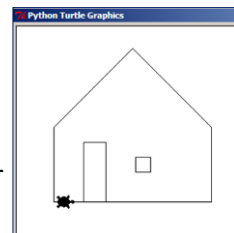


Módulo turtle (tartaruga)

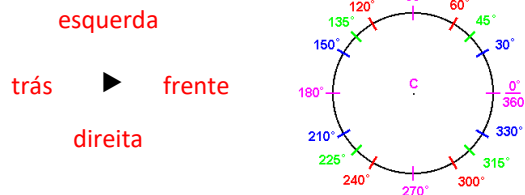
Turtle Robots (Robôs tartaruga)

- Uma classe de robôs (educacionais) que obedecem a uma sequência de comandos
- Origem: linguagem LOGO (Seymour Papert)
 - ✓ A tela (janela gráfica) é um plano cartesiano
 - ✓ A sequência de comandos movimenta o cursor
 - ✓ O cursor poderia ser qualquer coisa. No caso, é uma tartaruga com uma caneta na cauda, pré-programada para realizar algumas ações
 - ✓ O cursor pode, ou não, traçar sua trajetória
 - ✓ Pode-se ajustar suas características, tais como a cor e a largura da caneta; o formato da tartaruga e a velocidade que desenha; a cor de fundo da janela gráfica e etc.

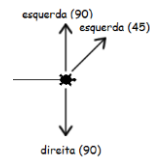


Desenhando com turtle

- ✓ A tartaruga pode se mover em todas as direções no plano xy.



- ✓ Ela se movimenta **uma distância** para frente ou para trás
- ✓ Ela pode se virar para a esquerda ou para a direita. Este movimento é medido em **graus**.



3

Usando o módulo da tartaruga

- 1) Importar o módulo

```
>>> import turtle
```

- 2) Criar um **objeto** tartaruga

```
>>> pat = turtle.Turtle()
```

Um objeto é capaz de realizar ações (métodos) e possui propriedades (atributos)

4

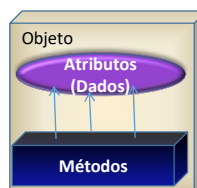
O que é um objeto?

- Um objeto é uma representação de qualquer coisa, real ou abstrata.
- Os objetos podem ser:
 - ✓ Físicos: um carro, um relógio, uma pessoa, um cachorro, um pássaro
 - ✓ Conceitos: uma matriz, uma data, um horário
 - ✓ Entidade de Software: **a turtle**, um arquivo, uma string, um inteiro, um float

5

Estrutura dos objetos

- Um objeto é composto por :
 - Atributos: itens de dados que descrevem suas características.
 - Métodos: conjunto de ações predefinidas (funções) que ele pode executar.

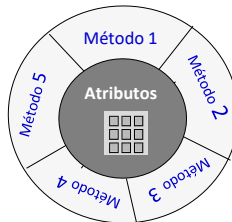


6

Comportamento

- O comportamento de um objeto é o que o objeto conhece e pode fazer.
- O comportamento de um objeto é traduzido em métodos (funções internas do objeto).

Modelo de Objeto

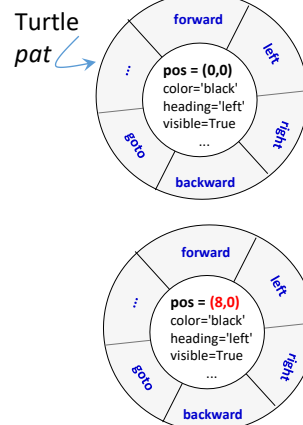


7

OO em Python

Programas utilizam os objetos em cálculos e/ou executando os métodos associados

```
import turtle  
  
pat=turtle.Turtle()  
  
pat.goto(8,0)
```



8

Usando a tartaruga criada

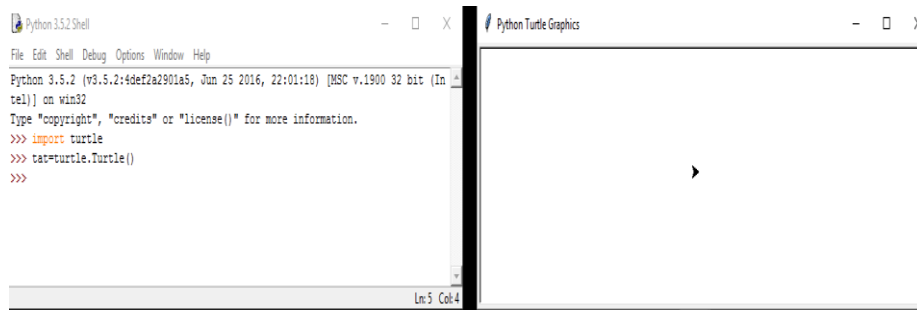
```
>>>pat = turtle.Turtle()
```

O objeto tartaruga criado é referenciado pela variável *pat*.

- ✓ Um objeto do tipo Turtle *nasce* sabendo realizar determinadas ações. Por exemplo, movimentar-se ou girar.
- ✓ O programador *invoca* as ações desejadas ativando os *métodos* definidos para o objeto.
- ✓ As *propriedades* (ou atributos) da tartaruga podem ser ajustadas.
 - Exemplos: forma, cor e espessura da caneta, direção, etc.

9

Janela da tartaruga



- A tartaruga atua em sua janela.
- Mantenha as duas janelas (da tartaruga e do interpretador) visíveis no monitor.

10

Ativando uma função

A tartaruga pode ser instruída a realizar alguma **ação** que ela conhece e sabe **responder** (comando). Isso deve ser feito do seguinte modo:

```
<nomedatataruga> . <ação>
```

11

Tartaruga em movimento

Movimentando a tartaruga

“Para frente distância n ”

```
tartaruga.forward(n) ou  
tartaruga.fd(n)
```

“Para trás distância n ”

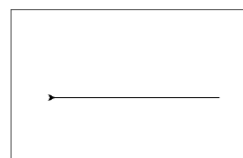
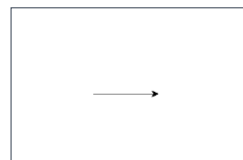
```
tartaruga.backward(n) ou  
tartaruga.bk(n)
```

Obs. n é um número real

13

Exemplos de movimentos

```
>>> import turtle  
>>> pat=turtle.Turtle()  
  
>>> pat.fd(100)  
  
>>> pat.bk(257.3)
```



14

Girando a tartaruga

Girar à direita x graus (x é um ângulo)

```
tartaruga.right(x)
```

```
tartaruga.rt(x)
```

Girar à esquerda x graus (x é um ângulo)

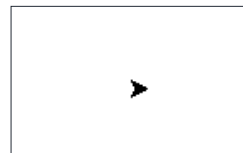
```
tartaruga.left(x)
```

```
tartaruga.lt(x)
```

15

Mais exemplos de movimentos

```
>>>pat.clear()  
#limpa a tela e mantém a posição e a direção
```



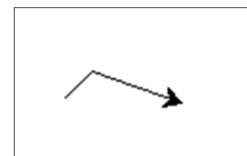
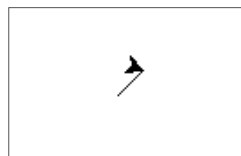
```
>>>pat.left(45)
```

```
>>>pat.fd(20)
```



```
>>>pat.right(65)
```

```
>>>pat.fd(50)
```



16

Levantando e abaixando a caneta

Quando a caneta está abaixada, o deslocamento deixa um rastro, isto é, risca uma linha.

Para riscar é preciso deslocar-se com a caneta abaixada:

```
tartaruga.down()
```

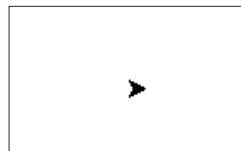
Para pular (deslocar-se sem riscar) é preciso antes, levantar a caneta

```
tartaruga.up()
```

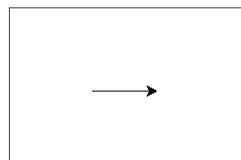
17

Rastros da tartaruga

```
>>>pat.clear()  
#limpa a tela e mantém a posição e a direção
```

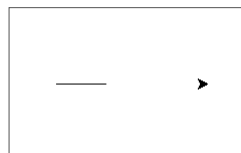


```
>>>pat.fd(100)
```



```
>>>pat.up()
```

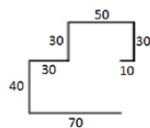
```
>>>pat.fd(100)
```



18

Editor e scripts grandes

Editor vs Modo Interativo



Este roteiro é bem grande!!!!
Se errar um comando da sequência,
tem de começar tudo de novo....

Se quiser que a *tat o* desenhe de novo, é preciso
reescrever toda a sequência

Editor e arquivos .py

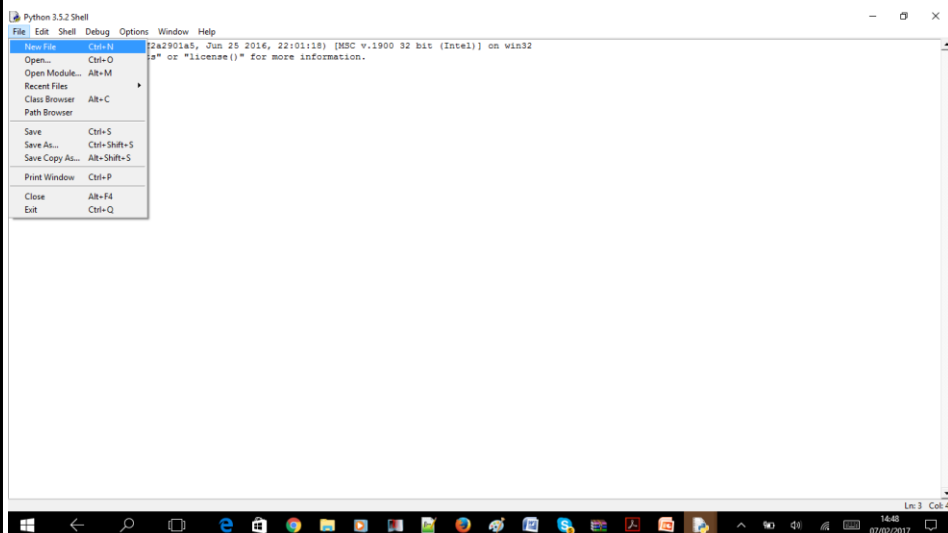
Scripts grandes, ou que se queira reusar, não devem ser digitados linha por linha no modo interativo.

Solução:

- ✓ Escrever os *scripts* em um arquivo com um editor de textos
- ✓ Salvar o arquivo como **.py**
- ✓ Executar o *script* salvo no arquivo no Interpretador Python

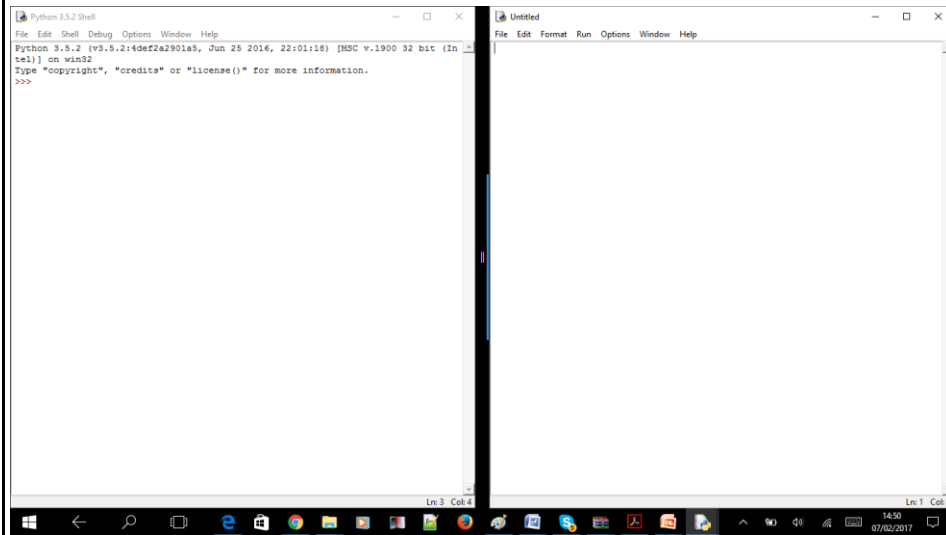
21

Editor IDLE (1/10)



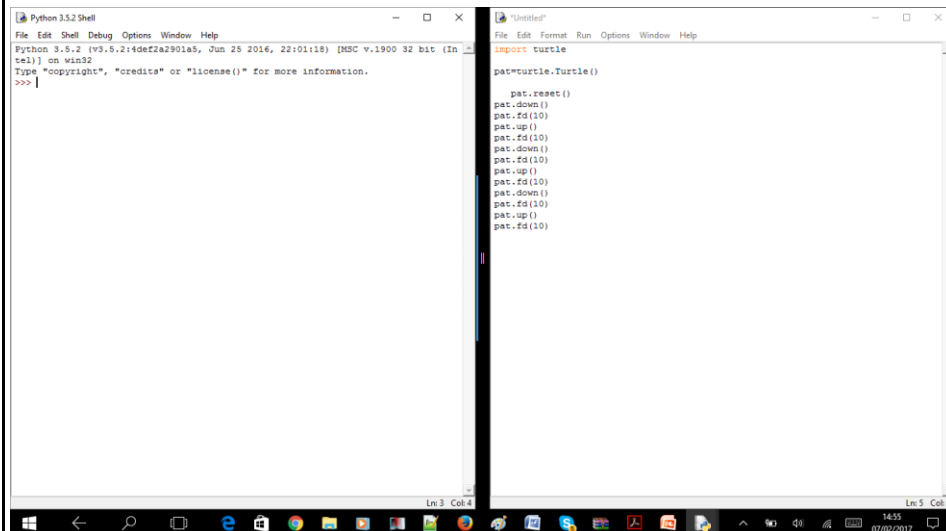
22

Editor IDLE (2/10)




23

Editor IDLE (3/10)



24



DEPARTAMENTO
DE INFORMÁTICA
PUC-RIO

Editor IDLE (4/10)

Python 3.5.2 Shell

File Edit Shell Debug Options Window Help

Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>

Untitled*


File Edit Format Run Options Window Help

Python Shell
Check Module Alt+X
Run Module F5
pat.reset()
pat.down()
pat.fd(10)
pat.up()
pat.fd(10)
pat.down()
pat.fd(10)
pat.up()
pat.fd(10)
pat.down()
pat.fd(10)
pat.up()
pat.fd(10)

Ln: 3 Col: 4

Ln: 5 Col: 3

25



DEPARTAMENTO
DE INFORMÁTICA
PUC-RIO

Editor IDLE (5/10)

Python 3.5.2 Shell

File Edit Shell Debug Options Window Help

Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>

Untitled*

File Edit Format Run Options Window Help

Import turtle
pat=turtle.Turtle()
pat.reset()
pat.down()
pat.fd(10)
pat.up()
pat.fd(10)
pat.down()
pat.fd(10)
pat.up()
pat.fd(10)
pat.down()
pat.fd(10)
pat.up()
pat.fd(10)

Ln: 3 Col: 4

Ln: 5 Col: 3

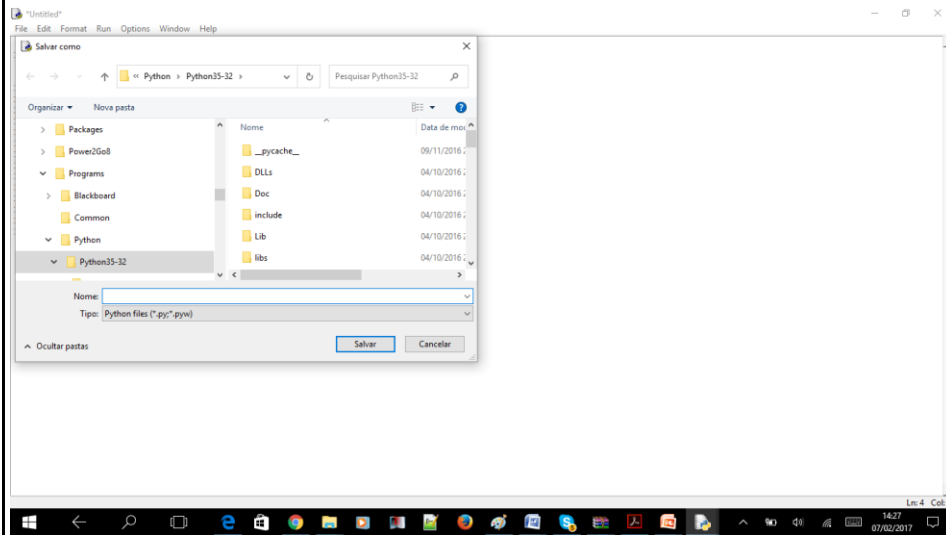
Save Before Run or Check

Source Must Be Saved
OK to Save?

OK Cancel

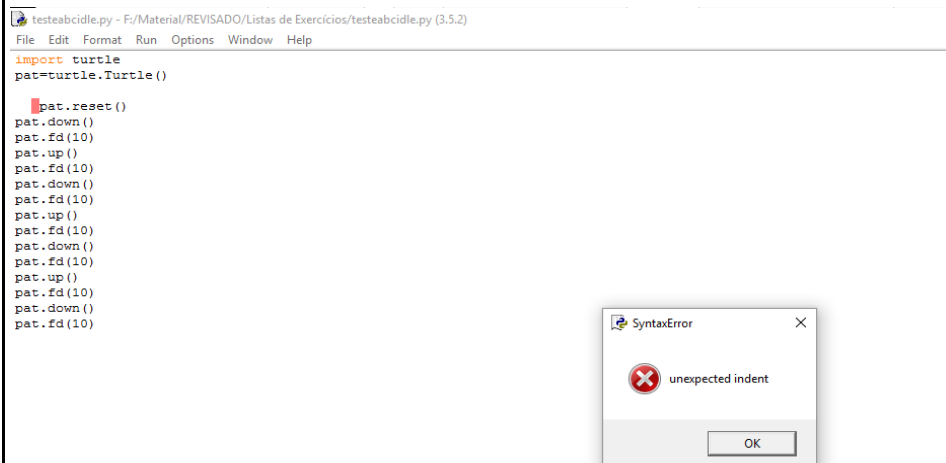
26

Editor IDLE (6/10)




27

Editor IDLE (7/10)



28



DEPARTAMENTO
DE INFORMÁTICA
PUC-RIO

Editor IDLE (8/10)

Python 3.5.2 Shell

File Edit Shell Debug Options Window Help


Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>

testeditor.py - C:/Users/claudia/fein/AppData/Local/Programs/Python/Python35-32/teste...

File Edit Format Run Options Window Help

Python Shell
Check Module Alt+X
Run Module F5
pat.reset()
pat.down()
pat.fd(10)
pat.up()
pat.fd(10)
pat.down()
pat.fd(10)
pat.up()
pat.fd(10)
pat.down()
pat.fd(10)
pat.up()
pat.fd(10)

29



DEPARTAMENTO
DE INFORMÁTICA
PUC-RIO

Editor IDLE (9/10)

Python 3.5.2 Shell

File Edit Shell Debug Options Window Help

Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>

testeditor.py - C:/Users/claudia/fein/AppData/Local/Programs/Python/Python35-32/teste...

File Edit Format Run Options Window Help

import turtle
pat=turtle.Turtle()
pat.reset()
pat.down()
pat.fd(10)
pat.up()
pat.fd(10)
pat.down()
pat.fd(10)
pat.up()
pat.fd(10)
pat.down()
pat.fd(10)
pat.up()
pat.fd(10)

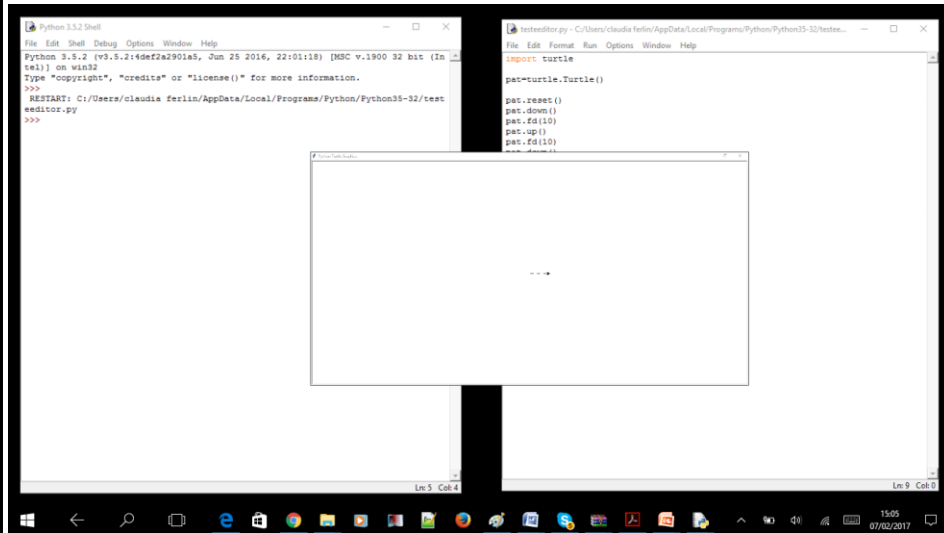
Save Before Run or Check

Source Must Be Saved
OK to Save?

OK Cancel

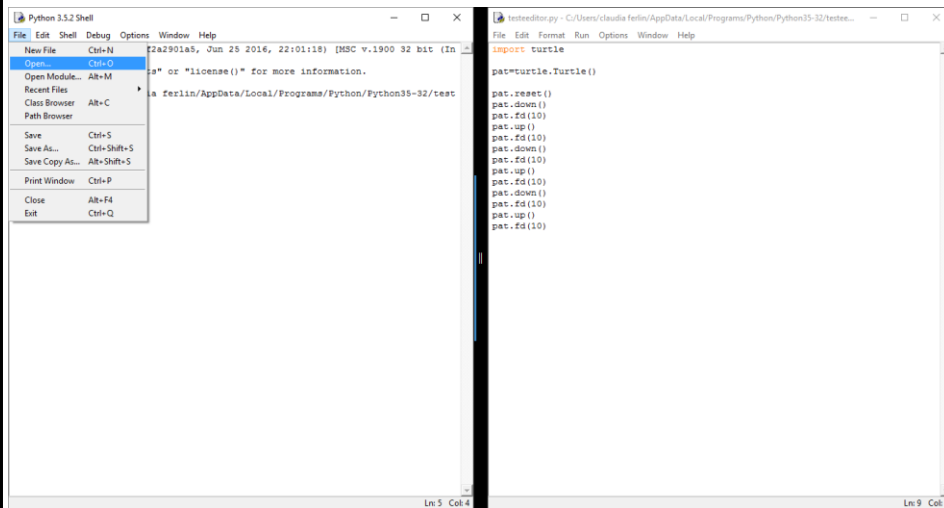
30

Editor IDLE (10/10)



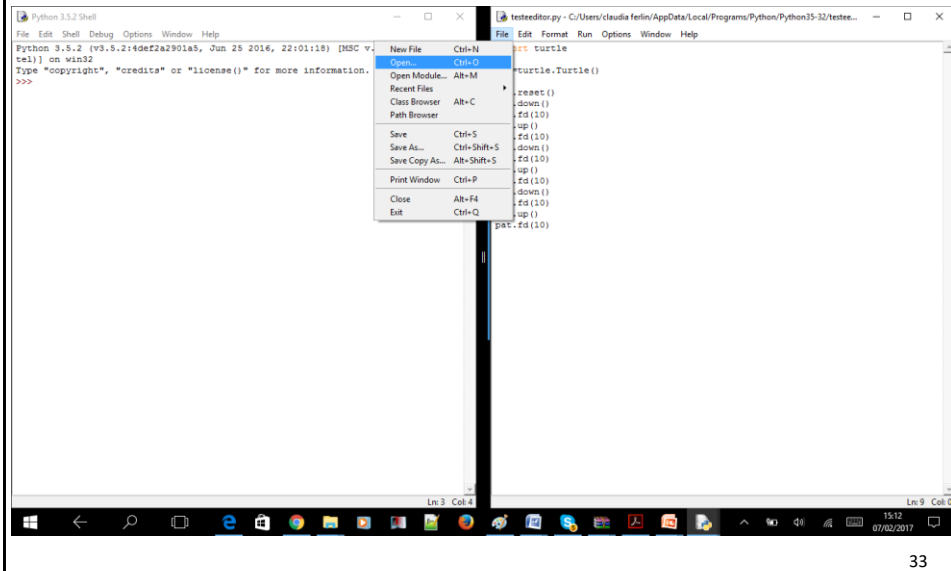
31

Editar arquivo existente (1/3)



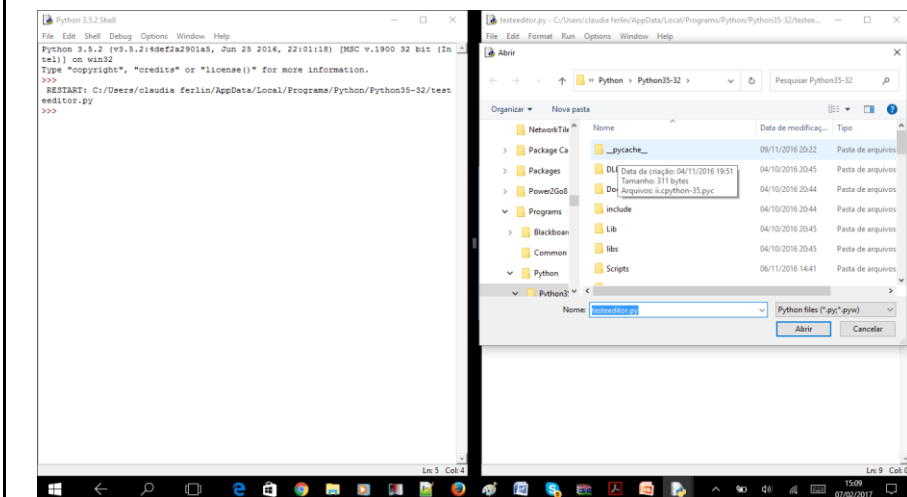
32

Editar arquivo existente (2/3)



33

Editar arquivo existente (3/3)



34

Mãos na massa no editor!!

Desenhar um linha com 6 segmentos de tamanho 100 e com duas cores intercaladas.



Novas instruções:

Mudar a largura do traço:

`tartaruga.width(valor)` em que *valor* é um nº positivo

Mudar a cor da Caneta:

`tartaruga.color(cor)` em que *cor* é um literal como 'red', 'green', 'blue',...

35

Linha colorida: uma solução

```
import turtle
pat = turtle.Turtle()
pat.reset()
pat.up()
pat.goto(-300,0)           # posiciona pat mais à esquerda
pat.down()
pat.width(10)
pat.color('green')
pat.fd(100)
pat.color('red')
pat.fd(100)
pat.color('green')
pat.fd(100)
pat.color('red')
pat.fd(100)
pat.color('green')
pat.fd(100)
pat.color('red')
pat.fd(100)
```



36

Turtle: métodos usuais

Método	Parâmetros	Descrição
forward	distância	Move para frente
backward	distância	Move para trás
right	ângulo	Vira no sentido horário
left	ângulo	Vira no sentido anti-horário
up	None	Levanta a cauda
down	None	Abaixa a cauda
color	cor	Muda a cor usada ao desenhar
fillcolor	cor	Muda a cor usada ao preencher um polígono
setheading	ângulo	Ajusta a orientação da tartaruga
position	None	Retorna a posição atual
goto	x,y	Move a tartaruga para a posição x,y
begin_fill	None	Use a posição atual para preencher polígono
end_fill	None	Termine o polígono na posição atual
dot	None	Deixe um ponto na posição atual
stamp	None	Deixe um carimbo da tartaruga na posição atual
undo	None	Desfaz a última operação
clear	None	Limpa a tela

37

Turtle e instruções adicionais

Desenhando um quadrado

Desenhar um quadrado de lado 100

O que deve ser feito?

39

Algoritmo do quadrado

1. Caminhar para frente 100 passos/pontos (deixando rastro)
2. Girar 90 graus para a esquerda (ou direita)
3. Caminhar para frente 100 passos/pontos (deixando rastro)
4. Girar 90 graus para a esquerda (ou direita)
5. Caminhar para frente 100 passos/pontos (deixando rastro)
6. Girar 90 graus para a esquerda (ou direita)
7. Caminhar para frente 100 passos/pontos (deixando rastro)
8. Girar 90 graus para a esquerda (ou direita)

40

Quadrado com Turtle

```
import turtle
tart = turtle.Turtle()
tart.forward(100)
tart.left(90)
tart.forward(100)
tart.left(90)
tart.forward(100)
tart.left(90)
tart.forward(100)
tart.left(90)
```

41

Quadrado: análise do script (1/5)

```
import turtle
tart = turtle.Turtle()
tart.forward(100)
tart.left(90)
tart.forward(100)
tart.left(90)
tart.forward(100)
tart.left(90)
tart.forward(100)
tart.left(90)
```

#risca e gira

42

Quadrado: análise do script (2/5)

```
import turtle
tart = turtle.Turtle()
tart.forward(100)
tart.left(90)
tart.forward(100)
tart.left(90)
tart.forward(100)
tart.left(90)
tart.forward(100)
tart.left(90)
```

#risca e gira

43

Quadrado: análise do script (3/5)

```
import turtle
tart = turtle.Turtle()
tart.forward(100)
tart.left(90)
tart.forward(100)
tart.left(90)
tart.forward(100)
tart.left(90)
tart.forward(100)
tart.left(90)
```

#risca e gira

44

Quadrado: análise do script (4/5)

```
import turtle
tart = turtle.Turtle()
tart.forward(100)
tart.left(90)
tart.forward(100)
tart.left(90)
tart.forward(100)
tart.left(90)
tart.forward(100)
tart.left(90)
```

#risca e gira

45

Quadrado: análise do script (5/5)

```
import turtle
tart = turtle.Turtle()
tart.forward(100)
tart.left(90)
tart.forward(100)
tart.left(90)
tart.forward(100)
tart.left(90)
tart.forward(100)
tart.left(90)
```

As ações:

riscar um lado e virar 90°
são realizadas 4 vezes.



Repetir 4 vezes:

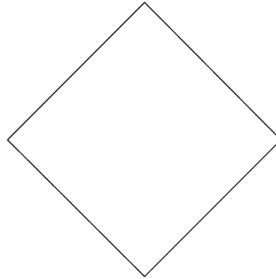
riscar um lado
virar 90°

*Mais a frente veremos como melhorar e otimizar a escrita
do código para especificar trechos de código em Python
que se repetem*

46

Revisitando o quadrado

Trace o quadrado abaixo. Os seus lados têm comprimento 100.



1. Virar 45°
2. Desenhar um quadrado de lado 100
3. Desvirar 45°

47

Quadrado inclinado: uma solução

```
import turtle
pat = turtle.Turtle()
pat.right(45)
pat.forward(100)
pat.left(90)
pat.forward(100)
pat.left(90)
pat.forward(100)
pat.left(90)
pat.forward(100)
pat.left(90)
pat.left(45)
```

inclina a pat
#desenha o quadrado

volta para direção original

48

Quadrado verde

Trace o quadrado abaixo. Os seus lados têm comprimento 100.



Novas instruções: Preencher um desenho

Mudar a cor do preenchimento:

```
tartaruga.fillcolor(cor)
```

Ligar o modo preenchimento antes de começar o desenho:

```
tartaruga.begin_fill()
```

Desligar o modo preenchimento após terminar o desenho

```
tartaruga.end_fill()
```

49

Lista de cores

Lista de cores e seus nomes

Algumas cores:



'red',	'white',
green',	'brown',
'blue',	'chocolate',
'yellow',	'gray',
'pink',	'magenta',
'orange',	'cyan',
'black',	'lime',
'violet',	'darkblue'

<http://erikasarti.net/html/tabela-cores/>

50

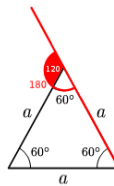
Quadrado verde: uma solução

```
import turtle
pat = turtle.Turtle()
pat.right(45)           # inclina a pat
pat.begin_fill()        # inicia o modo de preenchimento
pat.fillcolor('green')  # determina a cor do preenchimento
pat.forward(100)         # desenha o quadrado
pat.left(90)
pat.forward(100)
pat.left(90)
pat.forward(100)
pat.left(90)
pat.forward(100)
pat.left(90)
pat.end_fill()          # termina o modo de preenchimento (qdo pinta)
pat.left(45)            # volta para direção original
```

51

Exercícios de desenho

1. Desenhar um triângulo



Lembre-se da direção da tartaruga

2. Desenhar um círculo

Como a tartaruga desenha pelo perímetro $\rightarrow \text{passo} = \pi * \text{raio} / 180$

3. Desenhar 2 círculos um dentro do outro

52

Solução do exercício 1

Desenhar um triângulo

```
import turtle
t=turtle.Turtle()
lado=100
```

Uma solução:

```
t.forward(lado)
t.left(120)
t.forward(lado)
t.left(120)
t.forward(lado)
t.left(120)
```

53

Solução do exercício 2

Desenhar um círculo

```
import turtle
t=turtle.Turtle()
```

Uma solução:

```
import math
t.circle(360/2*math.pi) #pelo raio
```

54

Solução do exercício 3

Desenhar um circulo dentro do outro

```
import turtle
t=turtle.Turtle()
raio =100
t.circle(raio)

t.left(90)
t.up()
t.forward(50)
t.right(90)
t.down()
t.circle(50)
```

55

Criando funções

Mais quadrados!!!

Trace o desenho abaixo. Os lados das figuras têm comprimento 100.



Quadrado de novo!!!!

Como "ensiná-la" a
desenhar um quadrado
de lado 100 ?

57

Criando novas funções

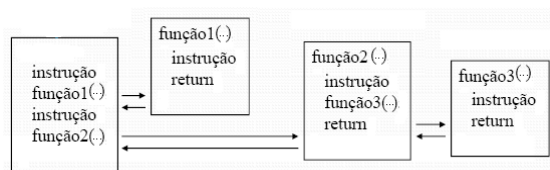
Uma **função** é uma sequência de instruções (bloco de código) independente, que realiza uma tarefa específica.

- ✓ Há funções fornecidas pela linguagem, como *math.sqrt()*, *math.sin()* e etc., cujos códigos estão nos módulos importados.
- ✓ O programador também pode escrever suas próprias funções, associando um nome a elas.
- ✓ Vantagens: código modularizado, mais legível e sem reescrita.

58

Chamadas de funções

- ✓ Uma função em geral computa um ou mais valores a partir de valores recebidos. Portanto, *uma função pode receber e/ou retornar valores*.
- ✓ As funções são "invocadas" (chamadas/ativadas) pelo nome, por outras partes do script ou por outra função.
- ✓ Quando uma função termina sua execução, o controle retorna para o ponto de onde ela foi chamada (invocada).



59

Função DesenhaQuadrado (1/9)

Qual é a tarefa desta função?

60

Função DesenhaQuadrado (2/9)

Qual a tarefa desta função?

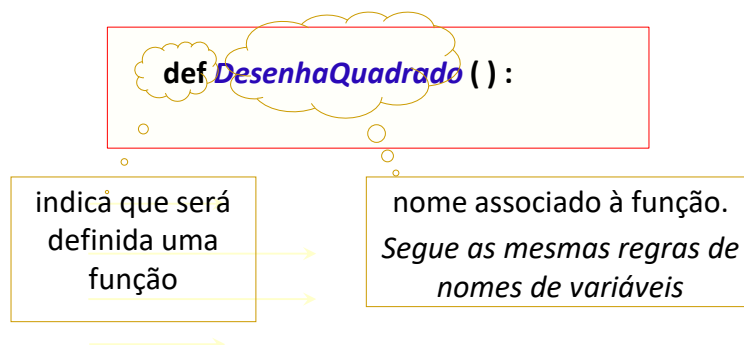
Instruir uma tartaruga a desenhar um quadrado

61

Função DesenhaQuadrado (3/9)

Qual a tarefa desta função?

Instruir uma tartaruga como desenhar um quadrado



62

Função DesenhaQuadrado (4/9)

Qual é a tarefa desta função? Desenhar um quadrado

Do que a função precisa para realizar sua tarefa?

```
def DesenhaQuadrado ( ) :
```

63

Função DesenhaQuadrado (5/9)

Qual a tarefa desta função? Desenhar um quadrado

Do que a função precisa para realizar sua tarefa?

De uma tartaruga para executar o desenho

```
def DesenhaQuadrado ( ) :
```

64

Função DesenhaQuadrado (6/9)

Qual a tarefa desta função? Desenhar um quadrado

Do que a função precisa para realizar sua tarefa?

De uma tartaruga para executar o desenho

`def DesenhaQuadrado(tart) :`

Parâmetro
necessário

65

Função DesenhaQuadrado (7/9)

Qual é a tarefa desta função? Desenhar um quadrado

Do que a função precisa para realizar sua tarefa? De uma tartaruga

Como a função realiza sua tarefa?

66

Função DesenhaQuadrado (8/9)

Qual a tarefa desta função? Desenhar um quadrado

Do que a função precisa para realizar sua tarefa? De uma tartaruga

Como a função realiza sua tarefa?

Identação

```
def DesenhaQuadrado (tart):  
    """ Quadrado lado 100 """  
    tart.forward(100)  
    tart.left(90)  
    tart.forward(100)  
    tart.left(90)  
    tart.forward(100)  
    tart.left(90)  
    tart.forward(100)  
    tart.left(90)  
    return
```

67

Função DesenhaQuadrado (9/9)

Qual a tarefa desta função? Desenhar um quadrado

Do que a função precisa para realizar sua tarefa? De uma tartaruga

Como a função realiza sua tarefa?

Identação

```
def DesenhaQuadrado (tart):  
    """ Quadrado lado 100 """  
    tart.forward(100)  
    tart.left(90)  
    ...  
    return
```

Termina a função. O controle volta ao ponto de onde a função foi chamada. Pode designar um valor a ser retornado.

68

Usando a função DesenhaQuadrado

69

DesenhaQuadrado em ação!

```
import turtle
def DesenhaQuadrado (tart):
    """ Quadrado lado 100 """
    tart.forward(100)
    tart.left(90)
    tart.forward(100)
    tart.left(90)
    tart.forward(100)
    tart.left(90)
    tart.forward(100)
    tart.left(90)
    return

pat = turtle.Turtle()
pat.right(45)
DesenhaQuadrado(pat) ←
pat.left(45)
```

70

Usando funções: como é! (1/23)

```
import turtle
def DesenhaQuadrado(tart):
    tart.forward(100)
    tart.left(90)
    ... ..
    return

pat = turtle.Turtle()
pat.color('red')
alex=turtle.Turtle()
alex.color('blue')
pat.right(45)
DesenhaQuadrado(pat)
DesenhaQuadrado(alex)
```

Módulo
quadrado

Sessão
Python

71

Usando funções: como é! (2/23)

```
import turtle
def DesenhaQuadrado(tart):
    tart.forward(100)
    tart.left(90)
    ... ..
    return

pat = turtle.Turtle()
pat.color('red')
alex=turtle.Turtle()
alex.color('blue')
pat.right(45)
DesenhaQuadrado(pat)
DesenhaQuadrado(alex)
```

Módulo
quadrado

Sessão
Python

72

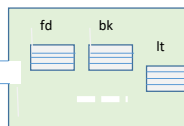
Usando funções: como é! (3/23)

```
import turtle
def DesenhaQuadrado(tart):
    tart.forward(100)
    tart.left(90)
    ... ..
    return

pat = turtle.Turtle()
pat.color('red')
alex=turtle.Turtle()
alex.color('blue')
pat.right(45)
DesenhaQuadrado(pat)
DesenhaQuadrado(alex)
```

Módulo
quadrado

Sessão
Python



Módulo
turtle

73

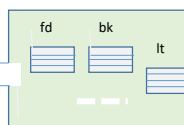
Usando funções: como é! (4/23)

```
import turtle
def DesenhaQuadrado(tart):
    tart.forward(100)
    tart.left(90)
    ... ..
    return

pat = turtle.Turtle()
pat.color('red')
alex=turtle.Turtle()
alex.color('blue')
pat.right(45)
DesenhaQuadrado(pat)
DesenhaQuadrado(alex)
```

Módulo
quadrado

Sessão
Python



Módulo
turtle

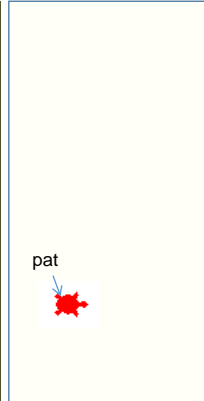
74

Usando funções: como é! (5/23)

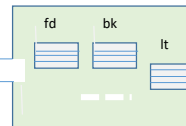
```
import turtle
def DesenhaQuadrado(tart):
    tart.forward(100)
    tart.left(90)
    ... ..
    return
```

```
pat = turtle.Turtle()
pat.color('red')
alex=turtle.Turtle()
alex.color('blue')
pat.right(45)
DesenhaQuadrado(pat)
DesenhaQuadrado(alex)
```

Módulo
quadrado



Sessão
Python



Módulo
turtle

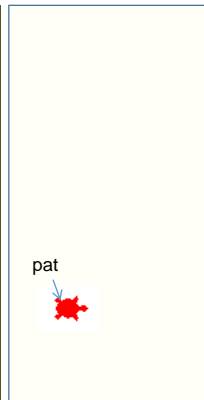
75

Usando funções: como é! (6/23)

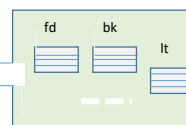
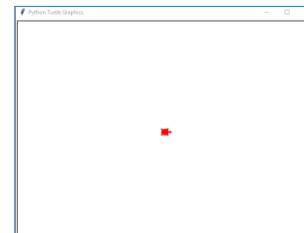
```
import turtle
def DesenhaQuadrado(tart):
    tart.forward(100)
    tart.left(90)
    ... ..
    return
```

```
pat = turtle.Turtle()
pat.color('red')
alex=turtle.Turtle()
alex.color('blue')
pat.right(45)
DesenhaQuadrado(pat)
DesenhaQuadrado(alex)
```

Módulo
quadrado



Sessão
Python



Módulo
turtle

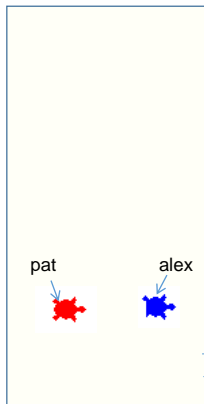
76

Usando funções: como é! (6/23)

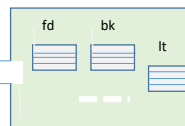
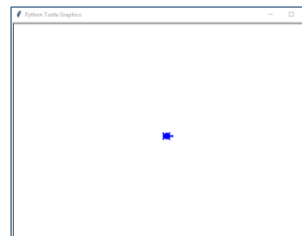
```
import turtle
def DesenhaQuadrado(tart):
    tart.forward(100)
    tart.left(90)
    ... ..
    return

pat = turtle.Turtle()
pat.color('red')
alex=turtle.Turtle()
alex.color('blue')
pat.right(45)
DesenhaQuadrado(pat)
DesenhaQuadrado(alex)
```

Módulo
quadrado



Sessão
Python



Módulo
turtle

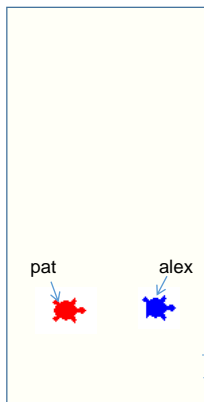
77

Usando funções: como é! (7/23)

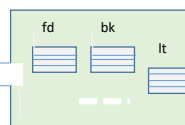
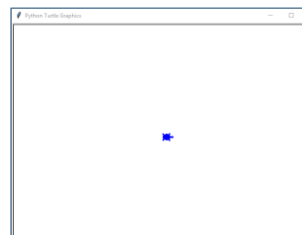
```
import turtle
def DesenhaQuadrado(tart):
    tart.forward(100)
    tart.left(90)
    ... ..
    return

pat = turtle.Turtle()
pat.color('red')
alex=turtle.Turtle()
alex.color('blue')
pat.right(45)
DesenhaQuadrado(pat)
DesenhaQuadrado(alex)
```

Módulo
quadrado



Sessão
Python



Módulo
turtle

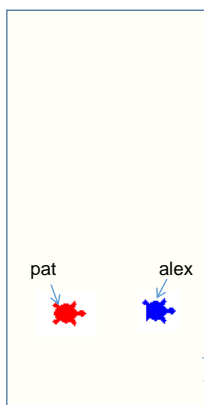
78

Usando funções: como é! (8/23)

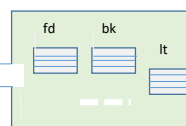
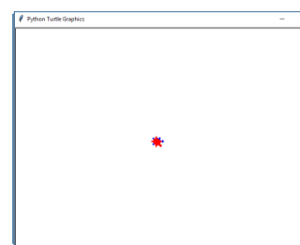
```
import turtle
def DesenhaQuadrado(tart):
    tart.forward(100)
    tart.left(90)
    ... ..
    return

pat = turtle.Turtle()
pat.color('red')
alex=turtle.Turtle()
alex.color('blue')
pat.right(45)
DesenhaQuadrado(pat)
DesenhaQuadrado(alex)
```

Módulo
quadrado



Sessão
Python



Módulo
turtle

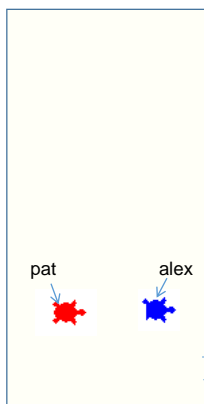
79

Usando funções: como é! (9/23)

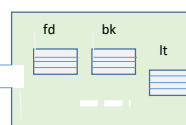
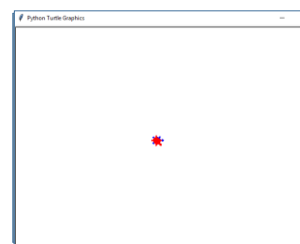
```
import turtle
def DesenhaQuadrado(tart):
    tart.forward(100)
    tart.left(90)
    ... ..
    return

pat = turtle.Turtle()
pat.color('red')
alex=turtle.Turtle()
alex.color('blue')
pat.right(45)
DesenhaQuadrado(pat)
DesenhaQuadrado(alex)
```

Módulo
quadrado



Sessão
Python



Módulo
turtle

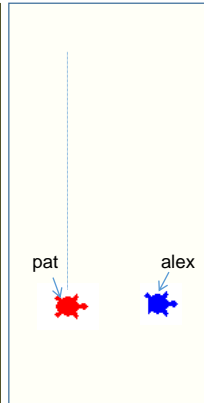
80

Usando funções: como é! (10/23)

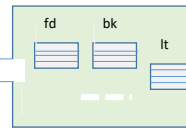
```
import turtle
def DesenhaQuadrado(tart):
    tart.forward(100)
    tart.left(90)
    ... ..
    return

pat = turtle.Turtle()
pat.color('red')
alex=turtle.Turtle()
alex.color('blue')
pat.right(45)
DesenhaQuadrado(pat)
DesenhaQuadrado(alex)
```

Módulo
quadrado



Sessão
Python



Módulo
turtle

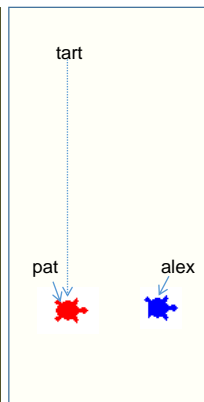
81

Usando funções: como é! (11/23)

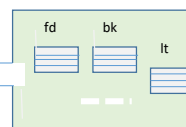
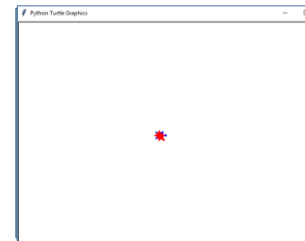
```
import turtle
def DesenhaQuadrado(tart):
    tart.forward(100)
    tart.left(90)
    ... ..
    return

pat = turtle.Turtle()
pat.color('red')
alex=turtle.Turtle()
alex.color('blue')
pat.right(45)
DesenhaQuadrado(pat)
DesenhaQuadrado(alex)
```

Módulo
quadrado



Sessão
Python



Módulo
turtle

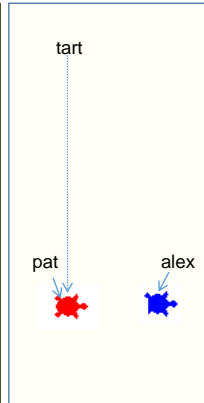
82

Usando funções: como é! (12/23)

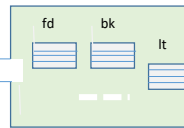
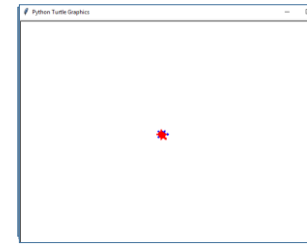
```
import turtle
def DesenhaQuadrado(tart):
    tart.forward(100)
    tart.left(90)
    ... ..
    return
```

```
pat = turtle.Turtle()
pat.color('red')
alex=turtle.Turtle()
alex.color('blue')
pat.right(45)
DesenhaQuadrado(pat)
DesenhaQuadrado(alex)
```

Módulo
quadrado



Sessão
Python



Módulo
turtle

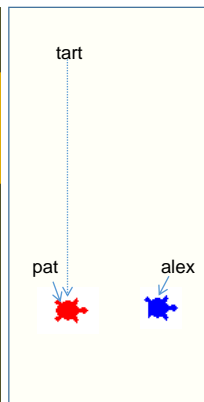
83

Usando funções: como é! (13/23)

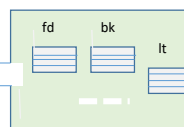
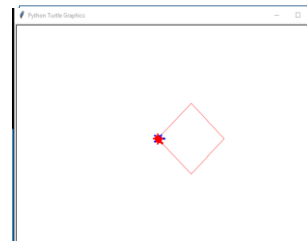
```
import turtle
def DesenhaQuadrado(tart):
    tart.forward(100)
    tart.left(90)
    ... ..
    return
```

```
pat = turtle.Turtle()
pat.color('red')
alex=turtle.Turtle()
alex.color('blue')
pat.right(45)
DesenhaQuadrado(pat)
DesenhaQuadrado(alex)
```

Módulo
quadrado



Sessão
Python



Módulo
turtle

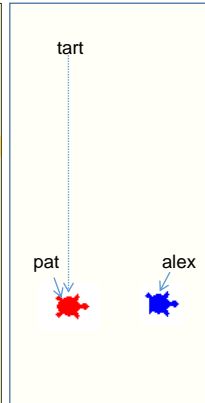
84

Usando funções: como é! (14/23)

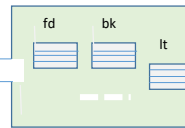
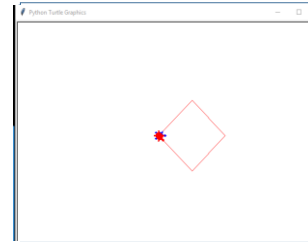
```
import turtle
def DesenhaQuadrado(tart):
    tart.forward(100)
    tart.left(90)
    ... ..
    return
```

```
pat = turtle.Turtle()
pat.color('red')
alex=turtle.Turtle()
alex.color('blue')
pat.right(45)
DesenhaQuadrado(pat)
DesenhaQuadrado(alex)
```

Módulo
quadrado



Sessão
Python



Módulo
turtle

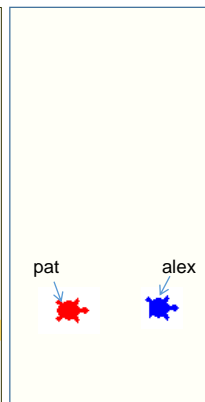
85

Usando funções: como é! (15/23)

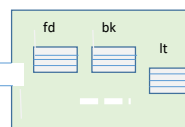
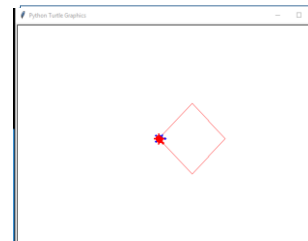
```
import turtle
def DesenhaQuadrado(tart):
    tart.forward(100)
    tart.left(90)
    ... ..
    return
```

```
pat = turtle.Turtle()
pat.color('red')
alex=turtle.Turtle()
alex.color('blue')
pat.right(45)
DesenhaQuadrado(pat)
DesenhaQuadrado(alex)
```

Módulo
quadrado



Sessão
Python



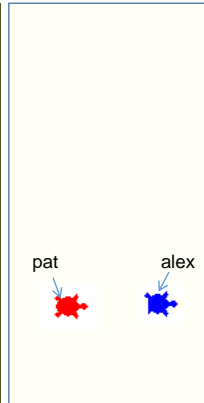
Módulo
turtle

86

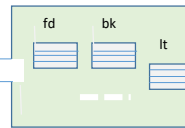
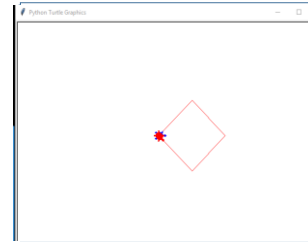
Usando funções: como é! (16/23)

```
import turtle
def DesenhaQuadrado(tart):
    tart.forward(100)
    tart.left(90)
    ... ..
    return
pat = turtle.Turtle()
pat.color('red')
alex=turtle.Turtle()
alex.color('blue')
pat.right(45)
DesenhaQuadrado(pat)
DesenhaQuadrado(alex)
```

Módulo
quadrado



Sessão
Python



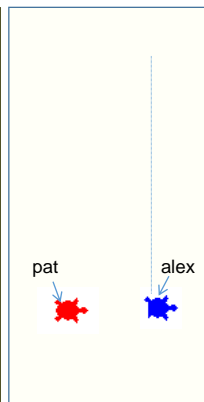
Módulo
turtle

87

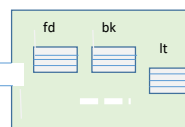
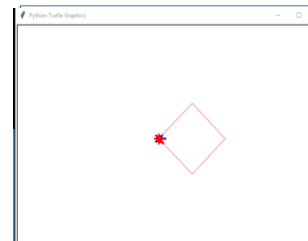
Usando funções: como é! (17/23)

```
import turtle
def DesenhaQuadrado(tart):
    tart.forward(100)
    tart.left(90)
    ... ..
    return
pat = turtle.Turtle()
pat.color('red')
alex=turtle.Turtle()
alex.color('blue')
pat.right(45)
DesenhaQuadrado(pat)
DesenhaQuadrado(alex)
```

Módulo
quadrado



Sessão
Python



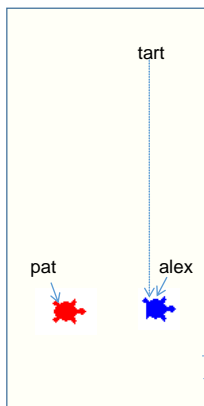
Módulo
turtle

88

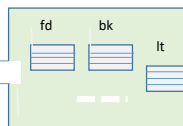
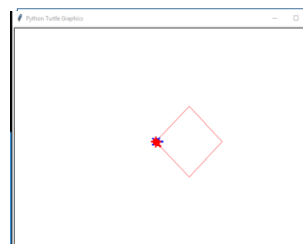
Usando funções: como é! (18/23)

```
import turtle
def DesenhaQuadrado(tart):
    tart.forward(100)
    tart.left(90)
    ... ..
    return
pat = turtle.Turtle()
pat.color('red')
alex=turtle.Turtle()
alex.color('blue')
pat.right(45)
DesenhaQuadrado(pat)
DesenhaQuadrado(alex)
```

Módulo
quadrado



Sessão
Python



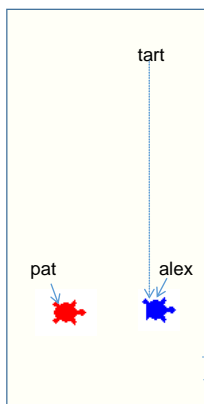
Módulo
turtle

89

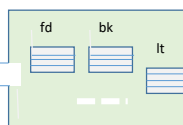
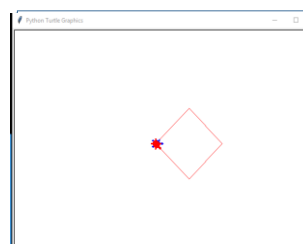
Usando funções: como é! (19/23)

```
import turtle
def DesenhaQuadrado(tart):
    tart.forward(100)
    tart.left(90)
    ... ..
    return
pat = turtle.Turtle()
pat.color('red')
alex=turtle.Turtle()
alex.color('blue')
pat.right(45)
DesenhaQuadrado(pat)
DesenhaQuadrado(alex)
```

Módulo
quadrado



Sessão
Python



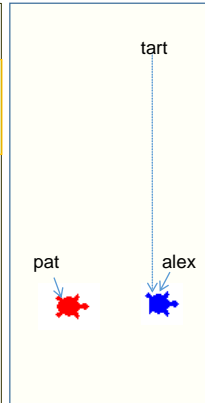
Módulo
turtle

90

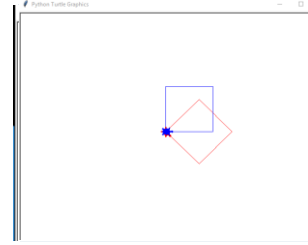
Usando funções: como é! (20/23)

```
import turtle
def DesenhaQuadrado(tart):
    tart.forward(100)
    tart.left(90)
    ...
    return
pat = turtle.Turtle()
pat.color('red')
alex=turtle.Turtle()
alex.color('blue')
pat.right(45)
DesenhaQuadrado(pat)
DesenhaQuadrado(alex)
```

Módulo
quadrado



Sessão
Python



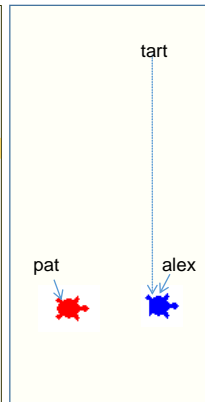
Módulo
turtle

91

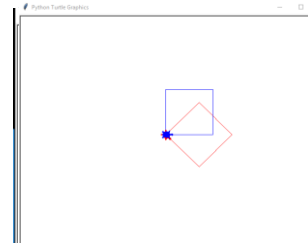
Usando funções: como é! (21/23)

```
import turtle
def DesenhaQuadrado(tart):
    tart.forward(100)
    tart.left(90)
    ...
    return
pat = turtle.Turtle()
pat.color('red')
alex=turtle.Turtle()
alex.color('blue')
pat.right(45)
DesenhaQuadrado(pat)
DesenhaQuadrado(alex)
```

Módulo
quadrado



Sessão
Python



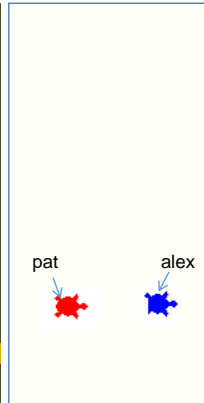
Módulo
turtle

92

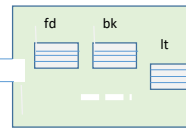
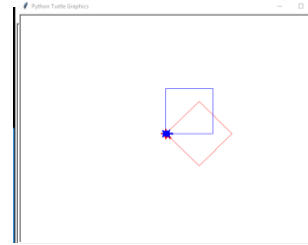
Usando funções: como é! (22/23)

```
import turtle
def DesenhaQuadrado(tart):
    tart.forward(100)
    tart.left(90)
    ... ..
    return
pat = turtle.Turtle()
pat.color('red')
alex=turtle.Turtle()
alex.color('blue')
pat.right(45)
DesenhaQuadrado(pat)
DesenhaQuadrado(alex)
```

Módulo
quadrado



Sessão
Python



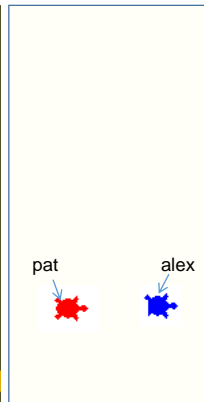
Módulo
turtle

93

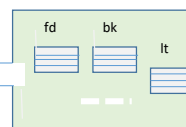
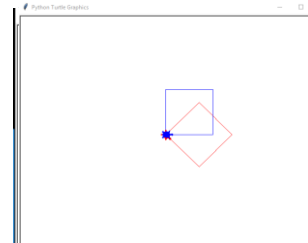
Usando funções: como é! (23/23)

```
import turtle
def DesenhaQuadrado(tart):
    tart.forward(100)
    tart.left(90)
    ... ..
    return
pat = turtle.Turtle()
pat.color('red')
alex=turtle.Turtle()
alex.color('blue')
pat.right(45)
DesenhaQuadrado(pat)
DesenhaQuadrado(alex)
```

Módulo
quadrado



Sessão
Python



Módulo
turtle

94



Criando uma nova função: sintaxe

A instrução **def** é utilizada para criar funções.

```
def NomedaFunção (< parâmetros>) :  
    <instrução1>  
    ...  
    <instruçãon>  
    return <valor>
```

Bloco de Código
ou
Corpo da Função

- ✓ Os parâmetros discriminam os valores (se existirem) que devem ser entregues à função para que ela possa executar sua tarefa.
- ✓ O comando **return** termina a função e retorna um <valor> para o ponto de onde a função foi chamada.

95



Observações

- ✓ A definição da função deve ser feita antes de sua chamada, de modo que o interpretador Python reconheça o seu nome.
- ✓ Não pode haver funções e variáveis com o mesmo nome.
- ✓ Os valores recebidos são associados aos parâmetros na ordem em que foram definidos na chamada da função.
- ✓ A chamada da função deve incluir um valor para cada parâmetro.
- ✓ Uma função pode ou não retornar um valor.
- ✓ Uma função pode ter 0, 1 ou mais *returns*.
- ✓ Uma função pode chamar outra função.

96

Revendo os múltiplos quadrados

Trace o desenho abaixo. Os lados das figuras têm comprimento 100.



1. Desenhar um quadrado colorido de verde
2. Deslocar-se
3. Desenhar outro quadrado colorido de amarelo

97

Múltiplos quadrados: uma solução

```
import turtle

def DesenhaQuadrado (tart):
    """ Quadrado lado 100 """
    tart.forward(100)
    tart.left(90)
    tart.forward(100)
    tart.left(90)
    tart.forward(100)
    tart.left(90)
    tart.forward(100)
    tart.left(90)
    return
```

98

Múltiplos quadrados: parte 2

```
t=turtle.Turtle()
t.reset()
t.begin_fill()                                # inicia o modo de preenchimento
t.fillcolor('green')                         # determina a cor do preenchimento do quadrado
t.left(45)
DesenhaQuadrado(t)                           #desenha o 1º quadrado
t.end_fill()                                 # termina o modo de preenchimento (quando pinta)
t.right(45)
t.up()                                        # desloca para a direita
t.fd(100)
t.down()
t.fillcolor('yellow')                        # determina a cor do preenchimento do quadrado
t.begin_fill()                               # inicia o modo de preenchimento
t.left(45)
DesenhaQuadrado(t)                           #desenha o 2º quadrado
t.end_fill()                                 # termina o modo de preenchimento (quando pinta)
```

99

Descoberta de funções úteis

```
t=turtle.Turtle()
t.reset()
t.begin_fill()                                # inicia o modo de preenchimento
t.fillcolor('green')                         # determina a cor do preenchimento do quadrado
t.left(45)
DesenhaQuadrado(t)                           #desenha o 1º quadrado
t.end_fill()                                 # termina o modo de preenchimento (quando pinta)
t.right(45)
t.up()
t.fd(100)
t.down()
t.fillcolor('yellow')                        # determina a cor do preenchimento do quadrado
t.begin_fill()                               # inicia o modo de preenchimento
t.left(45)
DesenhaQuadrado(t)                           #desenha o 2º quadrado
t.end_fill()                                 # termina o modo de preenchimento (quando pinta)
```

**criar uma função para
deslocar para a direita**

100



Deslocar para direita: uma solução

```
import turtle

def DesenhaQuadrado (tart):
    """ Quadrado lado 100"""
    tart.forward(100)
    tart.left(90)
    tart.forward(100)
    tart.left(90)
    tart.forward(100)
    tart.left(90)
    tart.forward(100)
    tart.left(90)
    return

def DeslocaDireita(tart):
    """ Desloca 100 p/direita"""
    tart.up()
    tart.fd(100)
    tart.down()
    return
```

101



Usando DeslocaDireita

```
t=turtle.Turtle()
t.reset()
t.begin_fill()                # inicia o modo de preenchimento
t.fillcolor('green')          # determina a cor do preenchimento do quadrado
t.left(45)
DesenhaQuadrado(t)            #desenha o 1º quadrado
t.end_fill()                  # termina o modo de preenchimento (quando pinta)
t.right(45)
DeslocaDireita(t)
t.fillcolor('yellow')          # determina a cor do preenchimento do quadrado
t.begin_fill()                # inicia o modo de preenchimento
t.left(45)
DesenhaQuadrado(t)            #desenha o 2º quadrado
t.end_fill()                  # termina o modo de preenchimento (quando pinta)
```

102

Reanalizando os quadrados

```
t=turtle.Turtle()
t.reset()
t.begin_fill()
t.fillcolor('green')
t.left(45)
DesenhaQuadrado(t)
t.end_fill()
t.right(45)
DeslocaDireita(t)
t.fillcolor('yellow')
t.begin_fill()
t.left(45)
DesenhaQuadrado(t)
t.end_fill()
```

Quadrado Verde

inicia o modo de preenchimento

determina a cor do preenchimento do quadrado

#desenha o 1º quadrado

termina o modo de preenchimento (quando pinta)

determina a cor do preenchimento do quadrado

inicia o modo de preenchimento

#desenha o 2º quadrado

termina o modo de preenchimento (quando pinta)

Quadrado Amarelo

103

Mais de um quadrado colorido?



1. Desenhar um quadrado colorido de verde
2. Deslocar-se
3. Desenhar outro quadrado colorido de amarelo

DESAFIO: Criar uma nova função *DesenhaQuadradoColorido* para desenhar um quadrado colorido cuja cor é fornecida à função

104



Criar DesenhaQuadradoColorido (1/4)

A função `DesenhaQuadrado()` desenvolvida ensina uma tartaruga a traçar um quadrado de lado 100, sem cor de preenchimento.

```
def DesenhaQuadrado (tart):  
    """ Quadrado lado 100 """  
    tart.forward(100)  
    tart.left(90)  
    tart.forward(100)  
    tart.left(90)  
    tart.forward(100)  
    tart.left(90)  
    tart.forward(100)  
    tart.left(90)  
    return
```

O que fazer para que ela possa desenhar um quadrado colorido com uma cor qualquer?

105



Criar DesenhaQuadradoColorido (2/4)

Do que a função precisa para realizar sua tarefa?

De uma tartaruga

```
def DesenhaQuadradoColorido (tart)
```

106



Criar DesenhaQuadradoColorido (3/4)

Do que a função precisa para realizar sua tarefa?

De uma tartaruga e da cor

```
def DesenhaQuadradoColorido (tart, cor)
```

107



Criar DesenhaQuadradoColorido (4/4)

```
def DesenhaQuadradoColorido (tart, cor):  
    """Quadrado lado 100 com cor"""  
    tart.begin_fill() # inicia o modo de preenchimento  
    tart.fillcolor(cor) # determina a cor  
    tart.forward(100)  
    tart.left(90)  
    tart.forward(100)  
    tart.left(90)  
    tart.forward(100)  
    tart.left(90)  
    tart.forward(100)  
    tart.left(90)  
    tart.end_fill() # pinta o interior do quadrado  
    return
```

108



Usando DesenhaQuadradoColorido

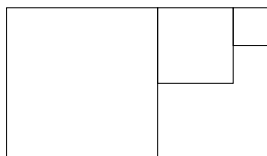
```
t=turtle.Turtle()  
t.reset()  
t.left(45)  
DesenhaQuadradoColorido(t,'green') #desenha o 1º quadrado  
t.right(45)  
DeslocaDireita(t)  
t.left(45)  
DesenhaQuadradoColorido(t,'yellow') #desenha o 2º quadrado
```

109



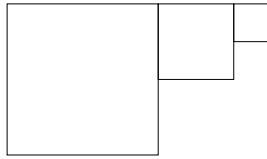
Mãos na massa nos quadrados!

Traçar a figura abaixo. O lado do quadrado maior mede 100. O lado do quadrado à direita mede a metade do lado do quadrado à esquerda.



110

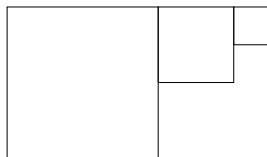
Três quadrados: solução (1/3)



1. Desenhar quadrado de tamanho 100
2. Deslocar 100
3. Desenhar quadrado de tamanho 50
4. Deslocar 50
5. Desenhar quadrado de tamanho 25

111

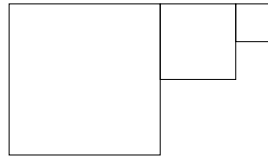
3 quadrados: solução (2/3)



1. Desenhar quadrado de tamanho 100
 2. Deslocar 100
 3. Desenhar quadrado de tamanho 50
 4. Deslocar 50
 5. Desenhar quadrado de tamanho 25
- Só muda o valor
- Só muda o tamanho do lado

112

3 quadrados: solução (3/3)



1. Desenhar quadrado de tamanho 100
 2. Deslocar 100
 3. Desenhar quadrado de tamanho 50
 4. Deslocar 50
 5. Desenhar quadrado de tamanho 25
- Só muda o valor ←
- Só muda o tamanho do lado →

Como modificar as funções já criadas para reutilizá-las?

DesenhaQuadrado(tart): desenha um quadrado de lado 100

DeslocaDireita(tart): deslocamento fixo (100) para a direita

113

função DesenhaQuadrado (1/3)

O que fazer para que DesenhaQuadrado() possa traçar um quadrado de lado especificado?


```
def DesenhaQuadrado(tart):
    """Quadrado lado 100"""
    tart.forward(100)
    tart.left(90)
    tart.forward(100)
    tart.left(90)
    tart.forward(100)
    tart.left(90)
    tart.forward(100)
    tart.left(90)
    return
```

114

função DesenhaQuadrado (2/3)

Do que a função precisa para realizar sua tarefa?

De uma tartaruga




```
def DesenhaQuadrado (tart
```

115

função DesenhaQuadrado (3/3)

Do que a função precisa para realizar sua tarefa?

De uma tartaruga **e do tamanho do lado**



```
def DesenhaQuadrado (tart, lado)
```

116

Desenha quadrado: uma solução

```
def DesenhaQuadrado(tart, lado):  
    """Quadrado lado recebido"""  
    tart.forward(lado)  
    tart.left(90)  
    tart.forward(lado)  
    tart.left(90)  
    tart.forward(lado)  
    tart.left(90)  
    tart.forward(lado)  
    tart.left(90)  
    return
```

117

Usando DesenhaQuadrado

```
import turtle  
def DesenhaQuadrado(tart, lado):  
    .....  
    return  
  
def DeslocaDireita(tart):  
    .....  
    return  
  
tart = turtle.Turtle()  
DesenhaQuadrado(tart, 100)  
DeslocaDireita(tart) # desloca para a direita 100  
DesenhaQuadrado(tart, 50)  
tart.penup() } # desloca para a direita 50  
tart.fd(50)  
tart.pendown()  
DesenhaQuadrado(tart, 25)
```

118

DeslocaDireita modificada

```
import turtle
def DesenhaQuadrado(tart,lado):
    .....
    return

def DeslocaDireita(tart):
    .....
    return

tart = turtle.Turtle()
DesenhaQuadrado(tart,100)
DeslocaDireita(tart)                # desloca para a direita 100
DesenhaQuadrado(tart,50)
tart.penup()                        # desloca para a direita 50
tart.fd(50)
tart.pendown()
DesenhaQuadrado(tart,25)
```

Modificar a função
DeslocaDireita
para aceitar como
parâmetro o comprimento
do deslocamento

119

DeslocaDireita: nova solução

```
import turtle
def DesenhaQuadrado(tart,lado):
    .....
    return

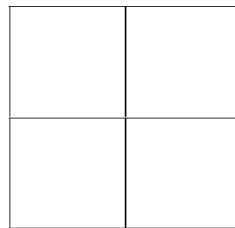
def DeslocaDireita(tart,dist):
    """Desloca distância p/direita"""
    tart.up()
    tart.fd(dist)
    tart.down()
    return

tart = turtle.Turtle()
DesenhaQuadrado(tart,100)
DeslocaDireita(tart,100)
DesenhaQuadrado(tart,50)
DeslocaDireita(tart,50)
DesenhaQuadrado(tart,25)
```

120

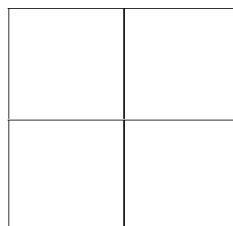
4 quadrados em um quadrado

Traçar a figura abaixo. Cada lado tem comprimento 200 e consiste de 4 quadrados idênticos.



121

4 quadrados: resolvendo

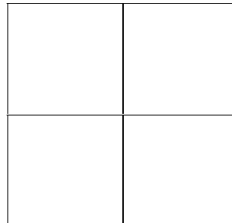


Exibir linha a linha
OU
Exibir coluna a coluna?

122



4 quadrados: pensando a solução



Exibir linha a linha
OU
Exibir coluna a coluna?

Tanto faz!!!

123

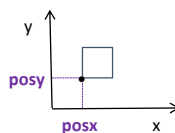


4 quadrados e eixos



Importante identificar duas coisas:

- a) Onde colocar a tartaruga no início de cada linha (ou coluna) → identificar o ponto (posx, posy) de referência:



- b) Onde recolocar a tartaruga para desenhar o outro quadrado dentro da linha (coluna)? → deslocar a tartaruga ao longo da linha (coluna) por um valor igual ao lado do quadrado.

124

4 quadrados: uma solução

```
import turtle

def DesenhaQuadrado(tart):
    ...
def DeslocaDireita(tart):
    .....

tart = turtle.Turtle()
DesenhaQuadrado(tart)
DeslocaDireita(tart)
DesenhaQuadrado(tart)
tart.right(90)
DeslocaDireita(tart)
DesenhaQuadrado(tart)
tart.left(90)
DesenhaQuadrado(tart)
```