Unidade 9: Módulo Turtle em TI-Python | **Aplicação: Construção de uma figura complexa**

Nesta lição de aplicação irá mobilizar os conhecimentos trabalhados nas anteriores lições por forma a construir um programa que lhe permita a conceção de um projeto.

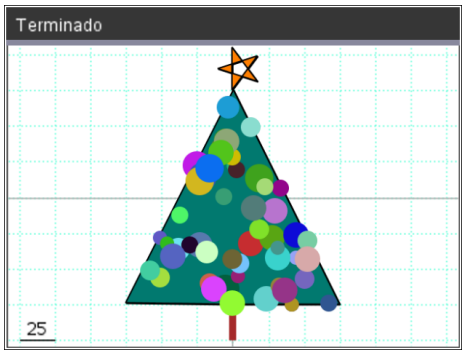
Objetivos:

- Aplicar os conceitos trabalhos nas lições anteriores.
- Construir uma figura complexa com recurso à programação.

0. Introdução

Nas atividades das três lições anteriores vimos como usar o módulo **Turtle**, num programa TI-Python, na construção de uma figura geométrica, sendo que foi necessário antecipadamente realizarem-se cálculos matemáticos.

Para melhor perceção do programa usamos, habitualmente, comentários inseridos entre o código, sempre que oportuno, e também procuramos que cada tarefa complexa fosse dividida num certo número de tarefas simples.



Nesta lição iremos construir, geometricamente, uma árvore de Natal, sendo que o nosso programa principal se dividirá nas seguintes funções mais simples:

- desenhar e colorir um triângulo isósceles;
- desenhar o pé da árvore;
- desenhar e colorir uma estrela;
- enfeitar e reluzir a árvore

OBSERVAÇÃO:

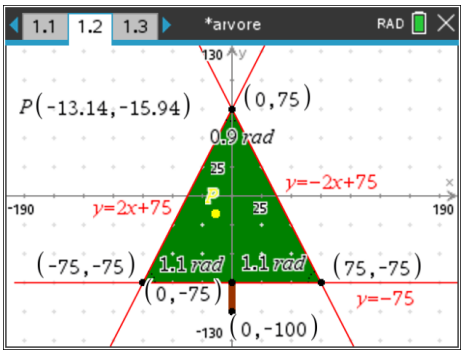
A programação com recurso a funções, elementos mais simples de código, promove a realização de projetos complexos em ambientes de trabalho colaborativo entre alunos.

1. Criação do projeto

Todos os projetos têm as suas especificidades. Neste, em que utilizaremos o módulo **Turtle** e mostrará um objeto gráfico, teremos os constrangimentos resultantes do tamanho do ecrã (grelha unitária de 25 pixéis).

As especificidades do projeto são:

- a decoração da árvore, que consiste na colocação de pontos coloridos, usando a instrução **t.dot(**diâmetro**)**, sendo o tamanho e posição aleatórios;
- os pontos coloridos, que devem estar contidos num triângulo isósceles (a árvore);
- a representação anterior será obtida utilizando um software de geometria dinâmica no qual o sistema de coordenadas ortonormais é idêntico à grelha do módulo **Turtle**, o que facilitará trabalho;



- os pontos de coordenadas (–75, –75); (75, –75) e (0, 75) serão os vértices do triângulo;
- as retas de equação reduzida y = 2x + 75, y = −2x + 75 e y = −75 delimitarão o domínio plano de representação dos pontos coloridos (os enfeites).

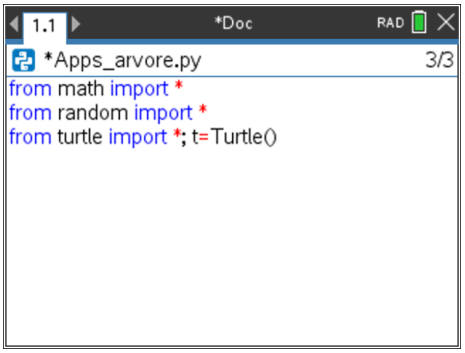
SUGESTÃO:

De acordo com as pretensões, é possível construir uma árvore cuja largura da base seja variável, sendo que as coordenadas (–a, –b) do primeiro ponto, as equações das linhas e as amplitudes dos ângulos podem ser calculadas, cada um, através de uma função. Também podemos considerar a variação da posição da origem do referencial e representar várias árvores no mesmo gráfico.

2. Criação do programa

Na aplicação TI-Python:

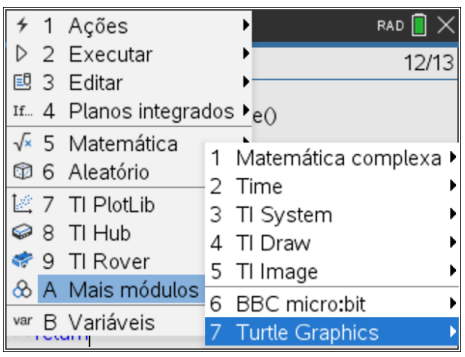
- abra um novo programa e designe-o por **Apps_Arvore**;
- prima na tecla **[Menu]** e importe os módulos de **Matemática**, **Aleatório** e **Turtle Graphics**, necessários para este projeto;
- o módulo **Aleatório** permitirá obter aleatoriamente as posições e o número de objetos coloridos (enfeites).



3. Desenho do pé da árvore

Comecemos, então, por construir a função que desenhe o pé da árvore. Assim, coloquemos as instruções em Python para:

- ocultar a tartaruga – **t.hideturtle()**;
- colocar o cursor, sem escrever – **t.penup(),** na posição inicial (0,– 100) – **t.goto(0, –100)**;
- baixar a caneta para se poder desenhar – **t.pendown()**;
- definir a cor do lápis como castanho, correspondente ao código RGB (165, 42, 42) usando a instrução **t.pencolor(165, 42, 42)**;
- de seguida orientar a tartaruga para cima – **t.setheading(90)**;
- para traçar o pé da forma mais simples e ficar bem visível, escolher a espessura mais grossa do lápis – **t.pensize(3)**;
- desenhar o pé com comprimento de 25 pixéis – **t.forward(25)**.



Todas as instruções anteriores estão disponíveis no módulo **Turtle**, premindo tecla **[Menu]**, depois a opção **A: Mais Módulos**, seguido de **7: Turtle Graphics** e, por fim, escolhendo a opção pretendida.

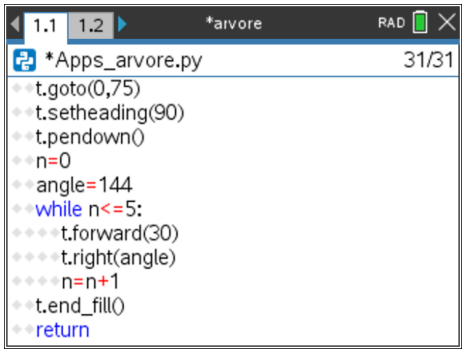
4. Desenhar a estrela do cimo da árvore

Para desenhar a estrela vai ser criada uma função que designaremos por **estrela()**. Poderemos aqui, se necessário, revisitar o guião da **Lição 1**, na qual foram desenhamos polígonos regulares.

Nesse caso, necessitaríamos de usar um programa num programa, procedendo-se à sua importação usando-se a instrução **from PROGRAMA import*** disponível no catálogo.

Neste caso, usaremos um novo programa, realizando as etapas:

- fixar a estrela no topo da árvore no ponto das coordenadas (0, 75) – **t.goto(0,75)**;
- com a instrução **t.fillcolor()** completar a coloração dos ramos da estrela;
- preencher os ramos com uma tonalidade laranja de código RGB (255,127,0) – **t.fillcolor(255,127,0)**;
- em cada iteração, mover a tartaruga **30 pixéis** para a frente e rodar para a direita por **144 graus**.
- cada um dos ângulos interiores da estrela será de **36 graus**;
- com **5 iterações** ficará perfeitamente construída a estrela, usando o ciclo de controlo **while**.

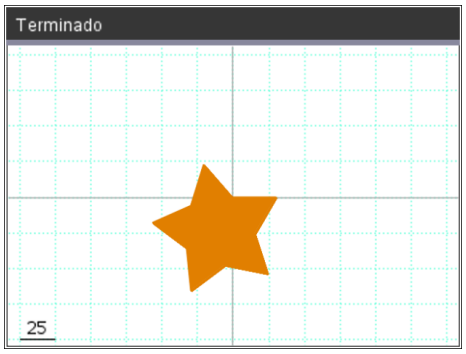


5. Desenhar uma dada estrela em função do seu tamanho

SUGESTÃO:

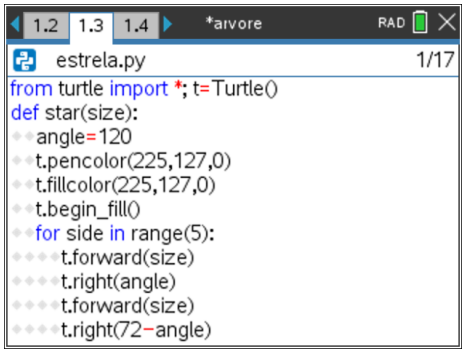
Como desenhar uma estrela completa?

Na verdade, o que se pretende desenhar é um polígono, partindo de 2 linhas para cada lado. Os ângulos devem ser incrementados em 72° (360/5).



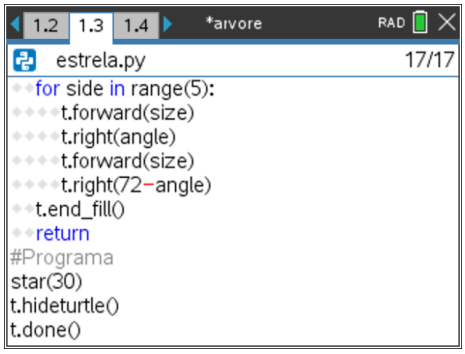
Vejamos como construir uma função que permita construir uma estrela sólida de uma dada cor. Dependendo do tempo que pretender disponibilizar ao seu projeto, pode optar por este triplo de estrela no cimo da árvore de Natal ou até como um dos elementos de enfeite. Vejamos as principais etapas do código dessa função que se encontra na imagem ao lado:

- a função terá um argumento, o tamanho da estrela - **size**, designando-se a função por **star()**, portanto chamando a função com **star(size)**;
- a estrela será preenchida por uma cor, neste caso cor de laranja cujo código RGB é (225,127,0), usando-se as instruções **t.pencolor(225,127,0)**, para escrever a laranja, e **t.fillcolor(255,127,0)**,



para colorir a laranja;

- de seguida usa-se, nos **Planos Integrados**, o ciclo de controlo **For index in range(size)**, em serão traços dois segmentos consecutivos da estrela;
- a repetição 5 vezes das instruções deste ciclo permitirá encerrar o polígono estrela e o preencher na cor pretendida;
- para usar esta função, por fim, fazemos a correr para um tamanho de 30 pixéis – **star(30)**;
- por fim, escondemos a tartaruga – **t.hideturtle()**.

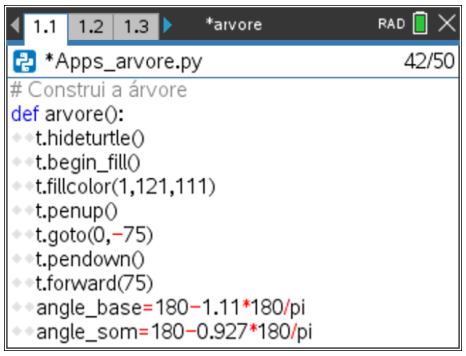


6. Desenhar a árvore

A tarefas para esta função consiste em construir um triângulo isósceles simples, como já referimos na implementação do projeto.

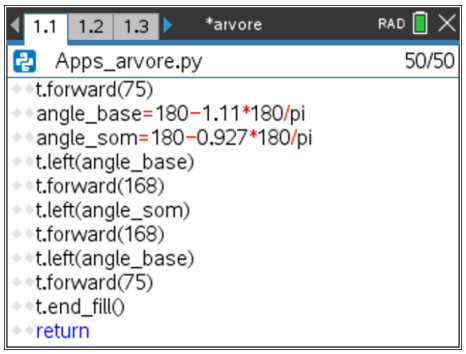
Todas as medidas (amplitudes dos ângulos, comprimento dos lados, ...) resultam de trabalho preparatório com ambiente de geometria dinâmica.

O triângulo isósceles, a nossa árvore, será preenchido pela cor verdade árvore cujo código RGB é (,121,111) – **t.fillcolor(1,121,111)**.



SUGESTÃO:

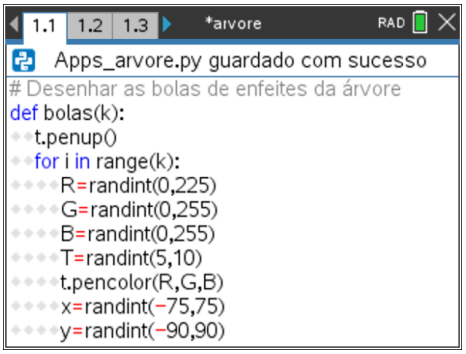
A não utilização da instrução de graus - **degrees()** do módulo **Matemática** é intencional. Pretende-se que os jovens estudantes devam ser capazes de converter uma medida de amplitude de ângulo de uma unidade para outra, neste caso de radianos para graus. Para isso devem usar a relação matemática, lei de conversão, de um sistema de medida para outro.



7. Desenhar as bolas (enfeites) de Natal

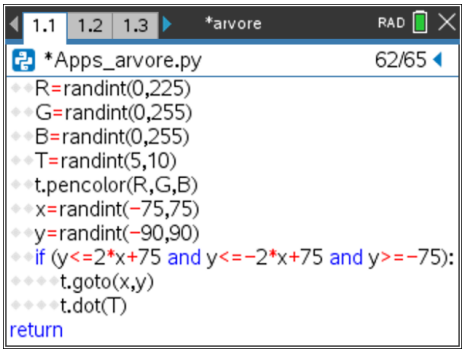
Vamos agora criar a função que nos permitirá desenhar as bolas de enfeites da árvore de Natal, sendo de tamanhos e cores aleatórias. Recordar que para usar as funções aleatórias foi necessário importar, no início do programa, o módulo **Aleatório**. A função, designada por **bolas()**, dependerá de um argumento **k** que define o número de bolas a representar, portanto usaremos **bolas(k)**. Temos ainda que:

- para definir as cores usamos a função **randint()** que permite gerar



inteiros aleatórios, neste caso entre 0 e 255, que definirá a porção de vermelho, **R**, verde, **G**, e de azul, **B**, em cada bola;

- o tamanho da bola, mais propriamente o seu diâmetro, é obtida de forma aleatória com a função aleatória **randint(5,10)**;
- também a posição da bola, através das suas coordenadas em pixéis, são obtidas através da mesma função aleatória **randint()**, desta feita para valores entre –75 e 75 para a abcissa e –90 e 90 para a ordenada;
- por fim, através de um ciclo de controlo, neste caso **If …**, é garantida a representação da bola caso as suas coordenadas satisfação as condições que definem o domínio plano que define a árvore.



8. O programa principal

Estão terminadas todas as funções mais simples em que não dividimos o nosso projeto de construção de uma Árvore de Natal, bastará agora colocar no programa a execução destas funções.

Assim, embora sendo indiferente a ordem com que colocarmos as funções se não usarmos pausas, aqui vamos usar uma ordem especifica e com ligeiras pausas de 1 segundo.

A instrução colocada no final do nosso programa **t.done()** deixa a imagem exibida. Pressionando-se qualquer tecla desaparecerá a imagem e surgir a linha de entrada do Shell Python com >>>.

No editor do programa no Python pode gravar e verificar o programa usando o atalho **CTRL + B**, e poderá executá-lo através do atalho **CTRL + R**. No Shell Python podem também executá-lo com o atalho **CTRL + R**.