

FUNDAMENTOS DE PROGRAMAÇÃO

Estruturas de repetição
Prof. Bruno Góis Mateus



Índice

- Introdução
- Laços condicionais
- Laços contados
- Python Turtle
- Controlando fluxo

Introdução

- Até agora vimos como escrever programas capazes de executar comandos de forma linear
- Se necessário, tomar decisões com relação a executar ou não um bloco de comandos.
- Entretanto, eventualmente é necessário executar um bloco de comandos várias vezes para obter o resultado esperado.

Introdução

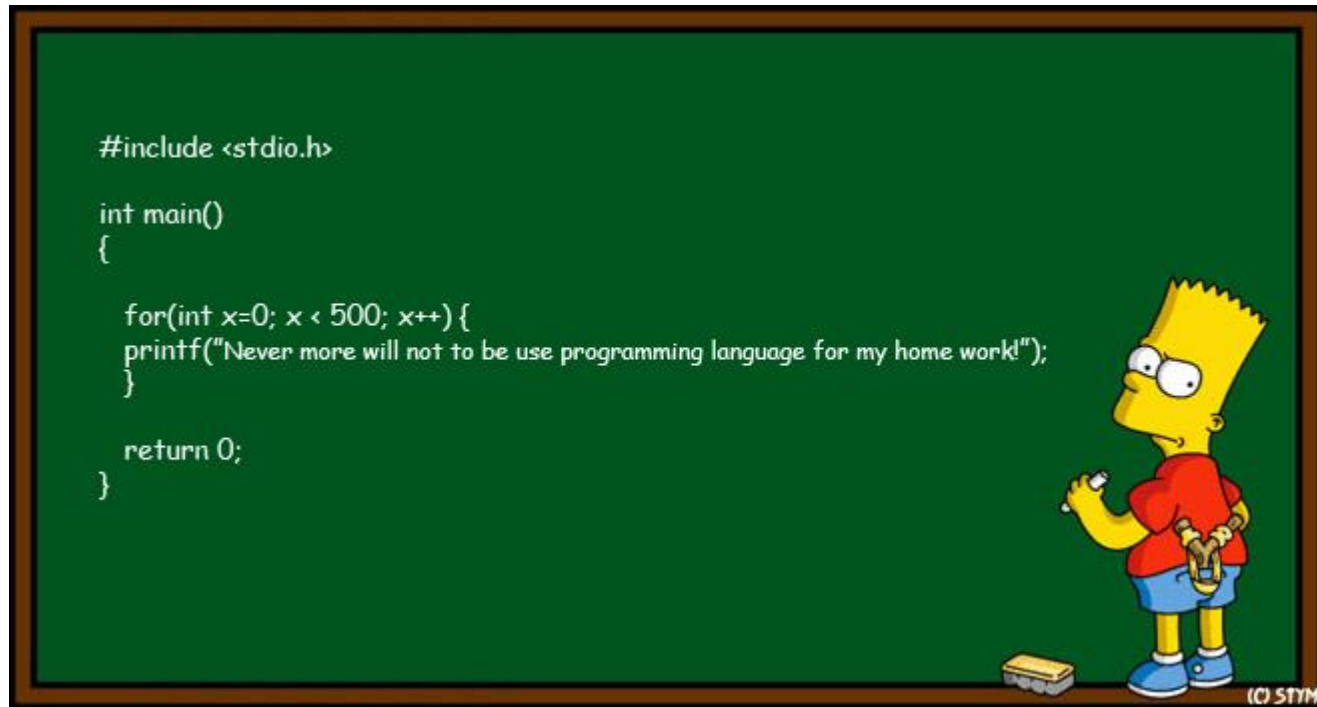
- Escreva um algoritmo para imprimir todos os números de 1 até 5
- Escreva um algoritmo para imprimir todos os números de 1 até 10
- Escreva um algoritmo para imprimir todos os números de 1 até 100

Introdução

The image consists of a large, repeating grid of the text "I will lock my computer when I leave my desk." The text is arranged in a 10x10 grid pattern, with each row and column containing the same sentence. The text is in a simple, black, sans-serif font. In the bottom right corner of the grid, there is a small, colorful illustration of Bart Simpson, a character from the animated series "The Simpsons". Bart is depicted from the waist up, wearing his signature red shirt and blue shorts, and has a mischievous expression on his face. The overall image has a light gray background.



Introdução



Introdução

- São muito comuns situações em que se deseja repetir um determinado trecho de código várias vezes
- As estruturas de repetição são muitas vezes chamados de **laços** ou **loops**
- O **corpo** da estrutura é o(s) comando(s) cuja execução serão repetidas

Introdução

- Laços condicionais
- Laços contados

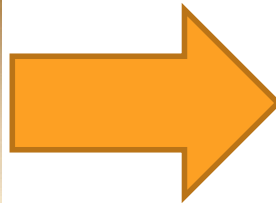
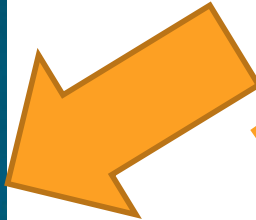
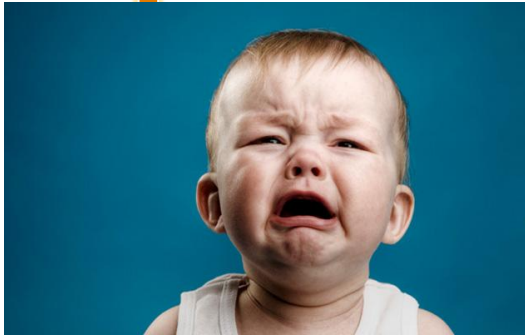
LAÇOS CONDICIONAIS

Laços condicionais

- Também conhecidos com laços indefinidos
- Não se conhece previamente o número necessário de repetições
- A quantidade de repetição é realizada enquanto uma condição for satisfeita

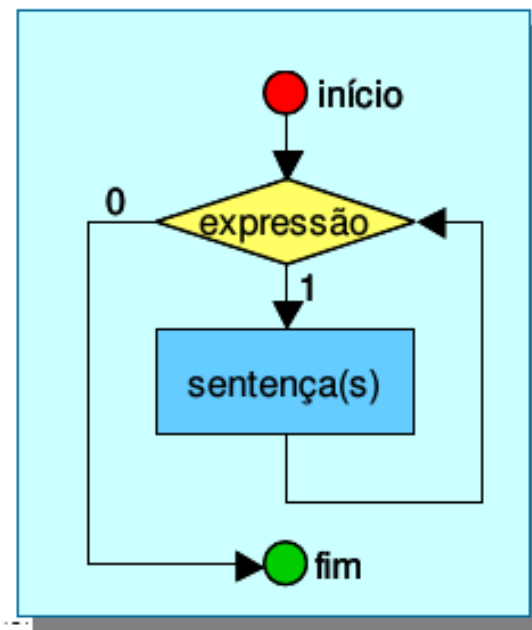
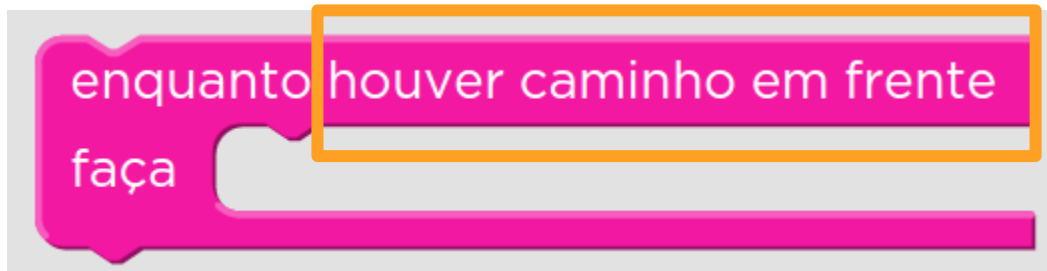
Laços condicionais – exemplo do cotidiano

Enquanto



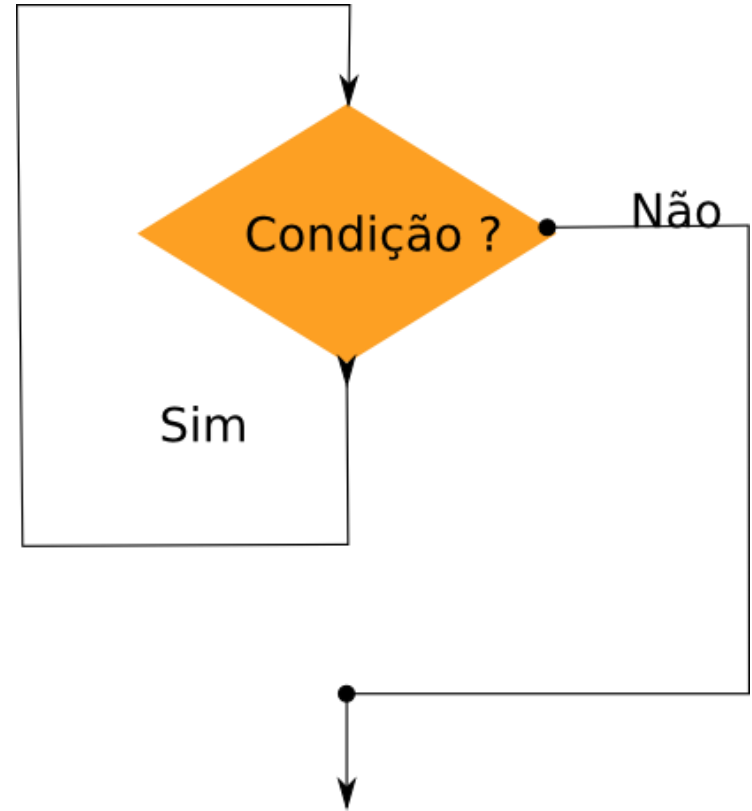
Laços condicionais

- Executa os comandos enquanto a condição é verdadeira
- Condição é verificada **antes** do bloco



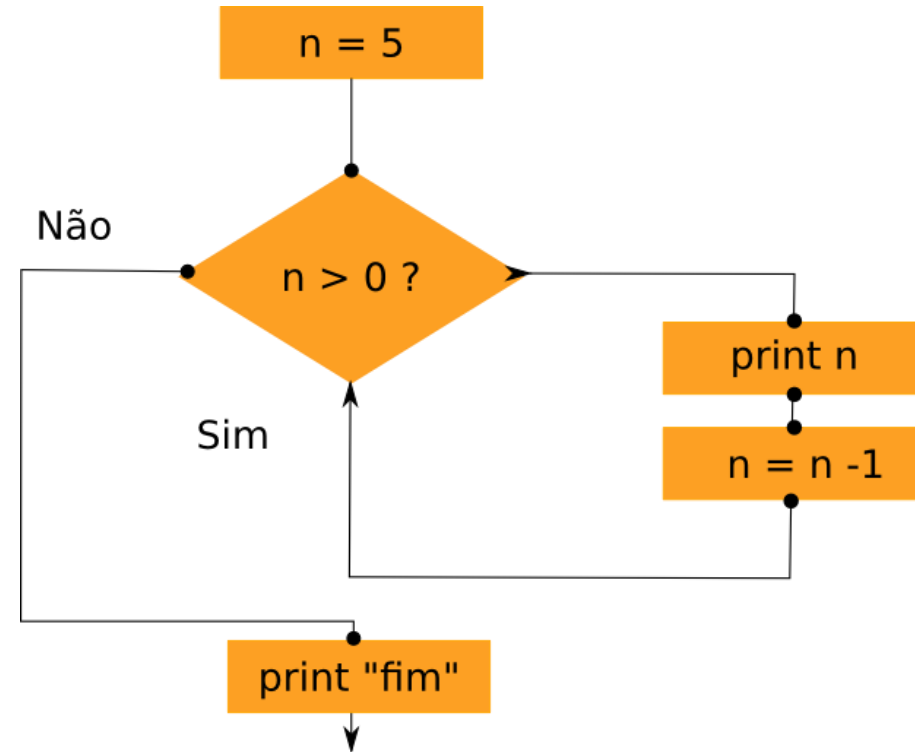
while

- Similar ao if
- Quem controla a execução do corpo é a condição
- O corpo será executado enquanto a condição for verdadeira
- Variáveis de iteração
 - São alteradas a cada execução do laço
 - Em geral são números



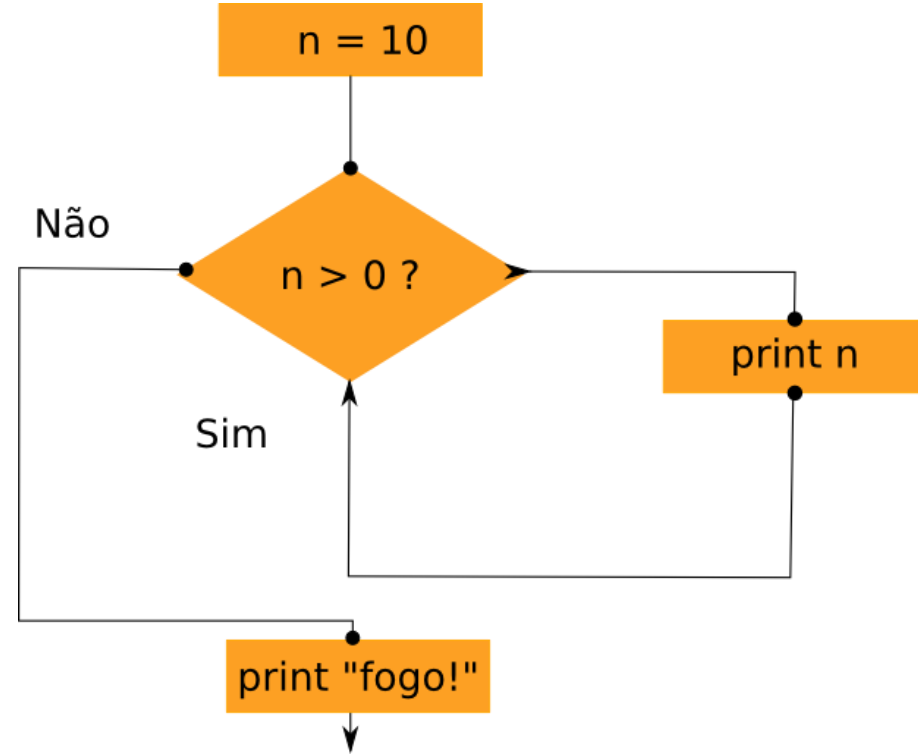
Exemplo - while

```
n = 5
while n > 0 :
    print n
    n = n - 1
print 'fim'
```



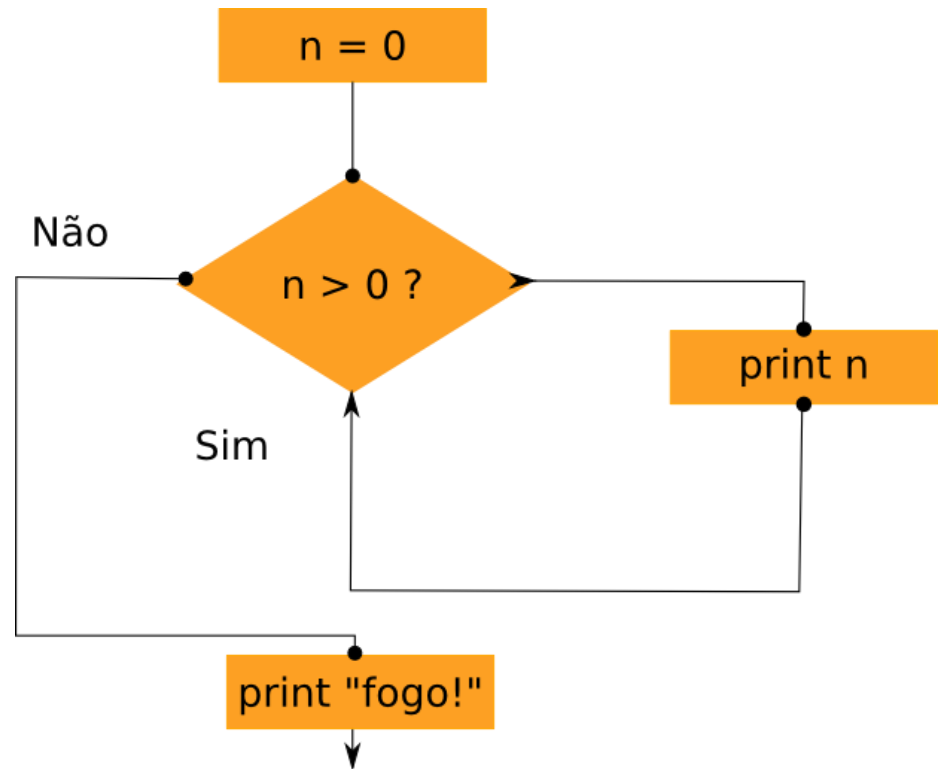
Exemplo - while

```
n = 10
while n > 0 :
    print n
print 'fogo!'
```



Exemplo - while

```
n = 0  
while n > 0 :  
    print n  
print 'fogo!'
```



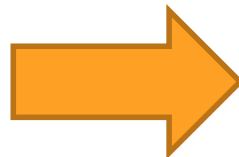
LAÇOS CONTADOS

Laços contados

- Também conhecidos como laços definidos
- Quando temos conhecimento prévio de quantas vezes o corpo deve ser executado
 - Lista de tarefas (quantidade)
 - Conjunto finito

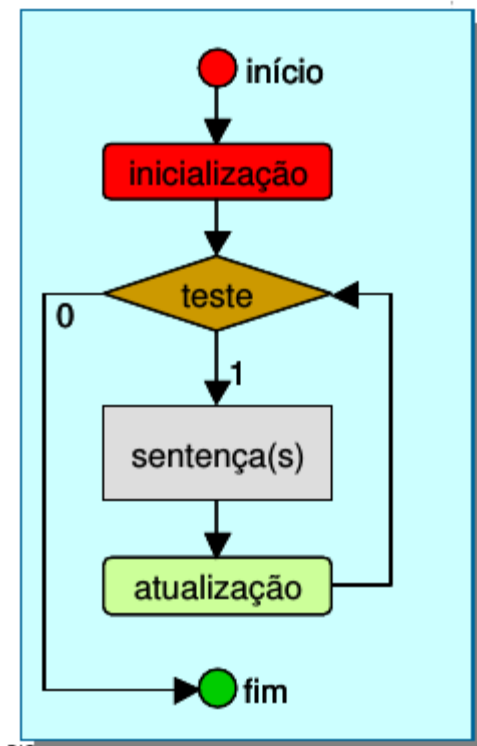
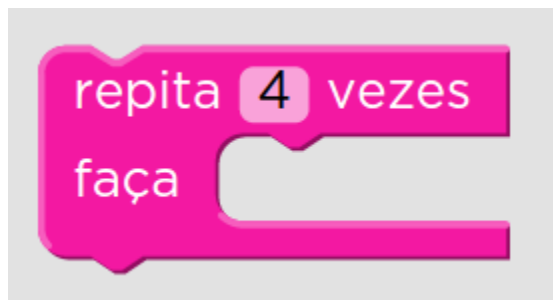
Laços contados

Repita de 1 até 3



Laços contados

- Verifica a condição (contador)
- Executa os comandos



Exemplo - for

```
for i in [5, 4, 3, 2, 1] :  
    print i  
print 'fim!'
```

- Esse tipo de laço possui uma variável de iteração **explícitas**
 - A sequência do conjunto dado

Exemplo - for


```
friends = ['Fulano', 'Cicrano', 'Cirano']  
for friend in friends :  
    print 'Feliz ano novo:', friend  
print 'valeu!'
```

for

- A variável de iteração itera(percorre) uma sequência(conjunto ordenado)
- O corpo do laço é executado uma única vez para cada valor do conjunto
- O valor da variável de iteração é alterado de acordo com o valores contidos no conjunto

Variável de
iteração

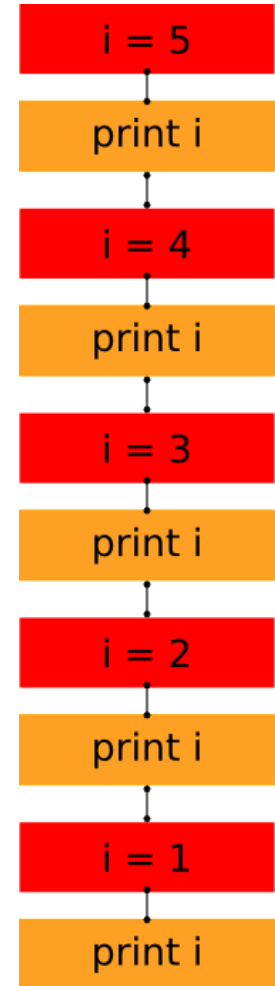
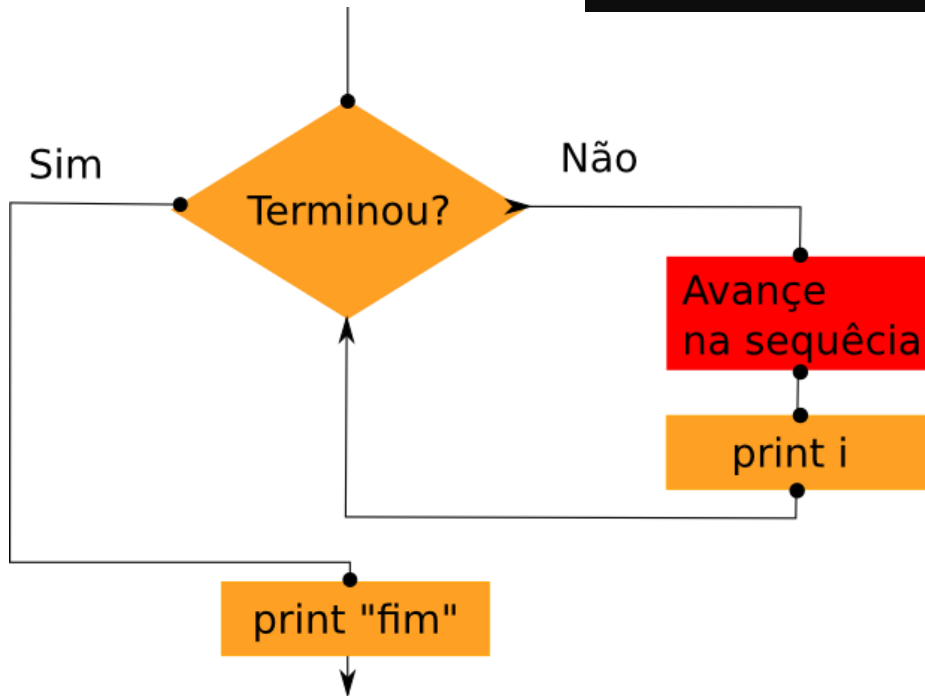
Conjunto com 5
elementos



```
for i in [5, 4, 3, 2, 1] :  
    print i  
print 'fim!'
```

for

```
for i in [5, 4, 3, 2, 1] :  
    print i
```



PYTHON TURTLE

Python Turtle

- Turtle é um módulo Python que oferece funcionalidades para fazermos desenhos na tela, com comandos muito simples.
 - Segue a idéia da linguagem de programação Logo
 - Muito utilizada em escolas como apoio ao ensino de disciplinas regulares
 - Também para introdução a programação para crianças.
 - A linguagem Logo segue a ideia de um robô (representado na tela por uma tartaruga) que o usuário pode controlar através de comandos simples de movimentação.

Python Turtle

- Primeiro programa

```
import turtle           # allows us to use the turtles library
wn = turtle.Screen()    # creates a graphics window
alex = turtle.Turtle()  # create a turtle named alex
```

Python Turtle

- Movendo a tartaruga
 - `forward()` ou `fd()`
 - Move a tartaruga para frente
 - `backward()` ou `bk()`
 - Move a tartaruga para trás
 - `left()` ou `lt()`
 - Gira a tartaruga para esquerda
 - `right()` ou `rt()`
 - Gira a tartaruga para a direita
 - `home()`
 - Move a tartaruga para a posição (0, 0)



Python Turtle

- Controle da caneta
 - `pendown()` ou `down()`
 - `penup()` ou `up()`
 - `pensize()` ou `width()`
 - Alterar a largura do traço
 - `pencolor()`
 - Define a cor do traço
 - `fillcolor()`
 - Define a cor da tartaruga



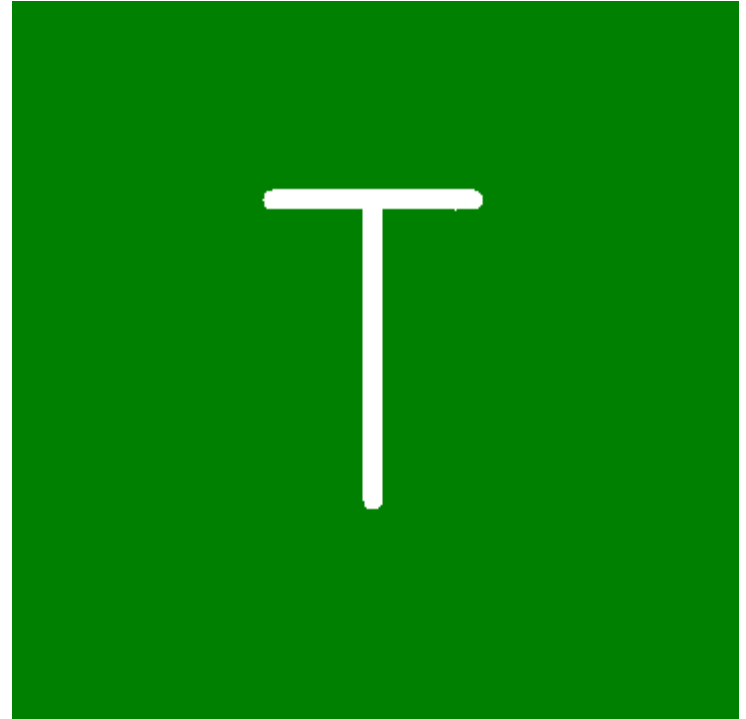
Python Turtle

- Alterando a aparência da tartaruga
 - `shape()`
 - “arrow”, “turtle”, “circle”, “square”, “triangle”, “classic”



Exemplo

```
import turtle
wn = turtle.Screen()
wn.bgcolor("green")
jamal = turtle.Turtle()
jamal.color("white")
jamal.pensize(10)
jamal.left(90)
jamal.forward(150)
jamal.left(90)
jamal.forward(50)
jamal.right(180)
jamal.forward(100)
wn.exitonclick()
```

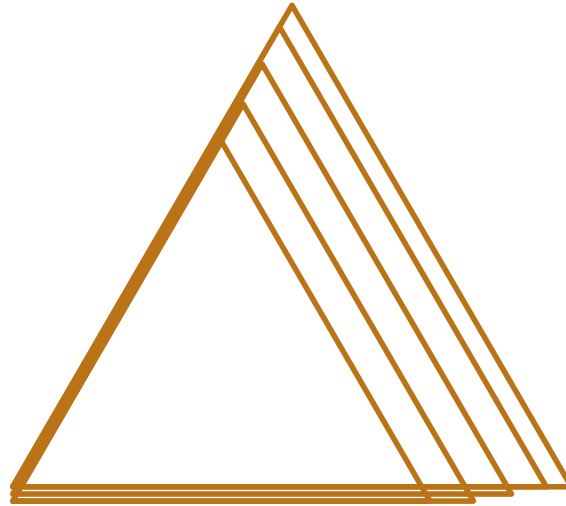


Hora da prática

- Escreva um programa para desenhar um triângulo
- Escreva um programa para desenhar um quadrado
- Escreva um programa para desenhar um hexágono
- Escreva um programa para desenhar um octágono

Hora da prática

- Escreva um programa que desenha N triângulos, um dentro outro, onde N é informado pelo usuário.



Hora da prática

- Escreva um programa para desenhar uma estrela de cinco lados
- Escreva um programa para desenhar uma estrela de seis lados



Hora da prática

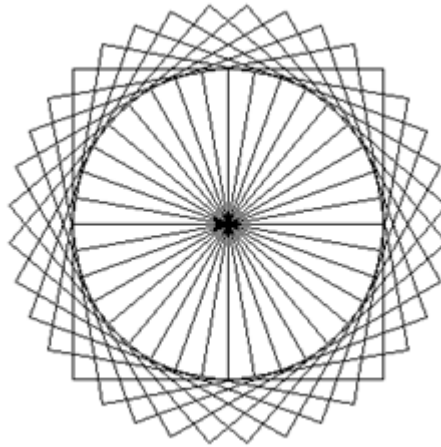
- Faça um programa para desenhar um círculo simulando um polígono de 360 lados
 - Dica: Cada lado tem $(2 \cdot \pi \cdot \text{raio}) / 360$
- Desenhar um boneco de neve usando círculos

Hora da prática

- Escreva um programa para desenhar a bandeira do Brasil

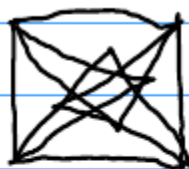
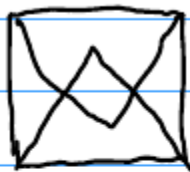
Hora da prática

- Faça um programa para desenhar a figura abaixo



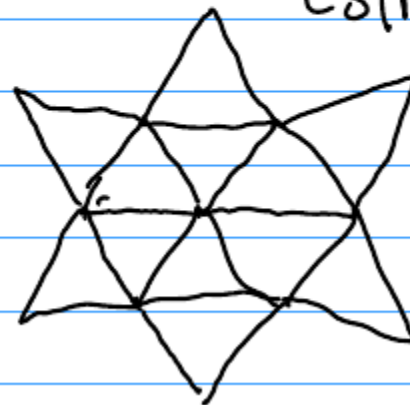
Hora da prática

4 triângulos



6 triâng

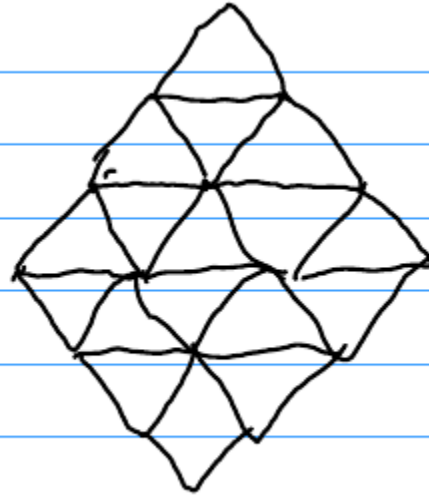
estrela



Hora da prática



Piramide



Losângulo

CONTROLANDO O FLUXO

Encontrado o maior elemento

3 41 12 9 74 15

Encontrando o maior elemento

60, 85, 24, 14, 26, 4, 88, 57, 63, 12, 100, 20, 23,
58, 83, 56, 11, 21, 25, 66, 33, 73, 8, 22, 27, 44,
29, 87, 84, 67, 41, 38, 48, 93, 47, 78, 92, 49, 76,
50, 62, 61, 34, 54, 71, 43, 96, 30, 55, 15, 9, 36,
86, 65, 3, 42, 97, 72, 18, 1

Encontrado o maior elemento

```
largest_so_far = -1
print 'Before', largest_so_far
for the_num in [9, 41, 12, 3, 74, 15] :
    if the_num > largest_so_far :
        largest_so_far = the_num
        print largest_so_far, the_num

print 'After', largest_so_far
```

Contando as iterações

```
zork = 0
print 'Before', zork
for thing in [9, 41, 12, 3, 74, 15] :
    zork = zork + 1
    print zork, thing
print 'After', zork
```

Acumulando

```
zork = 0
print 'Before', zork
for thing in [9, 41, 12, 3, 74, 15] :
    zork = zork + thing
    print zork, thing
print 'After', zork
```

Buscando por um elemento

```
found = False
print 'Before', found
for value in [9, 41, 12, 3, 74, 15] :
    if value == 3 :
        found = True
        print found, value
print 'After', found
```

Buscando por um elemento

```
found = False
print 'Before', found
for value in [9, 41, 12, 3, 74, 15] :
    if value == 3 :
        found = True
    print found, value
print 'After', found
```

```
found = False
print "Before", found
numbers = [9, 41, 12, 3, 74, 15]
i = len(numbers) - 1
while found != True and i >=0:
    if numbers[i] == 3:
        found = True
    print found, numbers[i]
    i = i - 1
print "After", found
```

Break

```
found = False
print 'Before', found
for value in [9, 41, 12, 3, 74, 15] :
    if value == 3 :
        found = True
        break
    print found, value
print 'After', found
```


Continue

```
while True:
    line = raw_input('> ')
    if line[0] == '#' :
        continue
    if line == 'done' :
        break
    print line
print 'Done!'
```

A seguir

- Funções