Linguagem de Programação Python

Repetição (`while`)

Professor: Ritomar Torquato

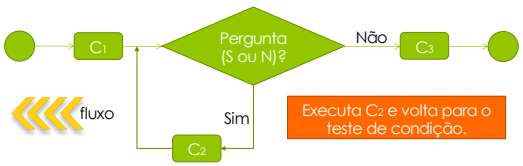
1

Repetição (`while`)

Objetivos: Repetição com final indeterminado; Repetição com teste no início e no final.

2

Estrutura de Repetição ou Iteração



3

Estrutura de Repetição ou Iteração

- Repetição com **Final Indeterminado**

x <> 0

x < 10

4

Estrutura de Repetição ou Iteração

- Repetição com **Final Indeterminado**

A quantidade de repetições do bloco só será conhecida após seu fim.

`while`

- com **Teste no Início**
- com Teste no Final

5

Estrutura de Repetição ou Iteração

- Repetição com **Final Indeterminado**
  - com **Teste no Início:** Faz um teste antes de executar o bloco de comandos.

`while`

O bloco de comandos poderá nunca ser executado.

6

Estrutura de Repetição ou Iteração

- **Final Indeterminado**: teste no início

➔Enquanto tiver saúde e dinheiro, vou **desfrutar a vida**.
O quê permitirá desfrutar a vida (**a ação**) é ter saúde e dinheiro (a condição). A condição é testada antes de executar a ação, podendo nunca ser satisfeita e **a ação pode nunca ser executada**.

7

exemplo

```
from random import randrange, seed
seed()

def tem_saude_e_dinheiro():
    return bool(randrange(10))

def main():
    while tem_saude_e_dinheiro():
        print('Aproveite a vida...')

    print('Não tem mais saúde ou dinheiro!')

if __name__ == '__main__':
    main()
```

8

exemplo

- Escreva um programa que leia um número inteiro e mostre quantas vezes esses é possível fazer a divisão inteira desse número por 2.
  - Por exemplo, é possível fazer a divisão inteira de 100 por 2 até 7 vezes: 50, 25, 12, 6, 3, 1, 0.

9

exemplo

```
def divisoes_por_2(n):
    quantidade = 0
    while n > 0:
        n //= 2
        quantidade += 1
    return quantidade

def main():
    n = int(input('Digite um número inteiro: '))
    qtd = divisoes_por_2(n)

    print(f'É possível dividir {n} por 2 até {qtd} vezes.')

if __name__ == '__main__':
    main()
```

quantidade é uma variável auxiliar que usamos para contar a quantidade de vezes que fazemos a divisão.

Como não sabemos a quantidade de vezes que faremos a divisão usamos o comando enquanto com Teste no Início.

10

exemplo

while

No **início** da repetição, não se sabe quantas vezes o bloco de comandos será repetido.

Quantos dígitos existem em um número inteiro positivo?

268 tem 3 dígitos
5697 tem 4 dígitos
25794586 tem 8 dígitos

O programa deve funcionar para qualquer inteiro. Mas não sabemos qual será o número.

11

exemplo

```
def conta_digitos(n):
    quantidade = 0
    while n > 0:
        quantidade += 1
        n //= 10
    return quantidade

def main():
    n = int(input("Número inteiro positivo: "))
    q = conta_digitos(n)
    print(f"{n} tem {q} dígitos")

if __name__ == '__main__':
    main()
```

12

Estrutura de Repetição ou Iteração

- Repetição com **Final Indeterminado**

A quantidade de repetições do bloco só será conhecida após seu fim.

while

- com Teste no Início
- com **Teste no Final**

13

Estrutura de Repetição ou Iteração

- Outras linguagens, tem um comando próprio para repetição com **Teste no Final**

Pascal

```
repeat
    <comando 1>
    <comando 2>
    ...
    <comando n>
until condicao;
```

C, C++, Java...

```
do{
    <comando 1>
    <comando 2>
    ...
    <comando n>
}while(condicao);
```

14

Estrutura de Repetição ou Iteração

- Em Python, o **while** é usado tanto para repetição com teste no início como na repetição com teste final.

```
while True:
    <comando 1>
    <comando 2>
    ...
    <comando n>
    if condicao: break
```

15

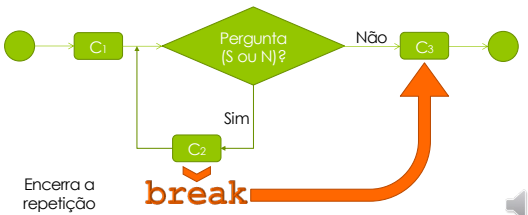
Desvio Incondicional

- Quebra o fluxo normal de execução da repetição
  - break
  - continue

16

break

Desvia incondicionalmente o fluxo de execução para o próximo comando após a repetição.



17

exemplo

No código abaixo encerramos a execução do **for** imediatamente após o contador **i** alcançar o valor "3". Como resultado serão impressos apenas "0 1 2", em lugar de "0 1 2 3 4".

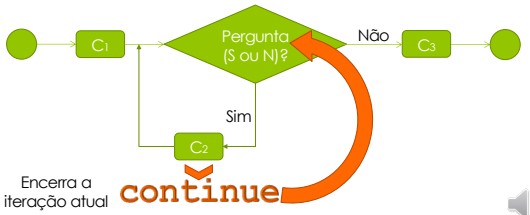
```
for i in range(5):
    if (i == 3): break
    print(f'{i}', end = ' ')
```

Troca a quebra de linha por um espaço

18

continue

Desvia incondicionalmente o fluxo de execução para a próxima iteração avaliando a pergunta da repetição.



19

exemplo

No código abaixo saltamos para a próxima iteração quando o contador `i` alcançar o valor "3". Como resultado o código não imprimirá esse valor, apenas "0 1 2 4"..

```
for i in range(5):
    if (i == 3): continue
    print(f'{i}', end = ' ')
```

Troca a quebra de linha por um espaço

20

Estrutura de Repetição ou Iteração

- Repetição com **Final Indeterminado**
  - com **Teste no Final:** Testa somente após executar o bloco de comandos.

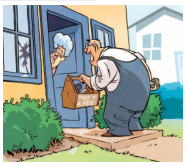
while

O bloco de comandos é executado ao menos UMA vez.

21

Estrutura de Repetição ou Iteração

- **Final Indeterminado:** teste no final



➔ Vou **atirar pedras** na vidraça até quebrar.
O que fará quebrar a vidraça (a condição) é atirar a pedra (**a ação**). Não faz sentido testar a condição antes de executar a ação, então, **a ação deve ser executada ao menos uma vez.**

22

exemplo

```
from random import randrange, seed
seed()

def vidraca_quebrada():
    return not bool(randrange(10))

def main():
    while True:
        print('Atirar pedra na janela.')
        if vidraca_quebrada(): break

    print('Janela quebrada')

if __name__ == '__main__':
    main()
```

23

exemplo

- Escreva um programa que leia números pelo teclado e mostra na tela o dobro de cada número lido. A execução termina com a leitura do número zero (**flag**).

flag?

24

Entendendo uma Flag

Flag significa bandeira

25

Em corridas de carros, por exemplo, ela sinaliza uma informação ao piloto que está (repetidamente) dando voltas na pista.

A bandeira passa uma informação

26

No futebol também temos as bandeiras.

A bandeira passa uma informação

27

Flag é um sinalizador.

Em programação, é um interruptor (isto é, valores como 1/0, ligado/desligado, ativo/inativo, continua/para, ...) e permite ativar ou desativar determinado comportamento.

Um valor esperado em determinada variável.

É comum o uso de flag em repetições com final indeterminado para definir se o processamento deve continuar ou ser finalizado.

28

exemplo

- Escreva um programa que leia números pelo teclado e mostra na tela o dobro de cada número lido. A execução termina com a leitura do número zero (**flag**).

flag?

O programa observa uma variável que é o **sinalizador** (flag). Quando a variável assumir o valor esperado uma ação é executada (final da repetição).

29

exemplo

```
def dobro(n):
    return n * 2

def main():
    while True:
        n = int(input("Digite um número: "))
        if n != 0:
            print(dobro(n))
        if n == 0: break

if __name__ == '__main__':
    main()
```

A **flag** é a variável n assumir o valor zero. Se essa condição for satisfeita a execução deve ser encerrada.

30