bibliotecas

Pode-se dizer que os pacotes são pastas que contém vários módulos. Os pacotes podem ser divididos em assuntos, conforme representado no quadro abaixo:

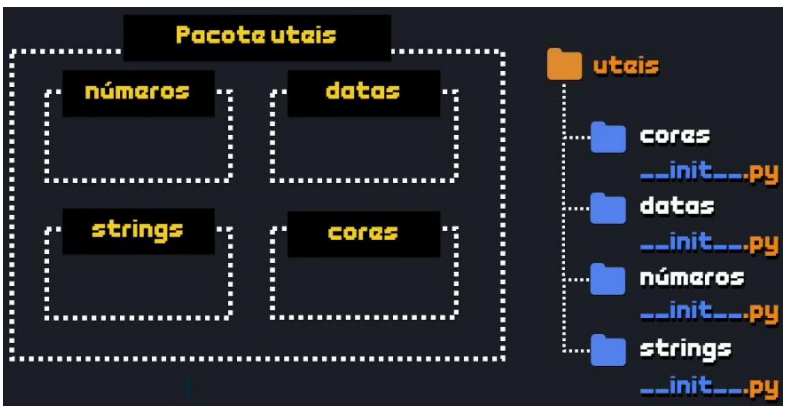
Onde número, strings, datas e cores são os módulos do pacote uteis.



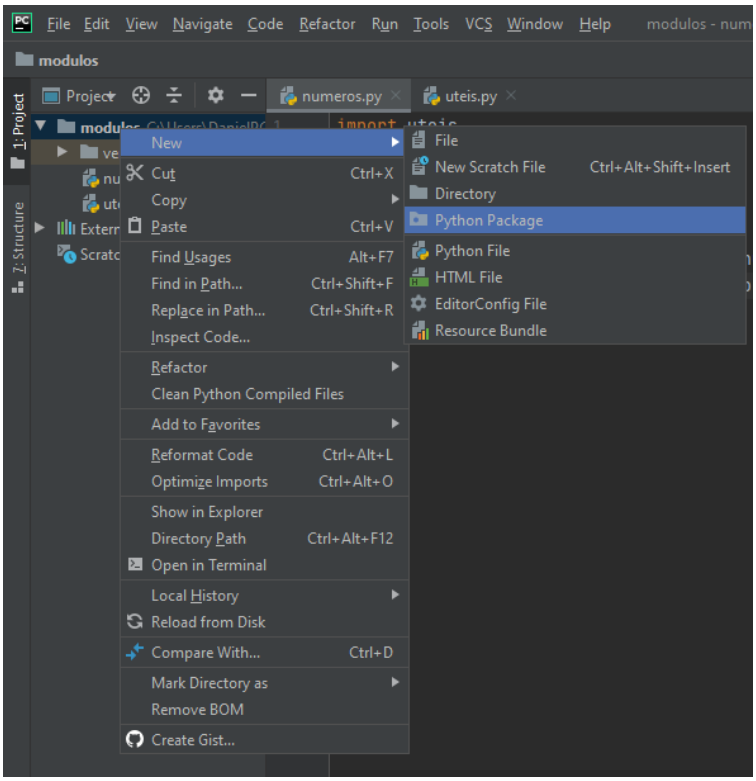
Para importar, basta utilizar o código conforme mostrado abaixo:

```
import uteis
from uteis import datas
from uteis import cores
```

Dentro de um projeto python, toda pasta é considerada um pacote

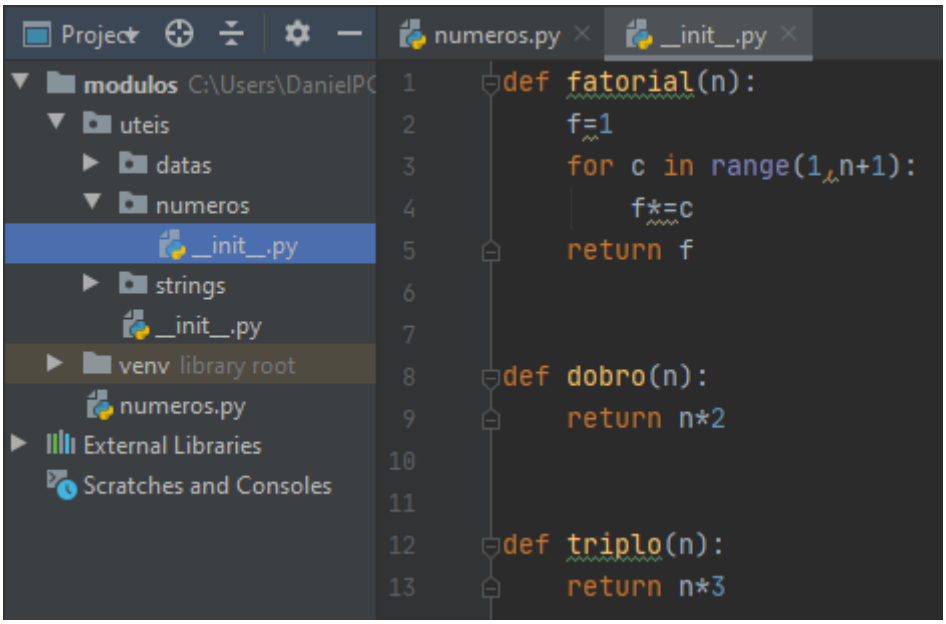


Para criar um pacote dentro do pycharm:



No exemplo abaixo, criei um pacote chamado uteis e dentro dele um módulo chamado números.

Dentro do módulo números coloquei o código contendo as funções.



Depois é só importar a biblioteca para o programa principal.

```
from uteis import numeros

num = int(input("Digite um numero: "))
fat = numeros.fatorial(num)
print(f"O fatorial de {num} é {fat}")
print(f"O dobro de {num} é {numeros.dobro(num)}")
print(f"O triplo de {num} é {numeros.triplo(num)}")
```

XII) Tratamento de Erros e Exceções

Um erro que não se dá de forma sintática, ou seja, um comando que normalmente funcionaria, chamamos de exceção. Conforme exemplo abaixo onde a variável x não foi declarada:

Outros exemplos de exceção são:

Estrutura do código para tratamento de exceção:

```
try:
    a = int(input('Numerador: '))
    b = int(input('Denominador: '))
    r = a / b
except:
    print('Infelizmente tivemos um problema :(')
else:
    print(f'O resultado é {r:.1f}')
finally:
    print('Volte sempre! Muito obrigado!')
```

```
Numerador: 45
Denominador: nove
Infelizmente tivemos um problema :(
Volte sempre! Muito obrigado!
```

Podemos também utilizar a função de tratamento de erro para nos mostrar em tela qual erro que ocorreu, utilizando a expressão **except Exception as** erro (**qualquer nome de variável**) conforme exemplo abaixo:

```
try:
    a = int(input('Numerador: '))
    b = int(input('Denominador: '))
    r = a / b
except Exception as erro:
    print(f'Problema encontrado foi {erro.__class__}')
else:
    print(f'O resultado é {r:.1f}')
finally:
    print('Volte sempre! Muito obrigado!')
```

```
Numerador: 7
Denominador: 0
Problema encontrado foi <class 'ZeroDivisionError'>
Volte sempre! Muito obrigado!
```

Além disso, é possível criar vários tipos de tratamentos de erros em um único só código, conforme abaixo:


```
try:
    a = int(input('Numerador: '))
    b = int(input('Denominador: '))
    r = a / b
except (ValueError, TypeError):
    print('Tivemos um problem com os tipos de dados que você digitou.')
except ZeroDivisionError:
    print('Não é possível dividir um número por zero!')
except KeyboardInterrupt:
    print('O usuário preferiu não informar os dados!')
else:
    print(f'O resultado é {r:.1f}')
finally:
    print('Volte sempre! Muito obrigado!')
```

```
Numerador: 7
Denominador: 0
Não é possível dividir um número por zero!
Volte sempre! Muito obrigado!
```

```
Numerador: 8
Denominador: dois
Tivemos um problem com os tipos de dados que você digitou.
Volte sempre! Muito obrigado!
```