Aula 2 - Conceitos de Indentação e Laços de Repetição

Indentação

In [7]:
```
#Diferentemente da maioria das linguagens, onde para criar um bloco de có
#Essa indentação pode ser feita utilizando um "tab".
#Desta forma, no Python indentação faz parte da sintaxe
```

In [5]:
```
#O exemplo abaixo mostra como seria um bloco em "PHP" e em "Python"

#em PHP o bloco de comandos do if é separado por chaves
#if(nota>60){
#    echo "Aprovado";
#    echo "Passou Direto";
#}

#em Python, o bloco de comandos do if será separado por um tab (indentaçã
#if(nota>60):
#    print("Aprovado");
#    print("Passou Direto");
```

Laço de Repetição WHILE

In [1]:
```
#Um laço de repetição WHILE repete a execução de um bloco de código até q
#Se a condição nunca for alcançada, o laço será executado infinitamente
```

In [1]:
```
# exemplo abaixo pede a entrada de um nome e imprime na tela, até que a p
#o comando break encerra o laço

condicao = True
while condicao==True:
    nome = input('Qual o seu nome: ')
    print(f'Seu nome é {nome}')
    if nome == 'sair':
        break
        print('Acabou')
```

```
Seu nome é Paulo
Seu nome é Zé
Seu nome é sair
```

In [4]:
```
#outra forma de fazer é colocando valor false na variável 'condicao'
condicao = True
while condicao:
    nome = input('Qual o seu nome: ')
    print(f'Seu nome é {nome}')
    if nome == 'sair':
        condicao=False
        print('Acabou')
```

```
Seu nome é Paulo
Seu nome é sair
Acabou
```

In [7]:
```
#Também é possível utilizar while com contadores numéricos
#O exemplo abaixo imprime o valor da variável cont até chegar em 10
contador=0
```

```
while contador <= 10:
    contador = contador + 1
    print(contador)
print('Acabou')
```

```
1
2
3
4
5
6
7
8
9
10
11
Acabou
```

Operadores de Atribuição

In [ ]:
```
#Operadores de atribuição é uma forma de deixar mais curta uma operação d
#Por exemplo, ao invés de fazer cont=cont+1, podemos fazer cont+=1
#Os operadores são: += -= *= /= //= **= %=
# Quando usa //= (divisão inteira) seria igual cont=cont//2 e cont//=2
```

In [ ]:
```
#O comando "continue" serve para pular um iteração de uma laço de repetiç
#O comando "break" encerra o laço
```

In [4]:
```
#O exemplo abaixo tem um laço que vai ser executado até a variável contad
#Em cada interação fazemos a atribuição de soma de 1 no contador
#Mas vamos pular iterações quando o contador for igual a 6 ou tiver entre
#Se o contador for igual a 40, é para encerrar o laço

contador = 0
while contador <= 50:
    contador += 1

    if contador == 6:
        print('Não vou mostrar o 6.')
        continue

    if contador >= 10 and contador <= 27:
         print('Não vou mostrar o', contador)
         continue

    print(contador)

    if contador == 40:
        break

print('Acabou')
```

```
1
2
3
4
5
Não vou mostrar o 6.
7
8
9
Não vou mostrar o 10
Não vou mostrar o 11
Não vou mostrar o 12
Não vou mostrar o 13
Não vou mostrar o 14
Não vou mostrar o 15
Não vou mostrar o 16
Não vou mostrar o 17
Não vou mostrar o 18
Não vou mostrar o 19
Não vou mostrar o 20
Não vou mostrar o 21
Não vou mostrar o 22
Não vou mostrar o 23
Não vou mostrar o 24
Não vou mostrar o 25
Não vou mostrar o 26
Não vou mostrar o 27
28
29
30
31
32
33
34
35
36
37
38
39
40
Acabou
```

Aninhamento de While (Um while dentro de outro)

In [3]:
```
#quando é inserido um while dentro do outro, para cada iteração do primei
#O exemplo abaixo mostra como funciona a execução.
#Para cada linha (no primeiro laço) é executado todas as colunas (segundo
```

In [4]:
```
qtd_linhas = 5
qtd_colunas = 5

linha = 1
#enquanto linha menor/igual a 5
while linha <= qtd_linhas:
    coluna = 1
    #para cada valor da linha, serão executados todas as colunas
    while coluna <= qtd_colunas:
        print(f'{linha=} {coluna=}')
        coluna += 1
```

```
    linha += 1
print('Acabou')
```

```
linha=1 coluna=1
linha=1 coluna=2
linha=1 coluna=3
linha=1 coluna=4
linha=1 coluna=5
linha=2 coluna=1
linha=2 coluna=2
linha=2 coluna=3
linha=2 coluna=4
linha=2 coluna=5
linha=3 coluna=1
linha=3 coluna=2
linha=3 coluna=3
linha=3 coluna=4
linha=3 coluna=5
linha=4 coluna=1
linha=4 coluna=2
linha=4 coluna=3
linha=4 coluna=4
linha=4 coluna=5
linha=5 coluna=1
linha=5 coluna=2
linha=5 coluna=3
linha=5 coluna=4
linha=5 coluna=5
Acabou
```

Exemplo: Faça um algoritmo que peça para digitar um nome. EM seguida, imprima na tela cada letra do nome separada por '-'. Por exemplo: Entra com o nome "Paulo" e imprima na tela "P-a-u-l-o".

In [5]:
```
nome=input()
tamanho=len(nome)
contador=0
resultado=''
while contador<tamanho:
    resultado+=nome[contador]+'-'
    contador+=1
print(resultado)
```

```
A-r-t-h-u-r-
```

Exemplo: Faça uma calculadora que calcule as operações básicas. O usuário deverá digitar os números e o operador, e o algoritmo deverá retorar o resultado e perguntar se deseja sair. É preciso fazer as validações das entradas do usuário.

In [6]:
```
sair=False
while sair==False:
    numero_1 = input('Digite um número: ')
    numero_2 = input('Digite outro número: ')
    operador = input('Digite o operador (+-/*): ')

    numeros_validos = None

    try:
        num_1_float = float(numero_1)
```

```
        num_2_float = float(numero_2)
        numeros_validos = True
    except:
        numeros_validos = None

    if numeros_validos is None:
        print('Um ou ambos os números digitados são inválidos.')
        continue

    operadores_permitidos = '+-/*'

    if operador not in operadores_permitidos:
        print('Operador inválido.')
        continue

    if len(operador) > 1:
        print('Digite apenas um operador.')
        continue

    #fazendo os calculos
    if operador=='+':
        print(num_1_float + num_2_float)
    elif operador=='-':
        print(num_1_float - num_2_float)
    elif operador=='/':
        print(num_1_float / num_2_float)
    elif operador=='*':
        print(num_1_float * num_2_float)


#se o usuário digitar algo começando com 's', sair receberá True e encerr
    sair = input('Quer sair? [s]im: ').lower().startswith('s')

    if sair is True:
        break
```

```
5.0
```

Else no While (específico do Python)

In [ ]:
```
#Este recurso é exclusivo do Python. Colocando o else no While, ele será
#Se houve algum interrupção no while por meio do break, o else não é exec
```

In [28]:
```
#Neste exemplo, na segunda iteração o break é executado, e não ativa o el
nome = 'Paulo'

i = 0
while i < len(nome):
    letra = nome[i]

    if letra == 'a':
        break

    print(letra)
    i += 1
else:
    print('Não encontrei a letra na string.')
print('Fora do while.')
```

```
P
Fora do while.
```

In [29]:
```
#Já neste caso, como o while foi executado inteiro, sem interrupção, foi
string = 'Valor qualquer'

i = 0
while i < len(string):
    letra = string[i]
    print(letra)
    i += 1
else:
    print('Não encontrei um espaço na string.')
print('Fora do while.')
```

```
V
a
l
o
r

q
u
a
l
q
u
e
r
Não encontrei um espaço na string.
Fora do while.
```

Laço de Repetição FOR

In [4]:
```
#O for pode ser usado mais facilmente do que o while quando já é conhecid
#No for não é necessário controlar o contador, ele avança automaticamente
```

In [3]:
```
#No exemplo para cada interação, a variavel letra receberá um aposição da
#No caso, colocaremos um - depois de cada letra
texto1= 'Paulo'
texto2= ''
for letra in texto1:
    texto2 += '-'+letra
    print(letra)
print(texto2)
```

```
P
a
u
l
o
*P*a*u*l*o
```

In [11]:
```
#A função range serve para gerar números no Python
#range(inicio, fim, step)
#Neste caso numeros receberá valores de 1 até 9 (10 não entra) com step d
numeros = range(1, 10, 2)
#com o for conseguimos pegar números individualmente
for numero in numeros:
    print(numero)
```

```
1
3
5
7
9
```

Out[11]: range

In [15]:
```python
#range(10) considera inicio=0 e step=1. É como se fosse range(0,10,1)
#No  exemplo, para i variando de 0 até 9, imprime i.
#Mas quando i=2 pula a iteração e quando for igual a 8, encerrará o laço
#Neste caso o else não será executado porque break interrompeu a execução
for i in range(10):
    if i == 2:
        print('i igual a 2, pulando a iteração')
        continue

    if i == 8:
        print('i igual 8, encerra o laço. O else não executará')
        break

    print(i)

else:
    print('For completo com sucesso!')
```

```
0
1
i é 2, pulando...
3
4
5
6
7
i é 8, seu else não executará
```

In [ ]: