



INSTITUTO
FEDERAL
Alagoas

ATLETAS DA PROGRAMAÇÃO

Aula 3



MONDAY

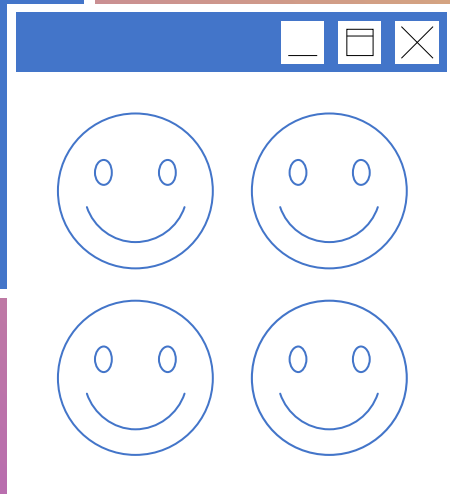
OK

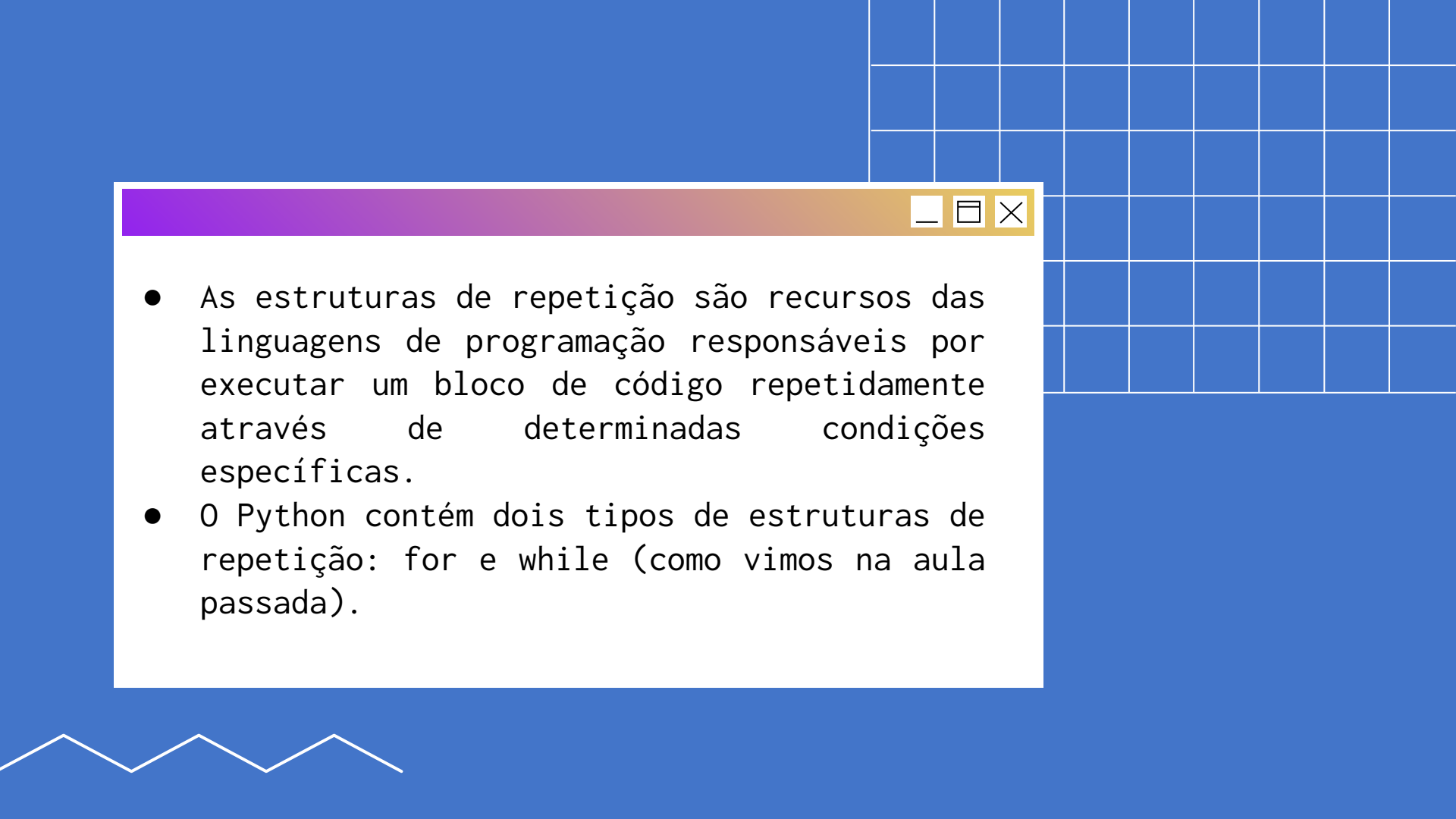
ASSUNTOS DE HOJE



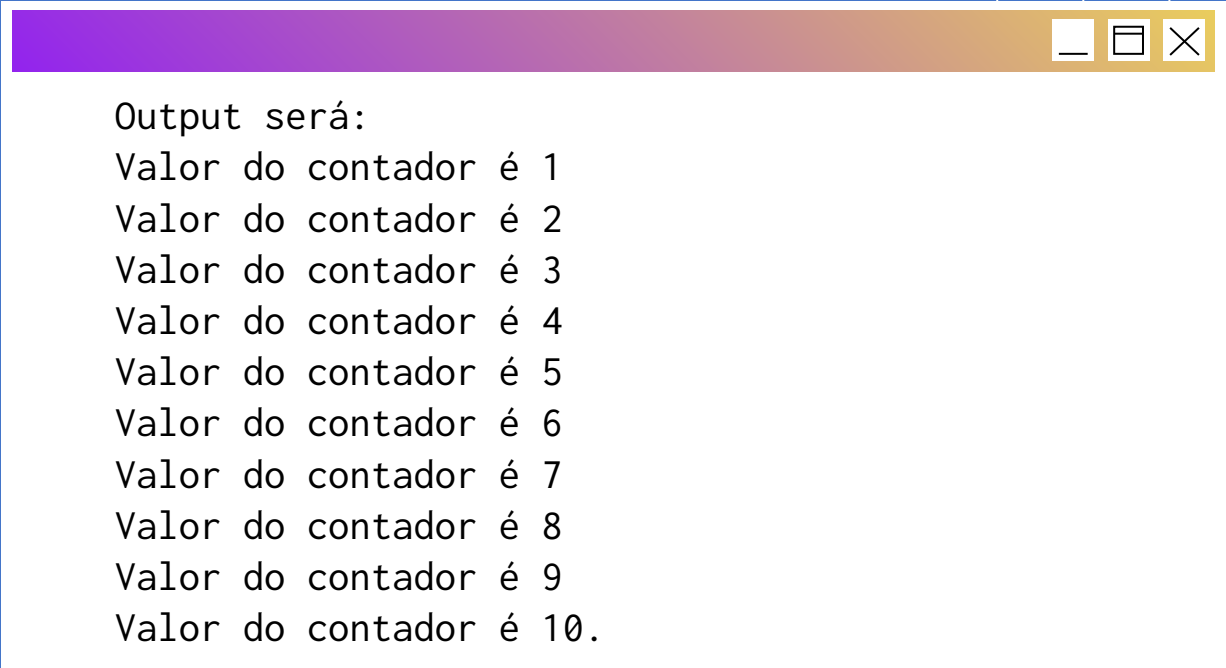
- Continuação de laços;
- Revisão de loops;
- Usando o for com range sem contar de um em um;
- While: True e Break;
- Usando o **pass** no while e no for;
- Usando o **continue** no while e no for;
- Prática com vários exercícios para fixação;

CONTINUAÇÃO DE LAÇOS



- 
- As estruturas de repetição são recursos das linguagens de programação responsáveis por executar um bloco de código repetidamente através de determinadas condições específicas.
 - O Python contém dois tipos de estruturas de repetição: for e while (como vimos na aula passada).

- **Exemplo de for:** `for i in range(0, 15, 2)`
`print(i, end=" ")` O output será: 0, 2, 4, 6, 8, 10, 12, 14
- **Exemplo de while:** `contador = 0` while
`contador < 10: print('Valor do contador é',`
`contador)` `contador = contador + 1`
P.s: output no próximo slide.



Output será:

Valor do contador é 1

Valor do contador é 2

Valor do contador é 3

Valor do contador é 4

Valor do contador é 5

Valor do contador é 6

Valor do contador é 7

Valor do contador é 8

Valor do contador é 9

Valor do contador é 10.

AUXILIADORES

Existem 3 comandos que nos auxiliam quando queremos alterar o fluxo de uma estrutura de repetição.

São eles: **break**, **continue** e **pass**.

AUXILIADORES: BREAK

É usado para finalizar um loop, isto é, é usado para parar sua execução. Geralmente vem acompanhado de alguma condição para isso, com um if.

Veja um exemplo em for no próximo slide:

AUXILIADORES: BREAK

```
for num in range(10):  
    # Se o número for igual a 5, devemos parar o loop  
    if num == 5:  
        # Break faz o loop finalizar  
        break  
    else: print(num)
```

AUXILIADORES: BREAK

Output:

0

1

2

3

4

AUXILIADORES: BREAK

Já com `while`, também podemos utilizar o `break` em uma condição utilizando `if`, assim:

```
num = 0

while num < 5:
    num = num + 1
    if num == 3:
        break
    print(num)
```

AUXILIADORES: BREAK

Quando a variável atribuir o valor 4 o laço é finalizado pelo break, encerrando o loop.

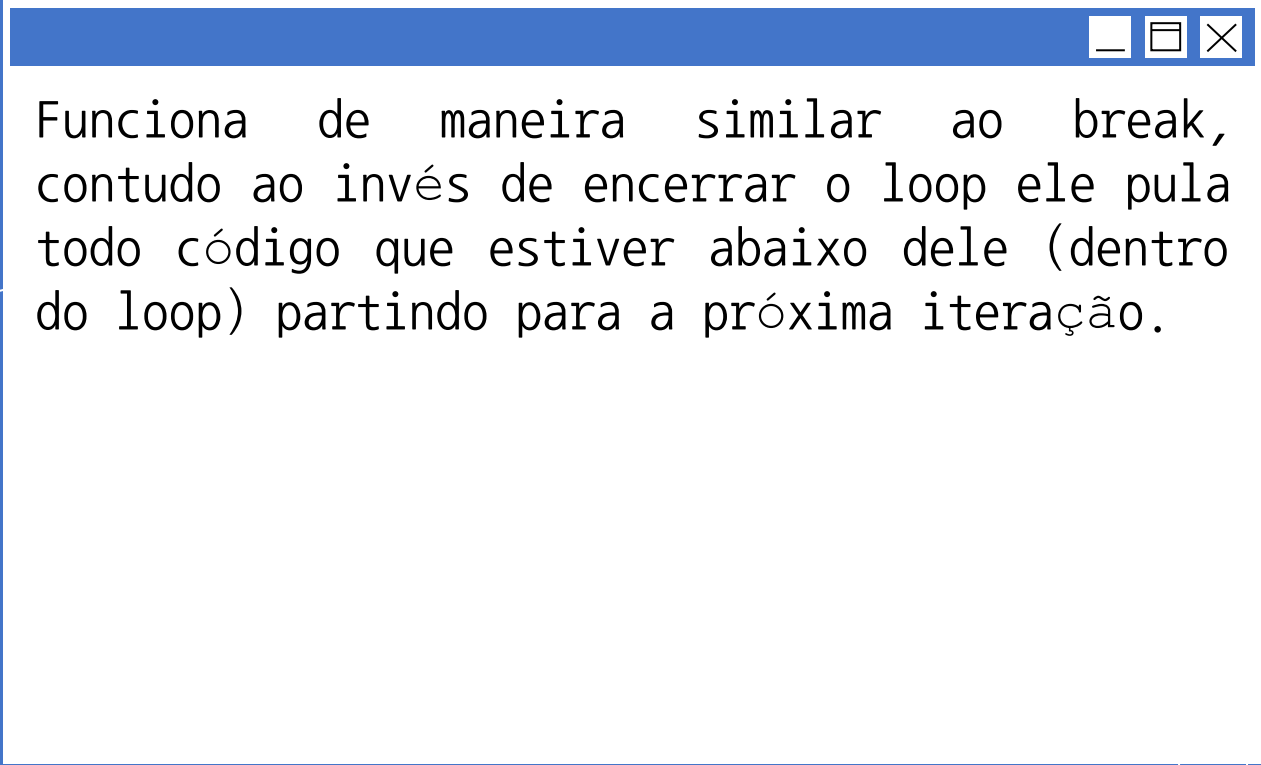
Output:

1

2

3

AUXILIADORES: CONTINUE



Funciona de maneira similar ao break, contudo ao invés de encerrar o loop ele pula todo código que estiver abaixo dele (dentro do loop) partindo para a próxima iteração.

AUXILIADORES: CONTINUE

EXEMPLO COM FOR

```
for num in range(5):  
    if num == 3:  
        print("Encontrei o 3")  
        # Executa o continue, pulando para  
        # o próximo laço continue  
    else: print(num)  
print("Estou abaixo do IF")
```

OUTPUT

```
0  
Estou abaixo do IF  
1  
Estou abaixo do IF  
2  
Estou abaixo do IF  
Encontrei o 3  
4
```

ATENÇÃO: Repare no output anterior que quando a condição `num == 3` for satisfeita, a string "Estou abaixo do IF" não será exibida.



AUXILIADORES: CONTINUE

EXEMPLO COM WHILE

```
num = 0
while num < 5:
    num += 1
    if num == 3:
        continue
    print(num)
```

OUTPUT

No resultado desse código o 3 não deve aparecer, pois o `print()` que imprime os números está abaixo do `continue`.

1

2

4

5

AUXILIADORES: PASS

O **pass** nada mais é que uma forma de fazer um código que não realiza operação nenhuma.

Como os escopos de Classes, Funções, If/Else e loops for/while são definidos pela indentação do código (e não por chaves {} como geralmente se vê em outras linguagens de programação), usamos o pass para dizer ao Python que o bloco de código está vazio.

Veja alguns exemplos no próximo slide:

Caso não utilizemos o pass, veja o que acontece:

```
class Classe:  
  
    def funcao():  
  
        pass
```

Output:

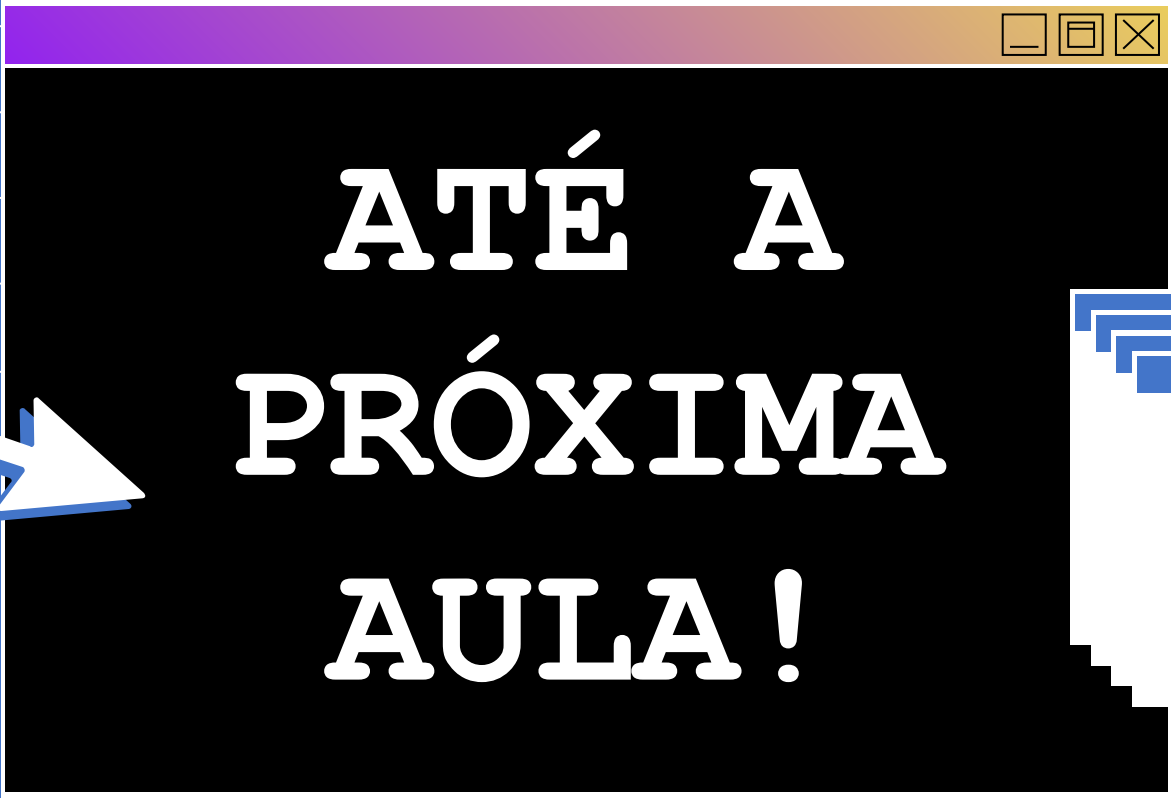
```
File "", line 2
```

```
^
```

```
IndentationError: expected an indented block
```

Isso acontece pois o Python entende que as próximas linhas de código fazem parte do mesmo escopo, mas como não estão indentadas um erro `IndentationError` é lançado.





ATÉ A
PRÓXIMA
AULA!



MONDAY