

Python Training

Sessions by Tiago Montes

Kids on Python | Guia do Instrutor

Workshop de programação Python para miúdos

Versão 1.0.1, Agosto de 2019

Boas vindas

Este é o guia do instrutor para as workshops **Kids on Python**, concebidas para introduzir miúdos à programação de computadores com Python. Divirtam-se!

Público Alvo

Miúdos com idades entre os 11 e os 13 anos, com o 5º ano de escolaridade completado, e ainda:

- Capacidade de utilização de um computador:
 - Iniciar e terminar programas.
 - Manipular janelas em tamanho e posição pondo, por exemplo, duas janelas lado a lado no ecrã.
 - Escrever texto com algum à vontade num processador de texto, incluindo a capacidade de corrigir, copiar, etc.
- Competências de matemática / geometria:
 - À vontade com aritmética entre números inteiros, positivos e negativos.
 - Conhecimento de números decimais.
 - Conhecimento de operações de comparação “maior do que” e “menor do que” com os símbolos $<$ e $>$.
 - Idealmente, mas não estritamente necessário, à vontade com coordenadas cartesianas.

Objectivos

Por ordem decrescente de importância, para os miúdos:

- Entender que os computadores podem fazer, dentro de limites, aquilo que eles quiserem.
A ideia fundamental é a de que cabe a cada um de nós decidir o que é que os nossos computadores devem fazer, e não o inverso, em que nós só fazemos aquilo que o computador nos mostra ou deixa.
- Divertirem-se o mais possível.
- Entender que há várias linguagens para programar os computadores: nós vamos utilizar o **Python**.
- Construir alguns programas com resultados visuais:
 - Desenhos e jogos de complexidade crescente.
 - Que possam entender e alterar para obter resultados diferentes.
- Entender, ao nível da programação, os seguintes conceitos:
 - Execução sequencial.
 - Expressões numéricas e aritméticas.
 - Expressões textuais elementares.
 - Variáveis.
 - Funções e parâmetros.
 - Execução condicional.
 - Execução em ciclo.

Plano de Trabalho

1 Como funcionam os Computadores.....	4
2 Utilizar o Mu.....	5
3 Fazer contas com o REPL.....	6
4 Tartaruga no REPL.....	7
5 Primeiro programa: um quadrado.....	8
6 Carregar um programa guardado.....	9
7 Dois quadrados Copy+Paste.....	10
8 Dois quadrados Usar Variáveis.....	11
9 Dois quadrados Tamanhos aleatórios.....	12
10 Dois quadrados Uma Função.....	13
11 Muitos quadrados com o teclado.....	14
12 Jogo da Corrida Apresentação.....	15
13 Jogo da Corrida Preparar a Janela.....	16
14 Jogo da Corrida Desenhar a Meta.....	17
15 Jogo da Corrida Criar os Jogadores.....	18
16 Jogo da Corrida Os Jogadores correm.....	19
17 Jogo da Corrida Corrida aleatória.....	20
18 Jogo da Corrida Passagem da Meta.....	21
19 Jogo da Corrida Determinar o vencedor.....	22
20 Jogo da Corrida Fim do Jogo.....	23
21 Jogo da Captura Apresentação.....	24
22 Jogo da Captura Preparar a Janela.....	25
23 Jogo da Captura Início da Grelha.....	26
24 Jogo da Captura Grelha completa.....	27
25 Jogo da Captura Criar o Jogador.....	28
26 Jogo da Captura O Jogador move-se.....	29
27 Jogo da Captura Criar o Fugitivo.....	30
28 Jogo da Captura O Fugitivo foge.....	31
29 Jogo da Captura Preparar a captura.....	32
30 Jogo da Captura Detectar a captura.....	33
31 Jogo da Captura O Fugitivo foge mais.....	34
32 Jogo da Captura Ganhar e Perder.....	35
33 Jogo da Captura Aperfeiçoamentos.....	36
34 Tempo Livre.....	37

Objectivos

Entender que computadores são programáveis.

Executam as instruções que nós indicamos... mas só conhecem algumas!

Há muitas linguagens para programar computadores.

Ferramentas

Miúdos sentados.

Tiago de pé.

Jogo de conversa e movimentos do Tiago pela sala.

Conversa + Jogo

Pergunta **Os computadores são espertos ou burros?**

Resposta **Nem uma nem outra: são muito obedientes e rápidos!**

Jogo Conduzir o Tiago a uma posição específica na sala.

Miúdos Dão uma de três instruções: **frente**, **esquerda**, ou **direita**.

Tiago Avança um passo, roda à esquerda/direita conforme instruções.

Reage negativamente a instruções desconhecidas.

Variantes Dar uma **instrução inválida** (exemplo: vai até à parede).

Poder dizer **quantos passos** avança na instrução **frente**.

Usar **instruções por gestos** em vez de instruções faladas.

Lição Computadores seguem instruções. Há várias línguas.

Nós vamos aprender uma chamada **Python**.

Objectivos

Saber abrir e fechar o Mu.

Saber posicionar e alterar o tamanho da janela do Mu.

Saber escrever letras maiúsculas, minúsculas e dígitos.

Saber escrever os símbolos:

.	,	()	"	:	#	?
+	-	*	/	>	<	=	_

Ferramentas

Mu

Teclado e Rato

Tiago faz + Miúdos repetem

Abrir o Mu Deslocar e alterar o tamanho da janela.

Conhecer o Mu Barra de Botões.
 Zona de texto.
 Barra inferior.

Modo do Mu Explicar que deve estar sempre no modo **Python**.
 Indicado no canto inferior direito.
 Altera-se no primeiro botão, em cima à esquerda, que diz **Mode / Modo**.

Zona de Texto Explicar separadores apresentados no topo desta zona.
 Criar um novo.

Escrever na zona de texto, em separadores diferentes:

Escrever o nome, a idade, e a data de hoje.

Escrever os símbolos:

.	,	()	"	:	#	?
+	-	*	/	>	<	=	_

Objectivo

Experimentar o REPL.

Fazer algumas contas.

Ver os primeiros erros de sintaxe.

Ferramentas

REPL.

Tiago faz + Miúdos repetem

Abrir e fechar o REPL

Forma de falar directamente com o Python.

O *prompt* do REPL

Forma do Python indicar que está à espera de instruções.

Primeiras interacções

Introduzir um número.

Números com casas decimais, Python usa ponto em vez de vírgula.

Introduzir um número inválido, como por exemplo: **12gr34**.

Introduzir letras: Python queixa-se com uma mensagem de erro.

Fazer algumas contas

Operações: soma (+), subtracção (−), multiplicação (*), divisão (/).

Podem utilizar-se parêntesis curvos para agrupar operações.

Exemplos

10 + 20

5 / 2

4 * 3

3 - 4

123456789 * 123456789

Que contas querem fazer?

Objectivos

Primeiro contacto com a Tartaruga.

Noção da importância de um programa.

Ferramentas

REPL.

Primeiras instruções do módulo **turtle**.

Tiago faz + Miúdos colaboram

Preparação	REPL	import turtle t = turtle.Turtle()
Jogo	Instruções possíveis: t.forward , t.left , e t.right . Explicar Rodar a Tartaruga da folha Desenhar com a Tartaruga .	
Jogo	Miúdos sugerem movimentos e rotação para a tartaruga.	
Colaborativo	t.reset()	
	Miúdos sugerem movimentos e rotação para desenhar um quadrado.	
Importância de um programa	Vou descansar e tenho que desligar computador. Quando abrir tenho que escrever tudo outra vez?	

Objectivos

Entender importância de um programa.

Entender execução sequencial.

Opcional: experimentar execução passo a passo com o debugger / diagnóstico do Mu.

Ferramentas

Instruções da tartaruga **import**, criação, **forward**, **left**, **right**, **color**, **width**

Tiago faz + Miúdos repetem

Tiago Novo separador para um programa.

Escreve primeiro programa que desenha quadrado.

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

Guarda o programa com **Save** / **Guardar**, como **quadrado.py**.

Verificar o programa.

Executar o programa.

Miúdos repetem

Tiago Receita para posicionar a janela da tartaruga

```
turtle.Screen().setup(startx=700, starty=100)
```

Mudar a cor e espessura do quadrado com

```
t.color(cor) e t.width(número)
```

Miúdos repetem

Tiago Executar o programa passo a passo com o debugger / diagnóstico do Mu?

Miúdos repetem

Objectivos

Aprender a carregar um programa guardado.

Ferramentas

Ações Mu **Save / Guardar e Load / Carregar.**

Tiago faz + Miúdos repetem

Tiago Fecha o Mu, abre o Mu: separador com o programa ainda lá está.
Fecha o separador, fecha o Mu: separador com o programa não está lá.
Carregar o programa **quadrado.py** com botão **Load / Carregar**.
Executar o programa.

Miúdos repetem

Objectivos

Aprender Copiar + Colar / Copy + Paste.

Fazer comentários no programa.

Ferramentas

As mesmas instruções da tartaruga e ainda **up** e **down**.

Tiago faz + Miúdos repetem

Tiago Abre programa **quadrado.py**

Desenhar segundo quadrado

Copiar as linhas do primeiro quadrado.

Colar abaixo.

Acrescentar um comentário antes de cada bloco de linhas.

Miúdos repetem

Tiago Alterar cor e espessura do segundo quadrado.

Miúdos repetem

Tiago Afastar o segundo quadrado com com instruções **up**, **forward**, **down**.

Miúdos repetem

Tiago Alterar o tamanho do segundo quadrado.

Notar que temos que mudar um número em quatro instruções diferentes.

Miúdos repetem

Objectivos

Introdução ao conceito de variáveis.

Ferramentas

Instrução de atribuição *nome = valor*

Tiago faz + Miúdos repetem

Tiago	<p>Vamos mudar o tamanho de cada um dos dois quadrados.</p> <p>Temos que mudar o tamanho de cada quadrado em quatro instruções!</p> <p>Há uma maneira melhor: vamos utilizar variáveis.</p> <p>Servem para os programas se lembrarem de coisas.</p> <p>Permitem associar nomes a coisas: números e tartarugas, por exemplo.</p> <p>Quando usadas, o Python substitui-as pela coisa que lhes está associado.</p>
Jogo	<p>Cada miúdo é uma variável.</p> <p>Tiago associa um número a cada miúdo.</p> <p>Tiago faz contas com os números de cada miúdo.</p> <p>Tiago re-associa números aos miúdos à medida que o jogo progride.</p>
REPL	<pre>a = 5 a 10 + a b = 3 b a + b</pre>
Tiago	<p>Usa uma variável chamada tamanho para cada quadrado no programa.</p> <p>Definida uma vez para cada quadrado e utilizada nas instruções forward.</p>
Miúdos repetem	

Objectivos

Introdução ao conceito de escolhas aleatórias.

Ferramentas

Instruções **random.randint** e **random.choice**, do módulo **random**.

Tiago faz + Miúdos repetem

Tiago O nosso programa faz sempre quadrados da mesma forma.

E se quiséssemos alguma variação?

Ideia E se o tamanho variasse conforme lançamos um dado?

Mote O Python pode fazer escolhas aleatórias por nós.

REPL **import random**
random.randint(1, 6)
random.randint(100, 300)

Tiago Muda o programa dos quadrados para tamanhos aleatórios.

No topo **import random**
Na variável **tamanho** **random.randint(100, 300)**

Miúdos repetem

[Questão: Interessante mas confuso? Cria dificuldade para o próximo passo? Talvez depois?]

Tiago E como fazer cores aleatórias?

REPL **import random**
random.choice(["yellow", "blue", "orange", "black"])

Tiago Muda o programa para que cada quadrado tenha uma cor aleatória.

Miúdos repetem

Objectivos

Introdução ao conceito de função como grupo de instruções.

Ferramentas

Instrução para definição de funções **def nome() :** ...

Tiago faz + Miúdos repetem

Tiago

Imaginem que queríamos desenhar muitos quadrados...

Tínhamos que escrever imensas linhas cheias de instruções.

Há uma maneira melhor, vamos ver:

Notem: as instruções para desenhar os dois quadrados são iguais.

Criamos um grupo: cada linha quatro espaços para a frente.

Na linha anterior escrevemos: **def quadrado() :**

O resultado deverá ser:

```
def quadrado() :  
    tamanho = random.randint(100, 300)  
    t.forward(tamanho)  
    t.left(90)  
    t.forward(tamanho)  
    t.left(90)  
    t.forward(tamanho)  
    t.left(90)  
    t.forward(tamanho)
```

O bloco começado por **def**, define uma função chamada **quadrado**.

- Passa a funcionar como uma nova instrução.

- Quando utilizada, executa as instruções definidas dentro da função.

Tiago

Incluir utilização da nova função **quadrado**.

Miúdos repetem

Variações

Incluir muitas vezes a utilização / invocação da função **quadrado**.

Objectivos

Introdução da receita para processamento do teclado com o módulo **turtle**.

Ferramentas

Uma das receitas apresentada no documento **Desenhar com a Tartaruga**.

Tiago faz + Miúdos repetem

Tiago Em vez do programa desenhar um número pré-definido de quadrado...
 ...vamos desenhar um quadrado de cada vez que se carregue numa tecla!

Receita:

```
turtle.listen()  
turtle.onkey(quadrado, "q")  
turtle.mainloop()
```

Explicar a receita.

Miúdos repetem.

Variações Alterar a função quadrado para:

- Cores aleatórias?
- Rotação aleatória?
- Deslocamento inicial aleatório?

Fazer com que a tartaruga ande mais depressa?

Extra Criar função **dois_quadrados...**
 ...chama duas vezes a função **quadrado**.

Fazer com que a tecla passe a desenhar dois quadrados.

Fazer com que uma tecla desenhe um quadrado e outra desenhe dois!

Objectivos

Apresentar o jogo, estimulando a sua construção.

Discutir ideias inerentes.

Ferramentas

O programa completo da corrida para mostrar.

Tiago faz + Miúdos repetem

Tiago E se vos dissesse que já sabemos quase tudo o que precisamos para fazer o nosso primeiro jogo?

Mostra o jogo

Dois corredores, que são tartarugas.

Meta desenhada com uma tartaruga.

Andam sempre para a frente, mas não desenharam linhas.

Ganha a que chegar à meta primeiro.

Objectivos

Iniciar a construção de um jogo: definir a geometria e cor de fundo da janela.

Ferramentas

Receitas simples para definir geometria e cor de fundo da janela.

Tiago faz + Miúdos repetem

Tiago Vamos começar por preparar a janela com o tamanho certo.

Novo separador.

```
import turtle  
turtle.Screen().setup(width=600, height=400)
```

Guardar como **corrida.py**.

Miúdos repetem

Tiago Agora vamos dar-lhe uma cor de fundo.

```
turtle.bgcolor("skyblue")
```

Miúdos repetem

Objectivos

Desenhar a meta na janela do jogo.

Ferramentas

Uma tartaruga dedicada para o efeito.

Tiago faz + Miúdos repetem

Tiago Vamos começar por desenhar a meta: uma linha vertical do lado direito.

Criamos uma tartaruga para o efeito.

```
t = turtle.Turtle()  
t.up()  
t.forward(200)  
t.left(90)  
t.down()  
t.forward(200)  
t.left(180)  
t.forward(400)
```

Miúdos repetem

Variações Espessura da linha da meta.

Cor da linha da meta.

Velocidade a que a linha da meta é desenhada.

Objectivos

Criar os corredores e posicioná-los.

Ferramentas

Um tartaruga para cada corredor.

Tiago faz + Miúdos repetem

Tiago

Precisamos de desenhar os corredores.

Vamos utilizar uma tartaruga para cada um:

```
cima = turtle.Turtle()  
cima.left(180)  
cima.forward(200)  
cima.right(90)  
cima.forward(100)
```

Não queremos que o corredor desenhe linhas

```
cima.up()
```

Segundo corredor:

```
baixo = turtle.Turtle()  
baixo.left(180)  
baixo.forward(200)  
baixo.left(90)  
baixo.forward(100)
```

Não estão bem orientados, pelo que nos falta:

```
cima.right(90) e baixo.left(90)
```

Duas setas? Podemos melhorar?

```
.shape("turtle") e .color(cor)
```

E outros bonecos?

```
turtle.register_shape(nome de ficheiro GIF)  
cima.shape(nome de ficheiro GIF)  
baixo.shape(nome de ficheiro GIF)
```

Objectivos

Retomar o conceito de função: define nova instrução que executa um grupo de instruções.

Ferramentas

Receita do módulo **turtle** para responder teclas **.listen**, **.onkey** e **.mainloop**.

Tiago faz + Miúdos repetem

Tiago

O que falta agora?

Os corredores têm que andar um bocadinho quando carregamos numa tecla!

Apresentar receita:

```
def corrida():  
    cima.forward(70)  
    baixo.forward(70)  
  
turtle.listen()  
turtle.onkey(corrida, "c")  
turtle.mainloop()
```

Explicar a receita.

Miúdos repetem.

Tiago

Alterar a ordem das instruções na função **corrida**.

Observar efeito – um move-se antes do outro – mas irrelevante para o jogo.

Tiago

O que é que falta para o jogo?

Ambos acabam sempre ao mesmo tempo, já viram?

Nunca param, mesmo depois de ultrapassar a meta.

Objectivos

Fazer com que jogadores corram aleatoriamente.

Ferramentas

Retomar a utilização do módulo **random** com **random.randint(mínimo, máximo)**

Tiago faz + Miúdos repetem

Tiago Como fazer com que os corredores variem em cada jogada?

Lembram-se que o Python sabe criar números aleatórios?

Vamos alterar a nossa função corrida

```
def corrida():  
    passos = random.randint(50, 100)  
    cima.forward(passos)  
    passos = random.randint(50, 100)  
    baixo.forward(passos)
```

Temos que dizer ao Python que queremos usar números aleatórios com:

```
import random
```

Miúdos repetem

Experiências Alterar a gama de números dos passos?

Um corredor com uma gama de números maior que o outro?

Variáveis para os valores mínimo e máximo dos passos?

Miúdos repetem

Tiago O que falta?

Falta ver quem ganhou...

...e fazer com que os jogadores parem assim que um deles ganhe.

Objectivos

Conceito de posição horizontal dos corredores.

Ferramentas

Coordenadas cartesianas e função **.xcor()** da tartaruga.

Instrução **print** como ferramenta auxiliar.

Tiago faz + Miúdos repetem

Tiago Como é que sabemos quem é o vencedor?
 Será aquele que ultrapassou a meta primeiro.
 Mas como sabemos se uma ou outra tartaruga ultrapassaram a meta?
 Temos que utilizar instruções **Python** para saber.

Interlúdio para **Introdução às Coordenadas Cartesianas**.

Tiago As tartarugas têm a instrução **.xcor()**, que representa a sua posição X.
 Mas como é que a podemos utilizar?

Interlúdio de Introdução da Instrução **print**.

Introduzir **print** para mostrar o número de passos de cada tartaruga.

Introduzir **print** para mostrar o **.xcor()** de cada tartaruga.

Miúdos repetem.

Objectivos

Introdução ao conceito de execução condicional.

Ferramentas

Expressões de comparação numérica.

Execução condicional com o **if**.

Tiago faz + Miúdos repetem

Tiago Como é que sabemos quem é o vencedor?
 Será aquele que ultrapassou a meta primeiro.
 Temos que ver o valor de **.xcor()** para as duas tartarugas.

Se a de cima ultrapassou a meta, vence a de cima.
Se a de baixo ultrapassou a meta, vence a de baixo.
Se ambas ultrapassaram a meta, é um empate.

Tiago Execução condicional

```
if cima.xcor() > 200:  
    print("Ganhou a de cima!")  
if baixo.xcor() > 200:  
    print("Ganhou a de baixo!")
```

Miúdos repetem

Tiago Acrescentar verificação para o caso de ambas as tartarugas passarem a meta.
 Qual tem que vir primeiro?
 Execução condicional com expressão conjuntiva **and**.

Miúdos repetem

Objectivos

Conceito de interromper a execução de uma função.

Melhorar a indicação do vencedor.

Ferramentas

Instrução **return** para interromper a execução de uma função.

Tiago faz + Miúdos repetem

Tiago Quando um ou ambos os corredores ultrapassam a meta o jogo termina.
 Se olharmos para a nossa função corrida...
 ...a partir de certa altura não queremos avançar os jogadores.
 Qual é essa condição?

Miúdos

Tiago Quando uma tartaruga ou a outra ultrapassaram a meta.
 Execução condicional com expressão disjuntiva **or**.
 No topo da função:
 if cima.xcor() > 200 or baixo.xcor() > 200:
 return

Miúdos repetem

*[Questão: zero referências a **.circle** e **.write** no na folha **Desenhar com a Tartaruga**]*

Tiago Vamos encontrar uma forma melhor de indicar a vitória.
 Hipóteses, como receita:
 t.circle(20)
 t.write(..., align="right", font=("Arial", 50, "bold"))

Miúdos repetem

Objectivos

Apresentar o jogo, estimulando a sua construção.

Discutir ideias inerentes.

Ferramentas

O programa completo da captura para mostrar.

Acções

Tiago

Mostra o jogo:

Grelha onde os jogadores se movem, desenhada com uma tartaruga.

Cada jogador é uma tartaruga, mas não desenha linhas quando se move.

Um jogador é controlado por nós, com o teclado.

O outro é controlado pelo nosso programa, e foge aleatoriamente.

Objectivos

Iniciar a construção de um jogo: definir a geometria e cor de fundo da janela.

Ferramentas

Receitas simples para definir geometria e cor de fundo da janela.

Acções

Tiago Vamos começar por preparar a janela com o tamanho certo.

Novo separador.

```
import turtle  
turtle.Screen().setup(width=700, height=700)
```

Guardar como **captura.py**.

Miúdos repetem.

Tiago Agora vamos dar-lhe uma cor de fundo.

```
turtle.bgcolor("orange")
```

Miúdos repetem.

Objectivos

Começar a desenhar a grelha de fundo: linhas horizontais.

Ferramentas

Coordenadas cartesianas e folha **Desenhar com a Tartaruga** para apoio.

Acções

Tiago Como desenhar a grelha?
São imensos quadrados, mas também são várias linhas horizontais e verticais.

Interlúdio **Introdução às Coordenadas Cartesianas.**

Tiago Vamos desenhar a grelha com linhas. A primeira vai ser horizontal:

```
t = turtle.Turtle()
t.up()
t.goto(-300, 300)
t.down()
t.goto(300, 300)
```

Miúdos repetem

Tiago Vamos desenhar a segunda linha horizontal:

```
t.up()
t.goto(-300, 200)
t.down()
t.goto(300, 200)
```

Miúdos repetem

Tiago Que trabalhadeira! Vamos fazer melhor, com uma função.

```
def linha_horizontal(y):
    t.up()
    t.goto(-300, y)
    t.down()
    t.goto(300, y)
```

Miúdos repetem

Objectivos

Completar o desenho da grelha de fundo: linhas verticais.

Ferramentas

Folha **Desenhar com a Tartaruga** para apoio nas coordenadas.

Accões

Tiago

Faltam-nos as linhas verticais, certo?

Vamos criar e usar uma função para isso:

```
def linha_vertical(x):  
    t.up()  
    t.goto(x, -300)  
    t.down()  
    t.goto(x, 300)
```

```
linha_vertical(-300)  
linha_vertical(-200)  
linha_vertical(-100)  
linha_vertical(0)  
linha_vertical(100)  
linha_vertical(200)  
linha_vertical(300)
```

Miúdos repetem

Variantes

Alterar a cor da linha da grelha: **t.color("dark orange")**

Alterar a espessura da linha da grelha.

Alterar a velocidade da tartaruga que desenha a grelha.

Objectivos

Criar o jogador, apresentando-o.

Ferramentas

Uma tartaruga e a receita para mudar a sua imagem.

Acções

Tiago

Vamos agora criar o nosso jogador.

Vai ser uma tartaruga que não desenha linhas quando se move.

```
jogador = turtle.Turtle()  
jogador.up()
```

Miúdos repetem

Tiago

Como queremos um boneco mais giro aplicamos a receita:

```
turtle.register_shape("boneco-c-cavaleiro.gif")  
jogador.shape("boneco-c-cavaleiro.gif")
```

Lembrem-se que estamos a utilizar o nome de um ficheiro de imagem!

Miúdos repetem

Objectivos

Deslocar a tartaruga do jogador com o teclado.

Ferramentas

Receita do módulo **turtle** para responder teclas **.listen**, **.onkey** e **.mainloop**.

Acções

Tiago Vamos fazer com que o jogador se mova conforme pressionamos teclas!

Lembra-se da receita que utilizámos com os quadrados?

```
def para_cima():  
    print("para cima!")  
  
turtle.listen()  
turtle.onkey(para_cima, "Up")  
turtle.mainloop()
```

Miúdos repetem

Tiago A tecla está ser detectada, mas o jogador não se move.

Como fazer? Discussão.

```
def para_cima():  
    x = jogador.xcor()  
    y = jogador.ycor() + 100  
    jogador.goto(x, y)
```

Miúdos repetem

Tiago Agora vamos fazer o movimento para baixo:

Criar a função **para_baixo**.

Acrescentar uma linha **turtle.onkey(para_baixo, "Down")**.

Miúdos repetem

Tiago Agora façam vocês o código para os movimentos esquerda e direita.

Miúdos fazem

Objectivos

Criar o fugitivo, apresentando-o

Ferramentas

Uma tartaruga e a receita para mudar a sua imagem.

Acções

Tiago

Vamos criar o nosso fugitivo.

Outra tartaruga que não desenha linhas quando se move.

```
fugitivo = turtle.Turtle()  
fugitivo.up()
```

Voltamos a aplicar a receita para mudar o seu aspecto.

```
turtle.register_shape("boneco-c-animal.gif")  
fugitivo.shape("boneco-c-animal.gif")
```

Miúdos repetem

Tiago

Qual dos bonecos aparece à frente?

A última tartaruga que se cria, fica à frente.

Se quisermos o jogador à frente, temos que trocar ordem de criação.

Miúdos experimentam

Tiago

O que nos falta?

O fugitivo tem que fugir, certo?

Objectivos

Posicionar o fugitivo aleatoriamente no início do jogo.

Ferramentas

Usar a função **random.randint** para posicionar o fugitivo.

Acções

Tiago Para onde é que o fugitivo pode fugir?

Vamos experimentar:

```
def fugitivo_foge():  
    x = 50  
    y = 250  
    fugitivo.goto(x, y)
```

E utilizamos essa função antes de ficar à espera do teclado.

```
fugitivo_foge()
```

Demonstra que estas coordenadas para o fugitivo não servem para o jogo...

...têm que ser múltiplos de 100.

Miúdos repetem

Tiago Estudo das combinações de coordenadas válidas.

Os valores do X e do Y têm que ser um de: -300, -200, -100, 0, 100, 200 ou 300!

Como fazer?

```
def fugitivo_foge():  
    x = random.randint(-3, 3) * 100  
    y = random.randint(-3, 3) * 100  
    fugitivo.goto(x, y)
```

Com **import random** no topo do programa.

Miúdos repetem

Objectivos

Entender a forma de detectar que o jogador captura o fugitivo.

Ferramentas

Coordenadas do jogador e fugitivo com instruções `.xcor()` e `.ycor()` das tartarugas.

Instrução **`print`** para ensaios e diagnóstico.

Acções

Tiago

Como detectar se o jogador capturou o fugitivo?

Discussão

Resposta Quando as suas coordenadas X e Y forem iguais.

Tiago

Em que ponto do programa temos que o verificar?

Discussão

Resposta Depois de cada movimento do jogador.

Tiago

Vamos criar uma função que será executada depois de cada movimento.

Ingredientes para estudo:

Instruções `.xcor()` e `.ycor()` que representam X e Y da tartaruga.

A instrução **`print`** para nos ajudar a entender o problema.

`def tenta_capturar():`

`print("X do Jogador", jogador.xcor())`

`print("Y do Jogador", jogador.ycor())`

`print("X do Fugitivo", fugitivo.xcor())`

`print("Y do Fugitivo", fugitivo.ycor())`

Acrescentar chamada a **`tenta_capturar`** depois de cada movimento...

...nas funções **`para_cima`**, **`_baixo`**, **`_esquerda`**, e **`_direita`**.

Miúdos repetem

Objectivos

Detectar que o jogador captura o fugitivo..

Ferramentas

Execução condicional e comparação das coordenadas do jogador e fugitivo.

Acções

Tiago Vamos escrever uma mensagem quando o jogador captura o fugitivo.

Como escrever uma mensagem apenas em algumas condições?

Introdução ao conceito de execução condicional e instrução **if**.

Alteramos a função

```
def tenta_capturar():  
    ...  
    if jogador.xcor() == fugitivo.xcor() and ...:  
        print("Capturado!")
```

Miúdos repetem

Tiago Vamos encontrar uma forma mais interessante de indicar a captura.

Hipóteses, como receita:

```
fugitivo.circle(20)  
.color("white")  
.write(..., align="center", font=("Arial", 50, "bold"))
```

Miúdos exploram

Tiago O jogo é muito fácil, porque o fugitivo só foge no início.

O que podemos fazer?

Objectivos

O fugitivo foge à medida que o jogador se move.

Ferramentas

Execução condicional e números aleatórios.

Acções

Tiago Vamos fazer com que, às vezes, o fugitivo mude de posição.

Imaginem, por exemplo, que queremos lançar um dado e, se sair 1...
...fazemos com que o fugitivo se mova ao calhas, como logo no início.

Como poderemos fazer?

- ...esperando ideias de números aleatórios.
- ...esperando ideias de execução condicional.

Miúdos

Tiago Depois de cada movimento:

- ...ou quatro vezes nas funções de movimento.
- ...ou uma vez na função `tenta_capturar`.

```
def tenta_capturar():  
    ...  
    if random.randint(1, 6) == 1:  
        fugitivo_foge()
```

Miúdos repetem

Tiago Notam que às vezes o fugitivo foge duas vezes quando é capturado?

Vamos diagnosticar com **print**.

Instrução **return** termina a execução de uma função.

Miúdos repetem.

Objectivos

Melhorar o jogo por forma a que se possa perder, ou ganhar.

Ferramentas

Uma nova tartaruga, para representar energia que desce com movimentos e sobe com capturas.

Acções

Tiago Vamos tornar o jogo mais interessante: para podermos perder ou ganhar.

Ideia:

Jogador tem energia que se gasta com movimento e sobe com capturas.

Energia é uma tartaruga “bola” que se desloca horizontalmente.

Se chegar à esquerda perdemos, se chegar à direita ganhamos.

Nova tartaruga da energia:

```
energia = turtle.Turtle()  
energia.up()  
energia.shape("circle")  
energia.color("gold")  
energia.goto(0, 330)
```

Miúdos repetem

Tiago Consumir energia sempre que há movimento:

```
energia.forward(-20)
```

Ganhar energia quando há uma captura:

```
energia.forward(100)
```

Miúdos repetem

Tiago Combinar tudo numa função com um argumento: **actualiza_energia**.

...verificar vitória ou derrota com **energia.xcor() < -300** ou **> 300**.

Miúdos repetem.

Objectivos

Aperfeiçoar o código, facilitando o controlo da dificuldade do jogo.

Ferramentas

Substituir constantes por variáveis.

Acções

Tiago

Como tornar o jogo mais fácil ou mais difícil?

Alterar energia ganha com capturas.

Alterar energia perdida com movimentos.

Alterar a possibilidade de fuga do fugitivo.

Para o fazer temos que alterar coisas espalhadas pelo nosso programa.

Vamos concentrá-las todas em variáveis, definidas no início do programa.

ENERGIA_MOVIMENTO = 20

ENERGIA_CAPTURA = 100

POSSIBILIDADE_DE_FUGA = 2

Miúdos repetem

Tiago

Nota: Também podíamos fazê-lo com a grelha, se quiséssemos!

Importante: já viram que o jogo não pára e que o jogador continua a andar?

Receita para dizer ao Python e ao módulo da tartaruga para esquecer o teclado:

`turtle.onkey(None, "Up")`

Experimentar utilizar para verificar o término do jogo.

Miúdos repetem

Tiago

Podemos simplificar as funções de movimento...

...passando os argumentos **x** e **y** à função **tenta_capturar**.

Miúdos repetem

Objectivos

Brincar livremente com o que aprenderam.

Consolidando conceitos ao aplicar ideias que tenham.

Ferramentas

Todas as que conhecem e talvez uma ou outra que possam ser introduzidas.

Possibilidade de receita para ciclos:

```
for vez in range(quantidade):  
    # instruções
```

Possibilidade de introdução da função **input** e brincadeiras com **input** e **print**.

Acções

Tiago Parabéns completámos o nosso curso!

 Querem experimentar alterar o vosso jogo?

 Querem experimentar mais coisas?

Actividade Explorar o **Guia de Experiências** e, em particular:

- Estrelas
- Arte abstracta
- Desenhar com o rato
- Cumprimentos
- Pedra, Papel, ou Tesoura?
- Fazer contas
- Adivinhar um número
- Ciclos de contagem
- Desenhar uma grelha

Conclusão

Como complemento a esta série de actividades, feitas em ambiente presencial, de convívio, exposição, discussão, e trabalho individual e colectivo, pode ser distribuído pelos miúdos o **Guia de Experiências**, contendo um conjunto de programas que cada miúdo, a seu tempo, ou durante as workshops, poderá explorar por si.

Obrigado!