ATIVIDADES CodeWeek | Projeto CAP3R

Adivinha a bandeira!

Duração estimada: 2 aulas de 45 minutos cada

Faixa etária: Alunos do ensino secundário

Secundário

Aptidões, competências e objetivos de aprendizagem:

- Desenvolver e testar um programa com Python
- Utilizar condições (if-else) para tomar decisões
- Utilizar ciclos para executar um bloco de instruções enquanto a condição for verdadeira
- Escrever e chamar funções (partes de código reutilizáveis)
- Utilizar o módulo *Turtle* e cálculos matemáticos para desenhar (plano cartesiano, coordenadas, ângulos)

Atividades e tarefas:

1. Alunos: desenvolvem o jogo *Adivinha a bandeira!* utilizando a linguagem de programação Python.
2. Professor: acompanha e faz a demonstração da atividade, auxilia quando necessário.

Material necessário:

Um computador por aluno com ligação à Internet ou computadores sem Internet com o ambiente integrado de aprendizagem e desenvolvimento (IDLE) do Python instalado. Caso não disponha de um número suficiente de computadores, incentive os alunos a trabalharem em pares ou em pequenos grupos (2-4 alunos).

Espaço de aprendizagem: Sala de informática

Descrição da atividade:

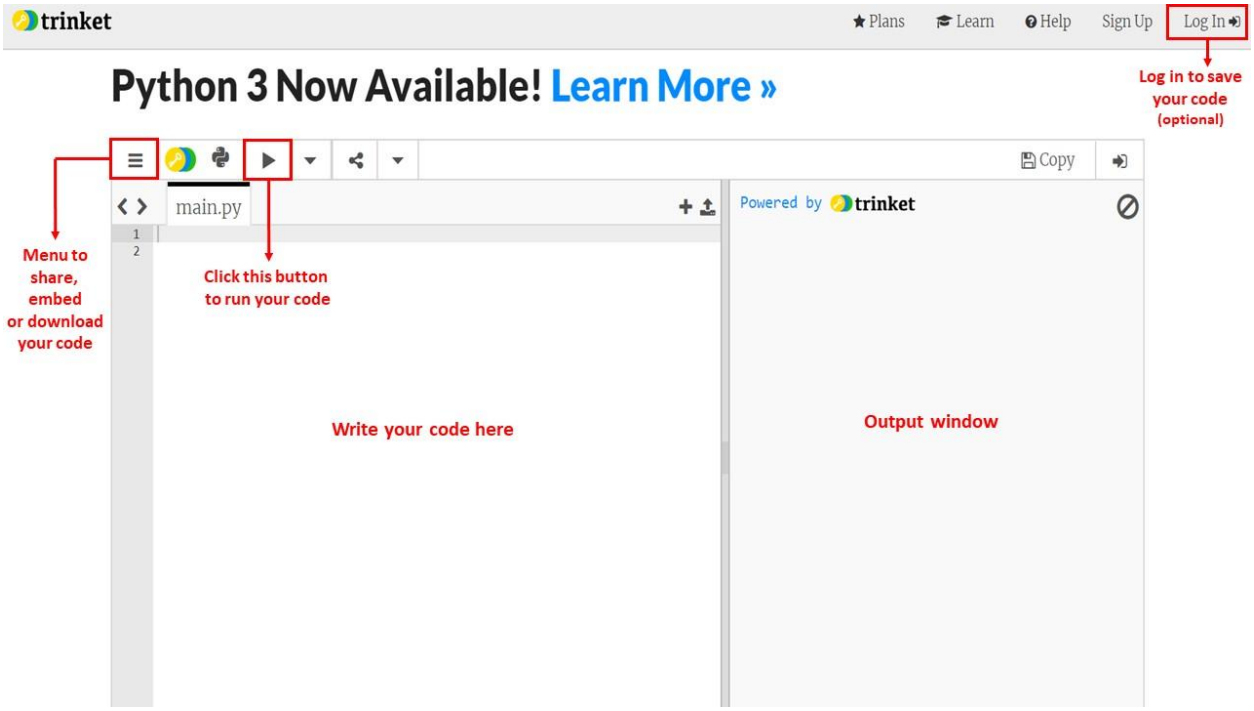
Nesta atividade, os alunos irão criar o jogo Adivinha a bandeira!. Vamos desenhar uma bandeira e o utilizador tem de adivinhar a que país pertence a bandeira exibida no ecrã. Se a resposta estiver certa, o utilizador ganha um ponto. Caso contrário, perde pontos e vidas.

**Sugestão: Para o ajudar no seu primeiro programa em Python, incluímos algumas orientações sob a forma de comentários no código a seguir. As orientações estão assinaladas a verde e começam sempre com #. Pode também deixar comentários como este no seu código para ajudar os alunos.*

```
#comment using a hashtag
```

Preparação:

Aceda ao sítio Web https://trinket.io/turtle e está pronto para criar o seu primeiro projeto. Pode escrever a sua primeira linha de código na janela do lado esquerdo. Depois de clicar no botão Run (executar), verá o resultado (a forma como os utilizadores veriam o seu código) na janela do lado direito.



Parte 1: Desenhar as bandeiras (45 minutos)

Passo 1: Adicione a tartaruga e as variáveis

Para desenhar bandeiras, utilizaremos o módulo *turtle*. Os módulos têm de ser importados para o nosso programa através da inserção de uma linha de código no início do programa:

```
import turtle
```

Importaremos também o módulo *random*, que utilizaremos posteriormente no nosso programa para gerar desenhos de bandeiras aleatoriamente.

```
import turtle
import random
```

Vamos dar um nome à nossa tartaruga. Este passo torna-se especialmente útil se tivermos mais do que uma tartaruga no nosso programa. Vamos chamar Bob à nossa tartaruga.

```
bob = turtle.Turtle() #give name to a turtle
```

Em seguida, definiremos algumas variáveis. As variáveis são utilizadas para armazenar valores e fornecer dados ao computador para processamento. Podem armazenar dados de diferentes tipos, que têm funções diferentes. As nossas variáveis aqui são *points* (pontos) e *life* (vida) e cada uma delas é um número.

No início do jogo, temos 0 pontos e 3 vidas.

```
points = 0
life = 3
```

Passo 2: Adicionar uma função

Vamos começar por desenhar a bandeira alemã em conjunto. Vamos desenhá-la dentro de uma função. As funções são blocos de códigos reutilizáveis, que apenas são executados quando são chamados. A função é definida pela palavra-chave *def*, seguida do nome que atribuímos à nossa função:

```
def germany():
```

Para chamar uma função, utilize o nome da função, seguido de parênteses (). Pode chamar a função para verificar o seu desenho. Adicione esta linha de código no fim do seu programa atual, fora da função.

```
germany() #calling the function
```

Passo 3: Desenhar formas

A bandeira alemã é constituída por três retângulos de igual tamanho. Vamos começar pelo retângulo preto.

Para preencher formas com cores, utilizaremos as instruções *begin_fill() e end_fill().* Pode consultar a lista de cores aqui: https://trinket.io/docs/colors.

```
def germany(): #define function which stores drawing of German flag

  bob.fillcolor("black") #fill the turtle with some color
  bob.begin_fill() #begin filling with color
```

Adicionamos *end_fill()* após a conclusão do desenho.

Antes de começar a desenhar, pode ensinar alguns comandos básicos do Turtle aos seus alunos:

fd(20)	Move forward 20 turtle steps. Number of steps is defined inside parenthesis().
lt(90)	Change which side the turtle is facing. Turn left for 90 degrees (right angle).
rt(45)	Change which side the turtle is facing. Turn right for 45 degrees.
circle(50)	Draw a circle of radius 50.
pu()	Pen up.
pd()	Pen down.
ht()	Hide turtle icon

Se os seus alunos precisarem de uma revisão sobre ângulos, existem muitos diagramas de ângulos na Internet que podem utilizar, nomeadamente: https://www.visnos.com/demos/basic-angles.

Passo 4: Desenhar bandeiras

Desenharemos uma bandeira com 150 passos de tartaruga de largura e 90 passos de altura. Dado que os retângulos superior, médio e inferior têm o mesmo tamanho, a dimensão de cada retângulo pode ser calculada em 90/3.

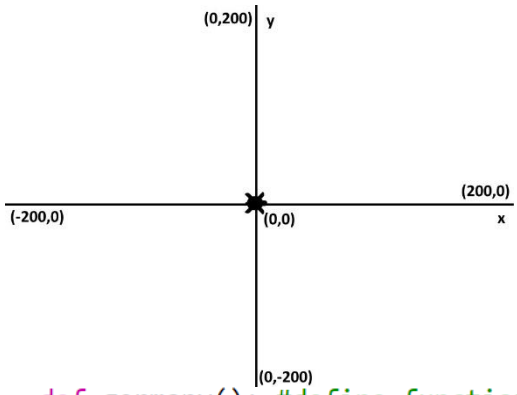
Vamos desenhar um retângulo preto.

```
def germany(): #define function which stores drawing of German flag

  bob.fillcolor("black") #fill the turtle with some color
  bob.begin_fill() #begin filling with color

  for i in range(2):
    bob.fd(150)
    bob.rt(90)
    bob.fd(30)
    bob.rt(90)
  bob.end_fill() #end filling with color
```

Para desenhar um retângulo, utilizámos o ciclo *for* para repetir o bloco de código duas vezes (o número entre parênteses define quantas vezes será repetido o bloco de códigos com indentação). A variável *i* é o contador/variável de iteração, que começa com 0 e aumenta em cada iteração do ciclo.



A tartaruga começa sempre no centro do ecrã, nas coordenadas (0,0).

Para desenhar um retângulo vermelho, utilizaremos a função *goto()* para ir para as coordenadas (0,-30). No entanto, como não queremos desenhar nada enquanto estivermos em movimento, utilizaremos *pu()* para levantar a caneta e *pd()* para a pousar:

```
def germany(): #define function which stores drawing of German flag

  bob.fillcolor("black") #fill the turtle with some color
  bob.begin_fill() #begin filling with color

  for i in range(2):
    bob.fd(150)
    bob.rt(90)
    bob.fd(30)
    bob.rt(90)
  bob.end_fill() #end filling with color

  bob.pu()
  bob.goto(0,-30)
  bob.pd()
```

Utilizaremos o mesmo código para desenhar um retângulo vermelho:

```
  bob.fillcolor("red")
  bob.begin_fill()
  for i in range(2):
    bob.fd(150)
    bob.rt(90)
    bob.fd(30)
    bob.rt(90)
  bob.end_fill()
```

Desloque-se para as coordenadas (0,-60) para desenhar o retângulo amarelo:

```
bob.pu()
bob.goto(0,-60)
bob.pd()

bob.fillcolor("yellow")
bob.begin_fill()
for i in range(2):
  bob.fd(150)
  bob.rt(90)
  bob.fd(30)
  bob.rt(90)
bob.end_fill()
```

No final, esconda a tartaruga do ecrã:

```
bob.ht()
```

Deixe os seus alunos desenharem algumas bandeiras mais simples ou, possivelmente, mais complexas. No modelo de projeto concluído, apresentado no final do documento, pode encontrar o código para desenhar quatro bandeiras: da Alemanha, da Polónia, de Itália e da Finlândia.

Parte 2: Inserir a lógica do jogo (45 minutos)

Passo 5: O ciclo *while*

Após concluídos todos os desenhos, podemos remover a linha de código para chamar a função *(germany())* e avançar para a lógica do jogo.

Vamos inserir todas as bandeiras numa lista e gerar aleatoriamente bandeiras no ecrã.

```
#put all the drawings into list
flags = [germany,poland,italy,finland]
```

Para o nosso jogo, utilizaremos o ciclo *while.* Caso os seus alunos não conheçam o ciclo *while*, poderá mostrar-lhes esta curta animação.

Todos os blocos de código incluídos no ciclo *while* serão repetidos enquanto estiverem preenchidas duas condições: o utilizador ter mais de 0 vidas e pelo menos 1 bandeira restante na lista. No início do ciclo, é necessário:

1. redefinir a posição da tartaruga sempre que for desenhada uma nova bandeira
2. definir a velocidade da tartaruga
3. gerar uma função de bandeira aleatória e chamá-la

4. pedir ao utilizador para decifrar a bandeira

```
while life>0 and len(flags)>0:
  bob.reset() #reset the position of the turtle
  bob.speed(200) #change the speed of the turtle
  flag = random.choice(flags) #generate random flag
  flag()
  answer = input("Guess the flag!")
```

Todas as linhas com indentação depois dos dois pontos pertencem ao mesmo bloco de código. Serão repetidas enquanto as nossas condições forem verdadeiras.

Passo 6: Verificar as respostas do utilizador

Se a resposta do utilizador estiver correta:

- aumentar uma unidade no número de pontos
- imprimir os pontos
- imprimir as vidas
- remover a bandeira da lista de bandeiras utilizando a palavra-chave *remove()*:

```
if answer == flag.__name__:
  print("Correct answer!")
  points += 1
  print("Points:", points)
  print("Life:", life)
  flags.remove(flag)
```

Se a resposta do utilizador não estiver correta:

- verificar se o número de pontos é superior a 1
- em caso afirmativo, reduzir 1 ponto na pontuação
- caso contrário, definir os pontos para 0 e reduzir 1 vida
- imprimir vidas e pontos.

Depois de concluído o jogo, imprimir *"GAME OVER"* no ecrã.

```
  if answer == flag.__name__:
    print("Correct answer!")
    points += 1
    print("Points:", points)
    print("Life:", life)
    flags.remove(flag)

  else:
    print("Try again")
    if points>0:
      points -= 1
    else:
      points = 0
    life -= 1
    print("Points:", points)
    print("Life:", life)

print("GAME OVER")
```

Projeto concluído: https://trinket.io/python/19b822d41f

Modelo do código: https://trinket.io/python/ce57c5cf54

Nome da autora: Ivana Vezjak