Computação 1 - Python
Aula 9 - Teórica: Interferindo no fluxo de repetição: Break e Continue
Laços Aninhados

Estrutura de Repetição - *break* e *continue*

break e **continue** : Comandos que permitem alterar o fluxo da estrutura de repetição.

```
# Tente descobrir o que faz esta função
# int → int
def soma(numero):
    soma = 0
    contador = 0
    while contador < numero:
        if contador == 5:
            break
        soma = soma + contador
        contador = contador + 1
    return soma
```

Qual a saída desta função se a chamada for soma(10)?

Estrutura de Repetição - *break* e *continue*

break e **continue** : Comandos que permitem alterar o fluxo da estrutura de repetição.

```
# Tente descobrir o que faz esta função
# int → int
def soma(numero):
    soma = 0
    contador = 0
    while contador < numero:
        if contador == 5:
            break
        soma = soma + contador
        contador = contador + 1
    return soma
```

Qual a saída desta função se a chamada for soma(10)? 10

O comando *break* interrompe o "loop" quando contador == 5

Estrutura de Repetição - *break* e *continue*

break e **continue** : Comandos que permitem alterar o fluxo da estrutura de repetição.

```
# Tente descobrir o que faz esta função
# int → int
def soma1(numero):
    soma = 0
    contador = 0
    while contador < numero:
        contador = contador + 1
        if contador == 5:
            continue
        soma = soma + contador
    return soma
```

Qual a saída desta função se a chamada for soma1(10)?

Estrutura de Repetição - *break* e *continue*

break e **continue** : Comandos que permitem alterar o fluxo da estrutura de repetição.

```
# Tente descobrir o que faz esta função
# int → int
def soma1(numero):
    soma = 0
    contador = 0
    while contador < numero:
        contador = contador + 1
        if contador == 5:
            continue
        soma = soma + contador
    return soma
```

Qual a saída desta função se a chamada for soma1(10)? 50

O comando *continue* pula para a próxima execução do "loop" quando contador == 5, ou seja, não acumula a soma quando contador == 5 !

Estrutura de Repetição - *break* e *continue*

break e **continue** : Comandos que permitem alterar o fluxo da estrutura de repetição.

```
# Tente descobrir o que faz esta função
# int → int
def soma2(numero):
    soma = 0
    contador = 0
    while contador < numero:
        if contador == 5:
            continue
        soma = soma + contador
        contador = contador + 1
    return soma
```

Qual a saída desta função se a chamada for soma2(10)?

Estrutura de Repetição - *break* e *continue*

break e **continue** : Comandos que permitem alterar o fluxo da estrutura de repetição.

```
# Tente descobrir o que faz esta função
# int → int
def soma2(numero):
    soma = 0
    contador = 0
    while contador < numero:
        if contador == 5:
            continue
        soma = soma + contador
        contador = contador + 1
    return soma
```

Qual a saída desta função se a chamada for soma2(10)?

Nenhuma!! Fica num loop infinito!!!

Estrutura de Repetição - *break* e *continue*

Faça uma função que gere números aleatórios entre 1 e 10 e calcule a soma destes números até que seja gerado o número 5.
Use a função **randint(inicio,fim)** do módulo random para gerar um número aleatório, onde os valores de (início,fim) representam o intervalo desejado para os números a serem gerados.
Exemplo: randint(1,10) → gera um número aleatório entre 1 e 10, inclusive.

```
from random import randint
# função que soma números gerados aleatóriamente

# sem parâmetro → int
def somaAleatoria():
    soma = 0
    numero = randint(1,10)
    while numero != 5:
        soma = soma + numero
        numero = randint(1,10)
    return soma
```

Estrutura de Repetição - *break* e *continue*

Faça uma função que gere números aleatórios entre 1 e 10 e calcule a soma destes números até que seja gerado o número 5.
Use a função **randint(inicio,fim)** do módulo random para gerar um número aleatório, onde os valores de (início,fim) representam o intervalo desejado para os números a serem gerados.
Exemplo: randint(1,10) → gera um número aleatório entre 1 e 10, inclusive.

```
from random import randint
# função que soma números gerados aleatóriamente

# sem parâmetro → int
def somaAleatoria():
    soma = 0
    while True: # True indica um loop infinito

        COMPLETE A FUNÇÂO


    return soma
```

Estrutura de Repetição - *break* e *continue*

Faça uma função que gere números aleatórios entre 1 e 10 e calcule a soma destes números até que seja gerado o número 5.
Use a função **randint(inicio,fim)** do módulo random para gerar um número aleatório, onde os valores de (início,fim) representam o intervalo desejado para os números a serem gerados.
Exemplo: randint(1,10) → gera um número aleatório entre 1 e 10, inclusive.

```
from random import randint
# função que soma números gerados aleatóriamente

# sem parâmetro → int
def somaAleatoria():
    soma = 0
    while True: # True indica um loop infinito
        numero = randint(1,10)
        if numero ==5:
            break # Interrompe o loop infinito
        soma = soma + numero
    return soma
```

Estrutura de Repetição - *break* e *continue*

Também podemos usar *break* e *continue* com *for*.

```
# Tente descobrir o que faz esta função
# sem parâmetro → int
def Exemplo1():
    lista = [ ]
    for x in range(1, 11):
        if x ==5:
            break
        lista += [x]
    return lista
```

O que será retornado na chamada Exemplo1()?

Estrutura de Repetição - *break* e *continue*

Também podemos usar *break* e *continue* com *for*.

```
# Tente descobrir o que faz esta função
# sem parâmetro → int
def Exemplo1():
    lista = [ ]
    for x in range(1, 11):
        if x ==5:
            break
        lista += [x]
    return lista
```

O que será retornado na chamada Exemplo1()?
[1,2,3,4]

Estrutura de Repetição - *break* e *continue*

Também podemos usar *break* e *continue* com *for*.

```
# Tente descobrir o que faz esta função
# sem parâmetro → int
def Exemplo2():
    lista = [ ]
    for x in range(1, 11)
        if x ==5:
            continue
        lista += [x]
    return lista
```

O que será retornado na chamada Exemplo2()?

Estrutura de Repetição - *break* e *continue*

Também podemos usar *break* e *continue* com *for*.

```
# Tente descobrir o que faz esta função
# sem parâmetro → int
def Exemplo2():
    lista = [ ]
    for x in range(1, 11)
        if x ==5:
            continue
        lista += [x]
    return lista
```

O que será retornado na chamada Exemplo2()?
[1,2,3,4,6,7,8,9,10]

Estrutura de Repetição - *break* e *continue*

1. Diga o que é retornado pela função abaixo para os seguintes valores de entrada: 501, 745, 384, 2, 7 e 1.
2. O que faz a função ?

```
# Tente descobrir o que faz esta função
# int → list
def contagemcedulas(valor):
    cedulas = 0
    atual = 50
    apagar = valor
    contagem = []
    while True:
        if atual <= apagar:
            apagar -= atual
            cedulas += 1
        else:
            contagem += [(cedulas,atual)]
            if apagar <= 1:
                break
            if atual == 50:
                atual = 20
            elif atual == 20:
                atual = 10
            elif atual == 10:
                atual = 5
            elif atual == 5:
                atual = 2
            cedulas = 0
    return contagem
```

Estrutura de Repetição - *break* e *continue*

3. Modifique a função para considerar cédulas de 100.
4. Modifique a função para retornar uma mensagem de erro caso o valor não possa ser completamente pago apenas por cédulas.

```
# Tente descobrir o que faz esta função
# int → list
def contagemcedulas(valor):
    cedulas = 0
    atual = 50
    apagar = valor
    contagem = []
    while True:
        if atual <= apagar:
            apagar -= atual
            cedulas += 1
        else:
            contagem += [(cedulas,atual)]
            if apagar <= 1:
                break
            if atual == 50:
                atual = 20
            elif atual == 20:
                atual = 10
            elif atual == 10:
                atual = 5
            elif atual == 5:
                atual = 2
            cedulas = 0
    return contagem
```

Repetições Aninhadas

Podemos combinar mais de uma estrutura de repetição de forma a obter resultados interessantes.
Exemplo: Gerar as tabuadas de multiplicação de 1 a 10.

Repetições Aninhadas

Podemos combinar mais de uma estrutura de repetição de forma a obter resultados interessantes.
Exemplo: Gerar as tabuadas de multiplicação de 1 a 10.

```
# função tabuadas que gera as tabuadas
# de multiplicação de 1 a 10
# sem parâmetro → int
def tabuadas():
    pivo = 1
    lista = []
    while pivo <= 10:
        tabuada = []
        numero = 1
        while numero <= 10:
                tabuada += [str(pivo) + '*' + str(numero) + '=' + str(pivo*numero)]
                numero += 1
        pivo += 1
        lista.append(tabuada)
    return lista
```

Repetições Aninhadas

Podemos combinar mais de uma estrutura de repetição de forma a obter resultados interessantes.
Exemplo: Gerar as tabuadas de multiplicação de 1 a 10.

```
# função tabuadas que gera as tabuadas
# de multiplicação de 1 a 10
# sem parâmetro → int
def tabuadas():
    pivo = 1
    lista = []
    while pivo <= 10:
        tabuada = []
        numero = 1
        while numero <= 10:
                tabuada += [str(pivo) + '*' + str(numero) + '=' + str(pivo*numero)]
                numero += 1
        pivo += 1
        lista.append(tabuada)
    return lista
```

Exercício: Reescreva a função *tabuadas* usando for.

Autores

- **João C. P. da Silva** Lattes
- **Carla Delgado** Lattes
- **Ana Luisa Duboc** Lattes

Colaboradores

- **Fabio Mascarenhas** Lattes
- **Anamaria Martins Moreira** Lattes
- **Leonardo de Oliveira Carvalho** Lattes
- **Charles Figueiredo de Barros** Lattes
- **Fabrício Firmino de Faria** Lattes

Computação 1 - Python
Aula 9 - Teórica: Interferindo no fluxo de repetição: Break e Continue
Laços Aninhados