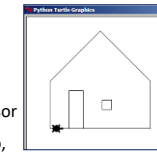
Módulo *Turtle* (tartaruga)

Turtle Robots (Robôs tartaruga)

- Uma classe de robôs (educacionais) que obedecem 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-programado para realizar algumas ações
  - ✓ O cursor pode, ou não, traçar sua trajetória
  - ✓ Pode-se ajustar suas características como cor e a largura da caneta, o formato da tartaruga e a velocidade que desenha, cor de fundo da janela gráfica, etc.



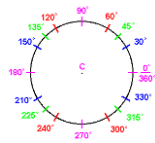
Desenhando com turtle

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

esquerda

trás ▸ frente

direita



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

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)

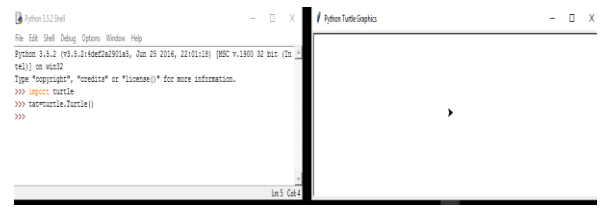
Usando a tartaruga criada

```
>>>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.

Janela da Tartaruga



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

Ativando uma função

A tartaruga é instruída a realizar alguma ação que ela conhece e sabe responder (comando) do seguinte modo:

```
<nomedatataruga> . <ação>
```


Tartaruga em movimento

Movimentando a tartaruga

"Para frente distância *n*"

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

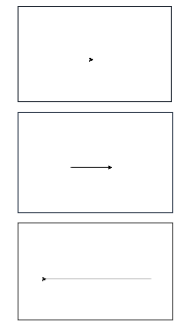
"Para trás distância *n*"

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

Obs. n é um número real

Exemplos de movimentos

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

>>> pat.fd(100)


>>> pat.bk(257.3)
```



Girando a tartaruga

Girar à direita *x* graus (x é um ângulo)

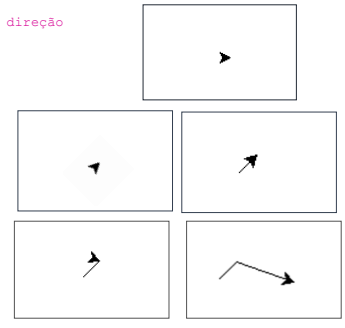
```
tartaruga.right(x)
tartaruga.rt(x)
```

Girar à esquerda *x* graus (x é um ângulo)

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


Mais exemplos de movimentos

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

>>>pat.left(45)
>>>pat.fd(20)

>>>pat.right(65)
>>>pat.fd(50)
```



Levantando/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()
```

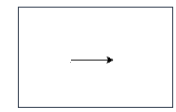
Para *pular* (deslocar-se sem riscar) é preciso antes, levantar a caneta

```
tartaruga.up()
```

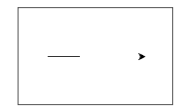

Rastros da tartaruga

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

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

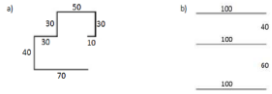


```
>>>pat.up()
>>>pat.fd(100)
```



Mãos na massa!!

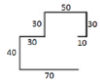
1) Um ciclista saiu pedalando no sentido norte. Percorreu 100 metros. Em seguida, virou 38º para a direita, andou 200 m, virou 57º para a esquerda, pedalou mais 50m e, finalmente, girou 9º para a direita.
   a. Supondo que a tartaruga *tat* represente o ciclista e que cada metro percorrido é um passo dela, trace a trajetória do ciclista.
   b. Para voltar a pedalar no sentido norte, quantos graus e para qual lado o ciclista deve virar?
2) Execute os comandos necessários para *tat* reproduzir os desenhos abaixo.



Para frente n:
`tat.fd(n)`
Para trás n:
`tat.bk(n)`
Virar ° p/esq :
`tat.left(ang)`
Virar ° p/direita:
`tat.right(ang)`
Levantar a caneta:
`tat.up()`
Abaixar a caneta:
`tat.down()`

Editor e Scripts grandes

Editor vs Modo Interativo



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

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

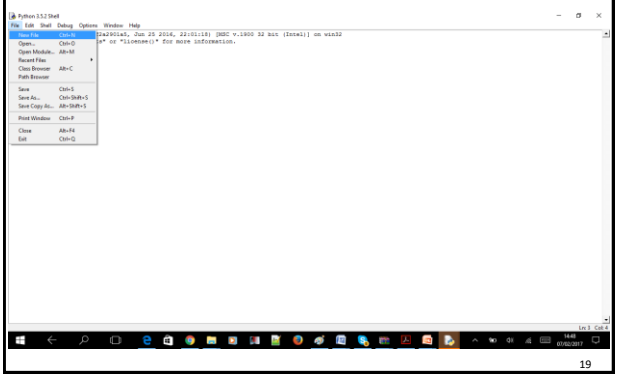
Editor e arquivos .py

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

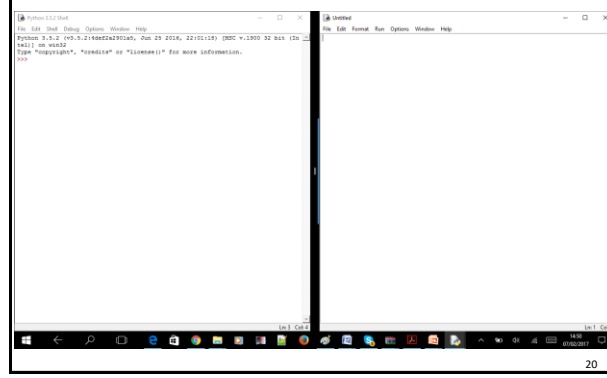
Solução:

- ✓ 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

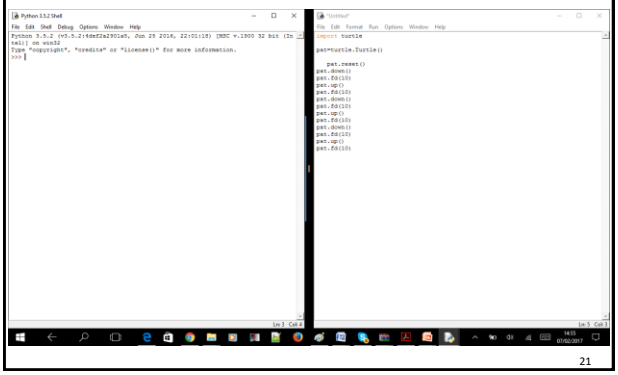
Editor IDLE (1/10)



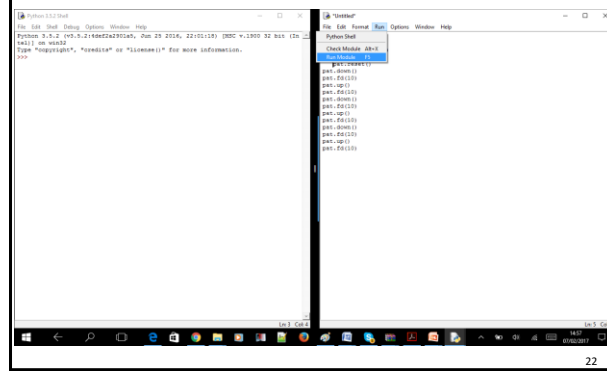
Editor IDLE (2/10)



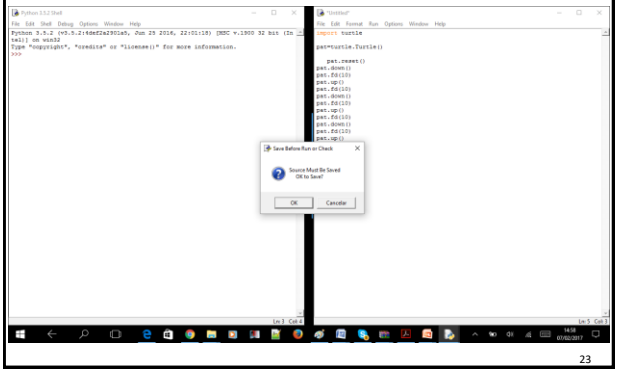
Editor IDLE (3/10)



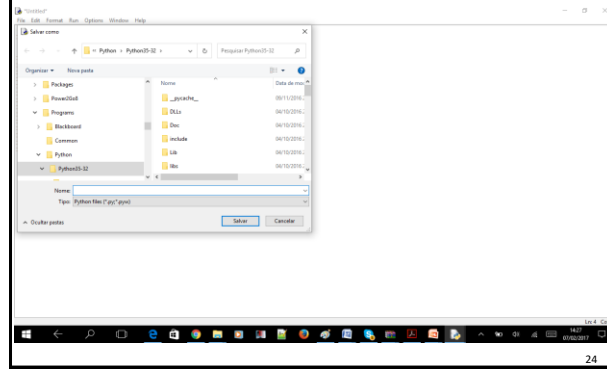
Editor IDLE (4/10)



Editor IDLE (5/10)



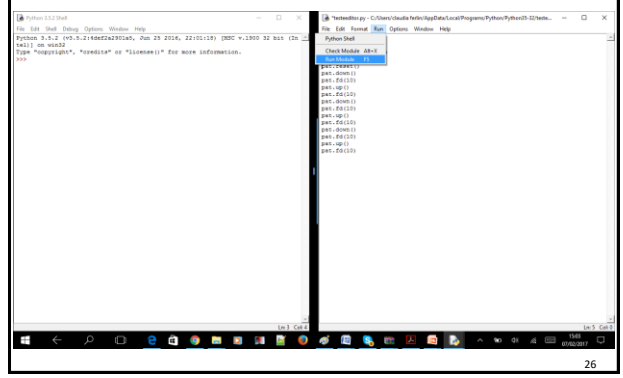
Editor IDLE (6/10)



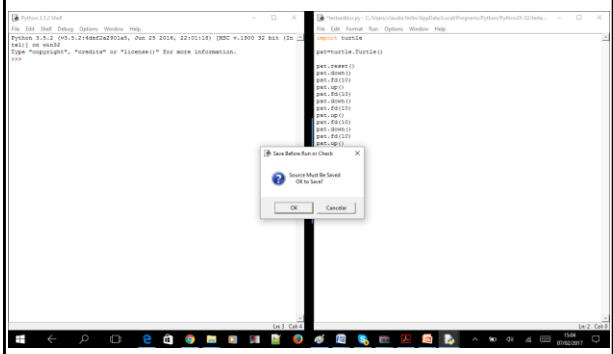
Editor IDLE (7/10)

```
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)
pat.down()
pat.fd(10)
```

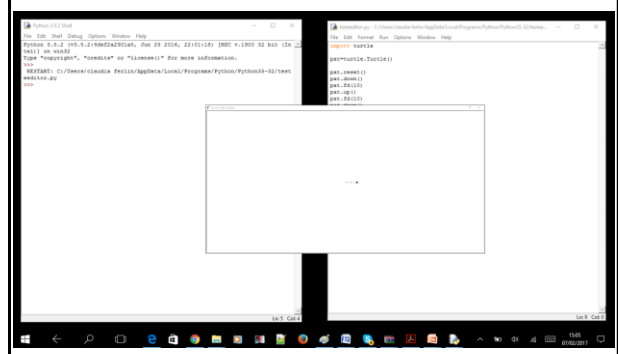
Editor IDLE (8/10)



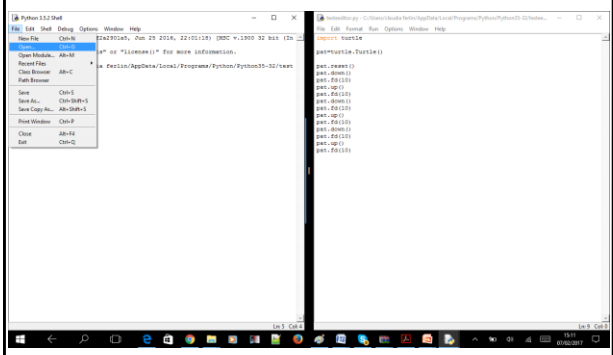
Editor IDLE (9/10)



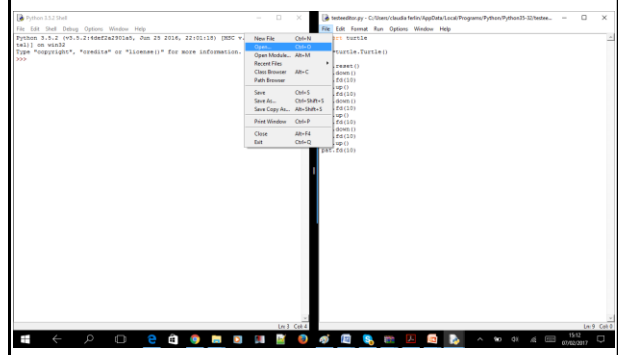
Editor IDLE (10/10)



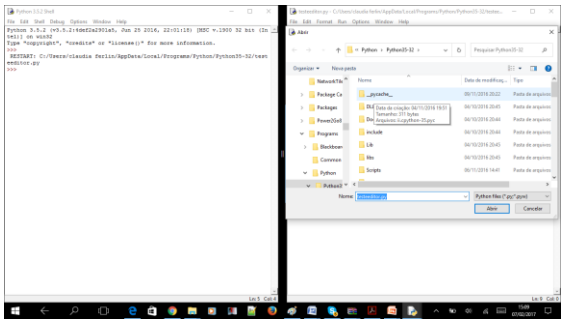
Editar arquivo existente (1/3)



Editar arquivo existente (2/3)



Editar arquivo existente (3/3)



Mãos na massa no Editor!!

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

Novas instruções:

Mudar a largura do traço:

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

Mudar a cor da Caneta:

`tartaruga.color(cor)` onde *cor* é o nome como 'red', 'green','blue',...

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)
```


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 o rabo
down	None	Abaixa o rabo
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 tartarua para a posição x,y
begin_fill	None	Use a posição atual para preencher um 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

Turtle e instruções adicionais

Desenhando um quadrado

Desenhar um quadrado de lado 100

O que deve ser feito?

Algoritmo do quadrado

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

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)
```


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

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

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

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

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 90º

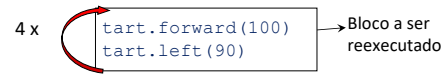
são realizadas 4 vezes.

Repetir 4 vezes:

riscar um lado

virar 90º

Bloco mais de uma vez!



Para reexecutar um bloco, volta-se para início.
Este ciclo deve ser interrompido quando a meta for atingida.

"Contar" quantas vezes o bloco já foi executado

- ✓ contagem de repetições assume valores sequenciais.
- ✓ O valor atual da contagem precisa ser guardado para ser utilizado.
- ✓ Valores são guardados em variáveis.

Instrução for (1/16)

for *VarValordaContagem* **in** *sequência de valores* **:**

<instrução $_1$>

....

<instrução $_n$>

Bloco a ser reexecutado

A cada iteração:

armazena o próximo valor da sequência na variável

executa as instruções do bloco (deslocadas para a direita - *identação*)

Para cada valor da sequência: executa o bloco

Instrução for (2/16)

for *VarValordaContagem* **in** *sequência de valores* **:**

<instrução $_1$>

....

<instrução $_n$>

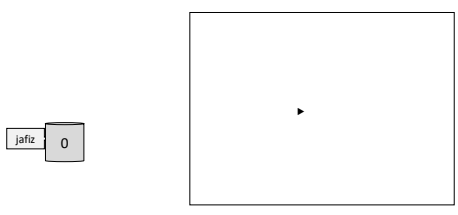
As instruções internas (que serão repetidas) devem estar identadas

- ✓ O bloco de comandos é delimitado pela identação.
- ✓ O fim do bloco é marcado pela primeira instrução não identada

OBS: *Não misture TAB e Espaços*

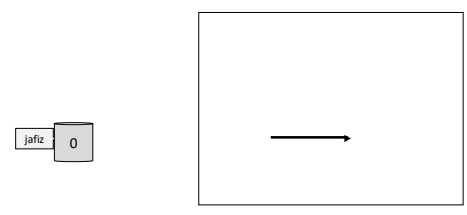
Instrução for (3/16)

```
for jafiz in [0,1,2,3]:
    tart.forward(100)
    tart.right(90)
```



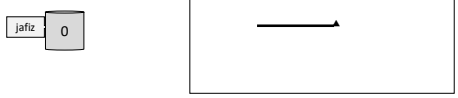
Instrução for (4/16)

```
for jafiz in [0,1,2,3]:
    tart.forward(100)
    tart.right(90)
```



Instrução for (5/16)

```
for jafiz in [0,1,2,3]:
    tart.forward(100)
    tart.right(90)
```

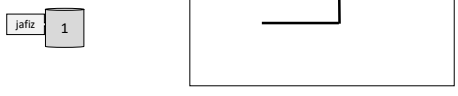


Instrução for (6/16)

```
for jafiz in [0,1,2,3]:
    tart.forward(100)
    tart.right(90)
```

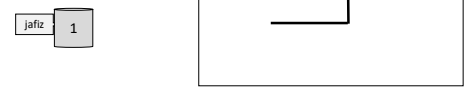
Instrução for (7/16)

```
for jafiz in [0,1,2,3]:
    tart.forward(100)
    tart.right(90)
```



Instrução for (8/16)

```
for jafiz in [0,1,2,3]:
    tart.forward(100)
    tart.right(90)
```

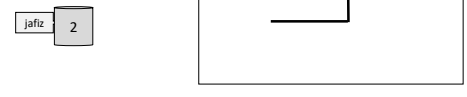


Instrução for (9/16)

```
for jafiz in [0,1,2,3]:
    tart.forward(100)
    tart.right(90)
```

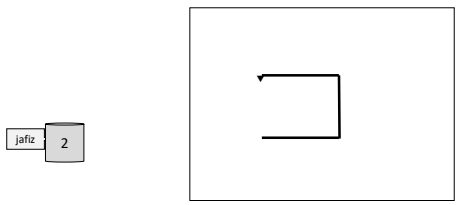
Instrução for (10/16)

```
for jafiz in [0,1,2,3]:
    tart.forward(100)
    tart.right(90)
```



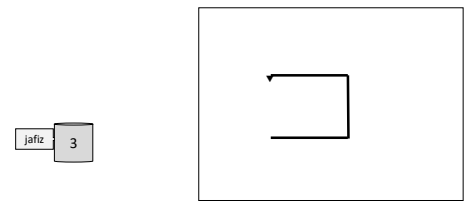
Instrução for (11/16)

```
for jafiz in [0,1,2,3]:
    tart.forward(100)
    tart.right(90)
```



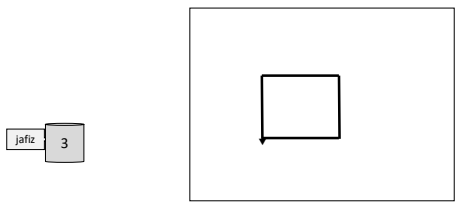
Instrução for (12/16)

```
for jafiz in [0,1,2,3]:
    tart.forward(100)
    tart.right(90)
```



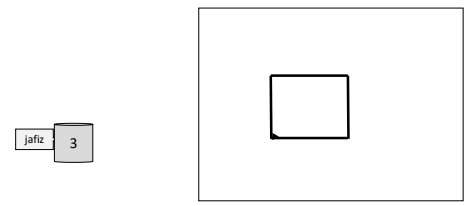
Instrução for (13/16)

```
for jafiz in [0,1,2,3]:
    tart.forward(100)
    tart.right(90)
```



Instrução for (14/16)

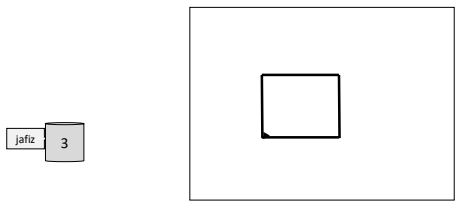
```
for jafiz in [0,1,2,3]:
    tart.forward(100)
    tart.right(90)
```



Instrução for (15/16)

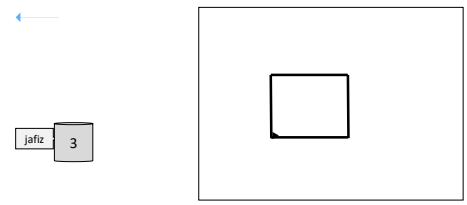
```
for jafiz in [0,1,2,3]:
    tart.forward(100)
    tart.right(90)
```

Repetiu 4 vezes



Instrução for (16/16)

```
for jafiz in [0,1,2,3]:
    tart.forward(100)
    tart.right(90)
```



Função range

Gera uma sequência de números inteiros que pode ser usada pela iteração do *for*

range(4)

0, 1, 2, 3

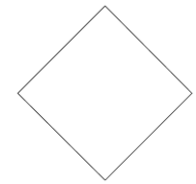
```
for jafiz in [0,1,2,3]:
    tart.forward(100)
    tart.right(90)
```

⇔

```
for jafiz in range(4):
    tart.forward(100)
    tart.right(90)
```


Revisitando o quadrado

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



1. Vira 45º
2. Desenha um quadrado de lado 100
3. Desvira 45º

Quadrado inclinado: uma solução

```
import turtle
pat = turtle.Turtle()
pat.right(45)                  # inclina a pat
for jafiz in range(4) :        #desenha o quadrado
      pat.forward(100)
      pat.right(90)
pat.left(45)                   # volta para direção original
```


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 teminar o desenho

```
tartaruga.end_fill()
```


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/

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
for jafiz in range(4):         # desenha o quadrado
      pat.forward(100)
      pat.right(90)
pat.end_fill()                 # termina o modo de preenchimento (qdo pinta)

pat.left(45)                   # volta para direção original
```

Exercícios de desenho

1. Desenhar um triângulo

Lembre-se da direção da tartaruga

2. Desenhar um círculo

   *Como a tartaruga desenha pelo perímetro → passo = π*raio/180*
3. Desenhar 2 círculos um dentro do outro
4. Desenhar a primeira letra do seu nome
5. Desenhar as figuras abaixo:



Criando funções

Mais quadrados!!!

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


Quadrado de novo!!!!

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

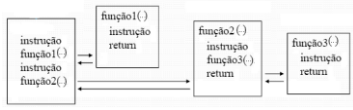
Criando novas funções

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()*, 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

Chamadas de funções

- ✓ Uma função em geral computa um ou mais valores a partir de valores recebidos, portanto, u*ma 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 a função termina, o controle retorna para o ponto de onde a função foi chamada (invocada)



Função DesenhaQuadrado (1/8)

Qual a tarefa desta função?

Função DesenhaQuadrado (2/8)

Qual a tarefa desta função?

Instruir uma tartaruga como desenhar um quadrado

Função DesenhaQuadrado (3/8)

Qual a tarefa desta função?

Instruir uma tartaruga como desenhar um quadrado

```
def DesenhaQuadrado ( ) :
```

indicá que será definida uma função

nome associado à função. *segue as mesmas regras de nomes de variáveis*

Função DesenhaQuadrado (4/8)

Qual a tarefa desta função? Desenhar um quadrado

Do que a função precisa para realizar sua tarefa?

```
def DesenhaQuadrado ( ) :
```


Função DesenhaQuadrado (5/8)

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 (  ) :
```


Função DesenhaQuadrado (6/8)

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

Função DesenhaQuadrado (7/8)

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?

Função DesenhaQuadrado (8/8)

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"""
    for jafiz in range(4):
        tart.forward(100)
        tart.left(90)
    return
```

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

DesenhaQuadrado em ação!

```
import turtle
def DesenhaQuadrado(tart):
    """ Quadrado lado 100"""
    for cont in range(4):
        tart.forward(100)
        tart.left(90)
    return

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


Criando uma nova função: Sintaxe

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

def *NomedaFunção* **(<** *parâmetros***>) :**
    <instrução$_1$>
    ...
    <instrução$_n$>
    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 retornando o que foi designado em <valor> para onde a função foi chamada

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

Múltiplos quadrados: uma solução

```
import turtle

def DesenhaQuadrado(tart):
  """ Quadrado lado 100"""
   for jafiz in range(4):
     tart.forward(100)
     tart.right(90)
     return
```


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 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)
```

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 direita

Deslocar para direita: uma solução

```
import turtle

def DesenhaQuadrado(tart):
    """ Quadrado lado 100"""
    for jafiz in range(4):
        tart.forward(100)
        tart.right(90)
    return

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


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)
```


Re-analisando os quadrados

```
t=turtle.Turtle()
t.reset()                   Quadrado Verde
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)
                            Quadrado Amarelo
```


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

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"""
    for jafiz in range(4):
        tart.foward(100)
        tart.left(90)
    return
```

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

Criar DesenhaQuadradoColorido (2/4)

Do que a função precisa para realizar sua tarefa?

De uma tartaruga

```
def DesenhaQuadradoColorido(tart
```


Criar DesenhaQuadradoColorido (3/4)

Do que a função precisa para realizar sua tarefa?

De uma tartaruga **e da cor**

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


Criar DesenhaQuadradoColorido (4/4)

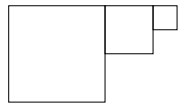
```
def DesenhaQuadradoColorido(tart,cor):
    """Quadrado lado 100 com cor"""
    tart.begin_fill() # inicia o modo de preenchimento
    tart.fillcolor(cor) # determina a cor
    for jafiz in range(4):
        tart.forward(100)
        tart.right(90)
    tart.end_fill() # pinta o interior do quadrado
    return
```


Usando DesenhaQuadradoColorido

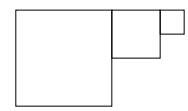
```
t=turtle.Turtle()
t.reset()
t.left(45)
DesenhaQuadradoColorido(t,'green') #desenha o 1º quadrado
t.right(45)
DeslocaDireita(t)
t.left(45)
DesenhaQuadradoColorido(t,'yellow') #desenha o 2º quadrado
```


Mãos na massa nos quadrados!

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

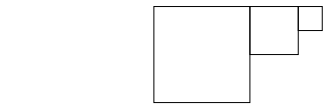


3 quadrados: solução (1/3)



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

3 quadrados: solução (2/3)

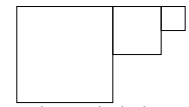


1. Desenhar quadrado de tamanho 100
2. Deslocar 100
3. Desenhar quadrado de tamanho 50
4. Deslocar 50
5. Desenhar quadrado de tamanho 25

Só muda o valor

Só muda o tamanho do lado

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 à direita

função DesenhaQuadrado (1/3)

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

```
def DesenhaQuadrado(tart):
    """Quadrado lado 100"""
    for jafiz in range(4):
        tart.foward(100)
        tart.left(90)
    return
```


função DesenhaQuadrado (2/3)

Do que a função precisa para realizar sua tarefa?

De uma tartaruga

```
def DesenhaQuadrado (tart
```


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)
```


Desenha quadrado: uma solução

```
def DesenhaQuadrado(tart,lado):
    """Quadrado lado recebido"""
    for jafiz in range(4):
        tart.forward(lado)
        tart.right(90)
    return
```

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)
```


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)
```

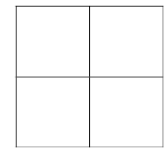
Modificar a função DeslocaDireita para aceitar como parâmetro o tamanho do deslocamento

DeslocaDireita: nova solução

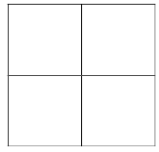
```
import turtle
def DesenhaQuadrado(tart,lado):
 ..............
 return
def DeslocaDireita(tart,dist):
    """Desloca distância p/direita"""
    tart.up()
    tart.fd(dist)
    tart.down()
    return

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


4 quadrados em um quadrado

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

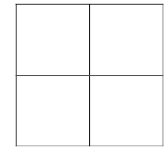


4 quadrados: resolvendo



Exibir linha a linha
OU
Exibir coluna a coluna?

4 quadrados: pensando a solução



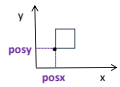
Exibir linha a linha
OU
Exibir coluna a coluna?

Tanto faz!!!

4 quadrados e eixos

Importante identificar duas coisas:

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



b) Onde <u>recolocar</u> a tartaruga para desenhar o <u>outro</u> quadrado dentro da linha (coluna) → deslocar a tartaruga na linha(coluna) um valor igual ao lado

4 quadrados: uma solução

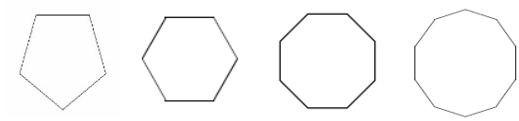
```
import turtle

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)
```


Funções de polígonos

Novas funções

O que muda nos códigos para desenhar estes polígonos?

Pentágono: Hexágono: Octógono: Decágono:



Polígonos: uma possibilidade

```
def Pentagono(tart,lado):
    """Desenha pentágono para tartaruga e lado recebidos """
    for i in range(5):
        tart.fd(lado)
        tart.rt(72)
    return

def Hexagono(tart,lado):
    """Desenha hexágono para tartaruga e lado recebidos """
    for i in range(6):
        tart.fd(lado)
        tart.rt(60)
    return
```

```
def Octogono(tart,lado):
    """Desenha octógono para tartaruga e lado recebidos """
    for i in range(8):
        tart.fd(lado)
        tart.rt(45)
    return
def Decagono(tart,lado):
    """Desenha um decágono para tartaruga e lado recebidos """
    for i in range(10):
        tart.fd(lado)
        tart.rt(36)
    return
```


Analisando o que muda

```
def Pentagono(tart, lado):
    """Desenha um pentágono """
    for i in range(5):
        tart.fd(lado)
        tart.rt(72)
    return
```

```
def Hexagono(tart, lado):
    """Desenha um hexagono """
    for i in range(6):
        tart.fd(lado)
        tart.rt(60)
    return
```

```
def Octogono(tart, lado):
    """Desenha um octogono """
    for i in range(8):
        tart.fd(lado)
        tart.rt(45)
    return
```

```
def Decagono(tart, lado):
    """Desenha um decagono """
    for i in range(10):
        tart.fd(lado)
        tart.rt(36)
    return
```

A quantidade de lados e o ângulo.

Polígono regular: Nº de lados * ângulo interno = 360

DESAFIO: Criar uma função que desenha um polígono regular com tartaruga, número de lados e tamanho do lado recebidos

Uma solução para o desafio

```
import turtle
def PoligonoRegular(tart, lado, nLados):
    """ Traça um poligono regular p/uma tartaruga, comprimento e nº de lados """
   angulo= 360/nLados
   for i in range(nLados):
      tart.fd(lado)
      tart.rt(angulo)
   return

pat=turtle.Turtle()
DesenhaPoligonoRegular(pat,100,9)          #desenha um eneágono
```


Polígonos fechados

- Criar uma função que desenhe um polígono com o comprimento do lado, número de lados e o ângulo também recebidos
- Caso o nº de lados * ângulo < 360º a figura desenhada não será fechada



Polígonos fechados: uma solução

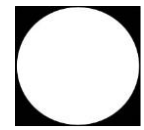
```
import turtle
def DesenhaForma (t,lado,num_lados,angulo):
 """Desenha sequências de segmentos eventualmente produzindo formas fechadas. """
  for i in range(num_lados):
      t.forward(lado)
      t.left(angulo)
  return


t=turtle.Turtle()
DesenhaForma(t,100,60,186)

DesenhaForma(t,100,5,144)

DesenhaForma(t,100,8,135)
```



Exercício do Ladrilho

Faça uma função para desenhar o ladrilho abaixo cuja área é fornecida.

Por exemplo: área = 10.000. Logo, deve-se construir um quadrado 100x100 e um círculo de raio 50



Ladrilho: Desenvolvendo a solução

Sequência de passos:

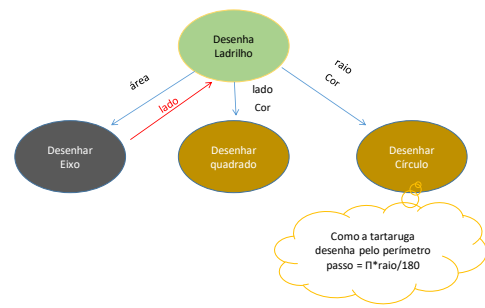
1. Desenhar Quadrado
2. Desenhar Círculo

A função DesenhaQuadrado desenvolvida espera o lado do quadrado: def DesenhaQuadrado(tart,lado):
.....
e foi fornecido a área.

Sequência de passos:

1. Calcular o lado
2. Desenhar Quadrado
3. Desenhar Círculo

Ladrilho: ideia da solução



Ladrilho: uma solução

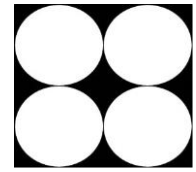
```
import turtle
import math
def desenhaQuadradoCor(t,lado,cor):
    """Quadrado preenchido dado lado e cor"""
    t.begin_fill()
    t.fillcolor(cor)
    for jafiz in range(4):
        t.forward(lado)
        t.right(90)
    t.end_fill()
    return
)

def detLado(area):
   """Lado Quadrado dado área"""
   return area**0.5
```

```
def desenhaCircCor(t,raio,cor):
    """Circulo preenchido dado raio e cor"""
    passo = 3.1415*raio/180
    t.begin_fill()
    t.fillcolor(cor)
    for jafiz in range(360):
        t.forward(passo)
        t.right(1)
        jkj
    t.end_fill()
    return


def desenhaLadrilho(area):
  """Ladrilho Quadrado com círculo
     preenchido dado area"""
  lado=detLado(area)
  desenhaQuadradoCor(t,lado,'black')
  t.goto(lado/2,0)
  desenhaCircCor(t,lado/2,'white')
  return
```

```
t=turtle.Turtle()
desenhaLadrilho(10000)
```


Desafio: 4 círculos em um quadrado

Desenhe



Retângulos e triângulos

A. Faça uma função que receba a altura de um retângulo, o numerador e o denominador de uma fração. Esta função retorna o valor da fração da altura recebida. Exemplo: recebe: alt=90, num=2, den=3 ; retorna 60

B. Faça uma função para desenhar um retângulo fracionado em 3 partes (tartaruga,altura e comprimento do retângulo são recebidos) conforme esquema abaixo. Use a função do ítem a) para calcular a altura do retângulo a ser desenhado

Retângulos e triângulos: solução

```
import turtle
def Retangulo(t,alt,compr,cor):
  """ Retang.lados fornec"""
  t.begin_fill()
  t.fillcolor(cor)
  for jafiz in range(2):
    t.forward(compr)
    t.left(90)
    t.forward(alt)
    t.left(90)
  t.end_fill()                  #
  return
def desloca(tart,dist):
   """ Desloca dist p/direita"""
  tart.up()
  tart.fd(dist)
  tart.down()
  return
def Particiona(lado,num,den):
  """ Fração de lado"""
  return(lado*num/den)
```

```
def RetFracionados(t,alt,larg):
  fr1 = Particiona(alt,1,3)
  fr2 = Particiona(alt,2,3)
  Retangulo(t,alt,larg,'blue')
  desloca(t,larg+20)

  Retangulo(t,alt,larg,'blue')
  Retangulo (t,fr2,larg,'orange')
  desloca(t,larg+20)

  Retangulo (t,alt, larg,'blue')
  Retangulo (t,fr2,larg,'orange')
  Retangulo (t,fr1,larg,'yellow')
  return

t=turtle.Turtle()
t.up()
t.goto(-210,0)
t.down()
RetFracionados(t,90,60)
```

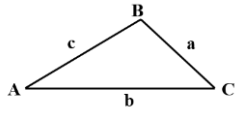

Mais triângulos

Faça uma função para desenhar um triângulo, dados seus 3 lados.

É garantido que $lado_a < lado_b + lado_c$

$lado_b < lado_a + lado_b$

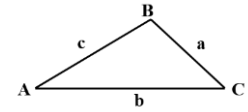
$lado_c < lado_a + lado_b$



Ângulos e triângulos

Para desenhar o triângulo, precisa-se dos lados, a,b,c (fornecidos)

e dos ângulos A,B,C



Pela lei dos cossenos, pode-se calcular os ângulos:

$$Cos\ \hat{A} = \frac{b^2+c^2-a^2}{2bc} \qquad Cos\ \hat{B} = \frac{a^2+c^2-b}{2ac} \qquad Cos\ \hat{C} = \frac{a^2+b^2-c^2}{2ab}$$

$$\hat{A} = a\cos(\cos\hat{A})$$

Fórmulas e triângulos

O que muda nestas fórmulas?

$$Cos\ \hat{A} = \frac{b^2+c^2-a^2}{2bc} \qquad Cos\ \hat{B} = \frac{a^2+c^2-b}{2ac} \qquad Cos\ \hat{C} = \frac{a^2+b^2-c^2}{2ab}$$

Reescrevendo:

$$\cos Ang_1 = (\ lado_2^2 + lado_3^2 - lado_1^2\ ) / (\ 2 * lado_2 * lado_3\ )$$

↓

Função que recebe 3 lados de um triângulo (o,p,q) e retorna o ângulo Ô

DesenhaTri: uma solução

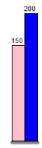
```
import turtle
import math
def calcAngulo(lado1,lado2,lado3):
  """ ângulo do lado1"""
  cosang1 = (lado2**2+lado3**2 - lado1**2) /(2*lado2*lado3)
  ang1=math.degrees(math.acos(cosang1))
  return ang1

def desenhaTri(t,a,b,c,angC):
  """ Desenha triangulo dados 3 lados e uma ângulo"""
  angA = calcAngulo(a,b,c)
  angB = calcAngulo(b,a,c)
  angC = calcAngulo(c,b,a)
  t.fd(b)
  t.left(180-angC)
  t.fd(a)
  t.left(180-angB)
  t.fd(c)
  t.setheading(0)
  return
```

```
t=turtle.Turtle()
desenhaTri(t,90,180,260)
```


Gráficos de barras

Escreva um script para desenhar um gráfico de barras, com cores distintas, da distribuição entre os gêneros dos jovens que responderam uma pesquisa de satisfação.

Foram entrevistadas 350 jovens, 150 eram do sexo feminino e 200 eram do sexo masculino



Eixos e duas barras

Dois sub-problemas para resolver:

1. O eixo
2. As barras:
   - 2.1. Retângulo
   - 2.2. Título com valor

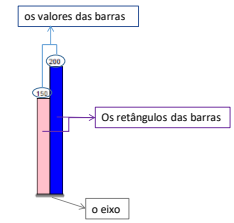


Gráfico de barras: ideia da solução

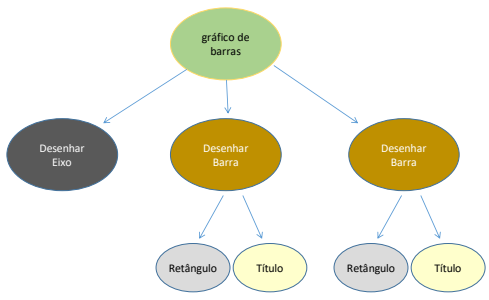
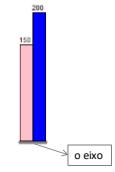


Gráfico de barras: eixo

Desenhar o eixo x

1. Qual o tamanho da reta?
2. Qual a cor e espessura da linha?
3. Onde posicionar a tartaruga no início?
4. Qual a orientação?



Eixo e posicionamento

Desenhar o eixo x

1. Qual o tamanho da reta?

   2 x largura das barras

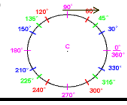
2. Qual a cor e espessura da linha?

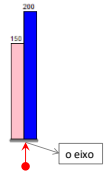
   Cinza, 5 pontos

3. Onde posicionar a tartaruga no início?

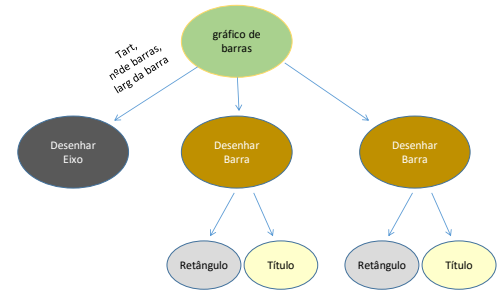
   Se quiser centralizado na janela, deve deslocar a tartaruga para a esquerda, tamanho/2 pontos, do ponto (0,0)

4. Qual a orientação?





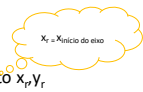
Eixo ok, agora barras

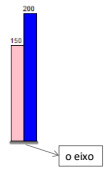


Barras em duas alturas e cores

Desenhar as barras

1. Barra rosa
   1. Desenhar retângulo no pto x_r,y_r com altura == nº de mulheres
   2. Escrever valor
2. Barra azul
   1. Desenhar retângulo no pto x_a,y_a com altura == nº homens
   2. Escrever valor

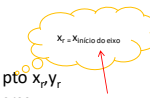




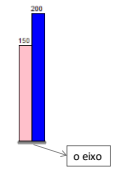
Posicionando as barras

Desenhar as barras

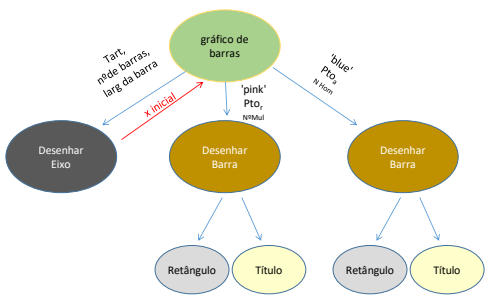
1. Barra rosa
   1. Desenhar retângulo no pto x_r,y_r com altura == nº de mulheres
   2. Escrever valor
2. Barra azul
   1. Desenhar retângulo no pto x_a,y_a com altura == nº homens
   2. Escrever valor



???????? Calculado pela função que desenha o eixo



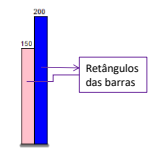
Evoluindo com a solução



Barras: problemas a resolver

Desenhar o retângulo da barra

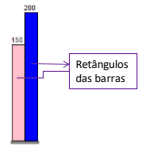
1. Qual a altura do retângulo?
2. Qual a largura do retângulo?
3. Qual a cor e espessura da linha?
4. Qual a cor do preenchimento?
5. Onde posicionar a tartaruga no início?
6. Qual a orientação?



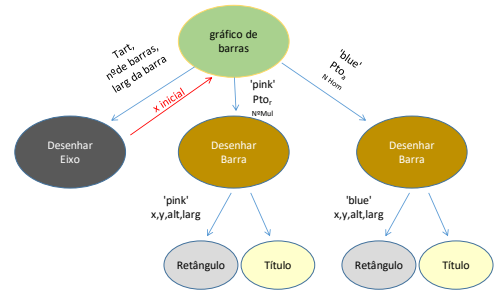
Barras: resolvendo problemas!

Desenhar o retângulo da barra

1. Qual a altura do retângulo?
   Número de pessoas
2. Qual a largura do retângulo?
   20
3. Qual a cor e espessura da linha?
   preto, 1 ponto
4. Qual a cor do preenchimento?
   azul/rosa (depende do gênero)
5. Onde posicionar a tartaruga no início?
   Sobre **um ponto do eixo x**,
6. Qual a orientação?



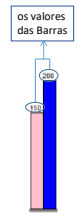
Desenvolvendo a solução



Identificando barras

Escrever título da barra

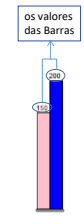
1. Qual a cor e tipo de fonte?
2. Onde posicionar a tartaruga no início?
3. Qual a orientação?



Textos na barras: instrução write

Escrever título da barra

1. Qual a cor e tipo de fonte?
   Padrão
2. Onde posicionar a tartaruga no início?
   Sobre o retângulo da barra: x da barra e
   y= altura da barra
3. Qual a orientação?



Nova instrução:

Escrever um texto:
`tartaruga.write(texto)` : Escreve o texto na posição corrrente

Solução encaminhada

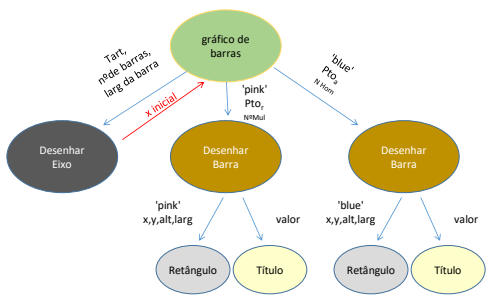


Gráfico de Barras: uma solução

```
def plotbarras(t,valor1,valor2):
    """ Plota duas barras de altura valor1 e valor2"""
    largura=20
    posX=ploteixo(t,2,largura) # Desenha eixo
    t.down() # Prepara para desenhar barras
    t.width(1)
    t.setheading(90) # orientação a 0 graus
    # Desenha barra feminina
    barra(t,posX,2.5, 150,largura,'black','pink')
    # Desenha barra masculina
    barra(t,posX+largura,2.5,200,largura,'black','blue')
    t.hideturtle()
    return

# Ativando o gráfico de barras
t=turtle.Turtle()
plotbarras(t,150,200)
```

Barras: função para eixo

```
def ploteixo(t,nbarras,larg):
    """ Desenha eixo X"""
    #Define o eixo dos Xs#
    tamanho_reta=nbarras * larg
    # Define parâmetros do eixo dos Xs
    turtle.width(5)
    turtle.color('gray')
    turtle.setheading(0)
    turtle.goto(-tamanho_reta/2, 0)
    # desenha eixo
    turtle.down()
    turtle.forward(tamanho_reta)
    # reposiciona a tartaruga no início do eixo
    turtle.goto(-tamanho_reta/2,0)
    # retorna posição atual do X
    return -tamanho_reta/2
```

Retorna o posX inicial da reta

Barras: função para desenho

```
def barra(t,posX,posY, alt, larg, corL,corC):
    """ Desenha uma barra na posição de altura tamanho c/valor alt """
    # posiciona
    t.up()
    t.goto(posX,posY)
    # desenha retangulo da barra
    retangulo(t,alt,larg,corL,corC)
    # escreve o valor da barra
    escreve(t,posX,posY+alt,alt)
    return
```


Barras: função retângulo

```
def retangulo(t, alt,larg,corLinha,corRet):
    """ Retangulo com lados e cor recebidos"""
    t.setheading(90)
    # parâmetros
    t.color(corLinha,corRet)
    # desenha
    t.down()
    t.begin_fill()
    for i in range(2):
        t.forward(alt)
        t.right(90)
        t.forward(larg)
        t.right(90)
    t.end_fill()
    t.hideturtle()
    return
```


Barras: função para texto

```
def escreve(t,posX,posY,valor):
    """ escreve valor no pto"""
    # posiciona
    t.goto(posX,posY)
    t.setheading(0)
    # escreve valor da barra
    t.write(valor)
    return
```


Exercícios adicionais para desenho

Construa funções para:

1. Desenhar esta sequência de quadrados, dado o lado:
3. Desenhar um retângulo, dados os lados

4. Desenhar o mosaico ao lado (octógno com quadrado)
5. Desenhar um círculo, dado o raio. *Lembre-se que a tartaruga desenha pelo perímetro → passo = π*raio/180*
6. Desenhar uma pirâmide com triângulos, dados o lado e nº de camadas
7. Desenhar um cacho de uvas, dado o raio e o nº de camadas
8. Desenhar a bandeira de um país
9. Desenhar um boneco com os símbolos geométricos
10. Desenhar a figura ao lado dado a área do quadrado

11. Desenhar um quadro do cubismo