**Dicas Rápidas PsychoPy**

(Última atualização em 07/08/2021)

*Caso detecte erros nas dicas, você poderá me avisar por e-mail: lima.piraju@gmail.com*

**Conteúdo**

[Como dar um *feedback* corretivo após cada tentativa? 2](#_Toc79241823)

[Como encerrar um *loop*? 2](#_Toc79241824)

[Como embaralhar uma lista? 3](#_Toc79241825)

[Como executar uma ação com base no objeto que recebeu um clique do *mouse*? 3](#_Toc79241826)

[Como salvar manualmente uma variável no arquivo de saídas gerado pelo PsychoPy? 4](#_Toc79241827)

[Como atribuir entradas do teclado a uma variável? 5](#_Toc79241828)

[Como tornar o cursor do *mouse* invisível ou visível durante as tarefas? 5](#_Toc79241829)

Como dar um *feedback* corretivo após cada tentativa?

Suponha que o componente **resp** controle as respostas do participante no teclado. Uma vez que a opção *store correct* (armazenar acerto ou erro) estiver habilitada, o *feedback* pode ser dado por meio do atributo **corr** da variável **resp**. Esse atributo tem valor 1, caso a resposta estiver correta; e tem valor 0, caso a resposta estiver incorreta. Se o participante quiser controlar tanto o texto, quanto a cor do texto de *feedback*, o seguinte bloco de códigos pode ser usado para atingir esse fim:

|  |
| --- |
| **if resp.corr:**  **msg = “Correto!”**  **msg\_color = “green”**  **else:**  **msg = “Incorreto!”**  **msg\_color = “red”** |

Suponha que o componente de texto **feedback\_msg** tenha sido criado. Agora, basta estabelecer sua cor e sua mensagem como **$msg\_color** e **$msg**, respectivamente (marcando a opção *set every repeat*, se o *feedback* estiver contido em um *loop* com várias tentativas). A mesma lógica se aplica quando a “mensagem” é definida como uma imagem, ao invés de como um texto.

Para detalhes sobre a implementação de *feedback* corretivo após cada tentativa, ver Aulas [005](https://www.youtube.com/watch?v=Kxcur55tQ7c), [006](https://www.youtube.com/watch?v=0SWTs_rcKNA) e [024](https://www.youtube.com/watch?v=E-JA0KAPu_M).

Como encerrar um *loop*?

Considere um *loop* cujo nome é **nome\_do\_loop**. É possível forçar o encerramento de **nome\_do\_loop** por meio do seguinte comando:

|  |
| --- |
| **if condicao:**  **nome\_do\_loop.finished = True** |

onde **condicao** se refere ao critério que deve ser verdadeiro para que o atributo **finished** do *loop* se torne verdadeiro. A Aula [013](https://www.youtube.com/watch?v=CraOm0b_KUY) ilustra um procedimento de encerramento de *loop* por critério de tempo.

Como embaralhar uma lista?

Como o PsychoPy já importa automaticamente as bibliotecas por padrão, realizar essa ação é bem simples. Suponha que temos a lista **x**. Para embaralhar a lista, basta colocar em um componente de códigos o comando **shuffle(positions)**. A escolha de qual aba inserir o código dependerá da frequência com que a lista deverá ser embaralhada (e.g., se for a cada rotina, deve-se inserir o código na aba *Begin Routine*).

Como executar uma ação com base no objeto que recebeu um clique do *mouse*?

Suponha que o componente de mouse se chame **resp** e que na tela tenham duas imagens, denominadas **alvo** e **distrator**. Esta ação pode ser executada usando o atributo de componentes de *mouse* chamado **isPressedIn()**. Isso é feito da seguinte maneira:

|  |
| --- |
| **if resp.isPressedIn(alvo):**  **acerto = True**  **else:**  **acerto = False** |

Podemos, a título de demonstração, salvar a variável **acerto** no arquivo de saídas gerado pelo PsychoPy. Isso é demonstrado na dica a seguir.

Como salvar manualmente uma variável no arquivo de saídas gerado pelo PsychoPy?

Suponha que armazenamos um valor *booleano* (**True** ou **False**) na variável **acerto**, conforme demonstrado na dica anterior. Podemos salvar essa variável *booleana*, convertida para inteiro, por meio do comando **thisExp.addData()**. Esse comando recebe dois argumentos. O primeiro argumento é o nome que será dado à coluna que conterá os valores da variável no arquivo de saídas. Esse valor deve ser colocado entre aspas. O segundo argumento é o nome da variável contida no programa atual que deverá ser salva no Excel. Com base nessa descrição e na variável **acerto**, poderíamos colocar o seguinte código em um componente de códigos, na aba *End Routine*:

|  |
| --- |
| **thisExp.addData(“resposta\_do\_participante”, int(acerto))** |

O comando **int** converte o valor contido em **acerto** para um número inteiro: **True** é convertido em **1**; **False** é convertido em **0**. Na dica anterior, poderíamos ter diretamente atribuído os valores **0** ou **1** a **acerto**, o que permitiria abrirmos mão do uso do comando **int** nesta dica. Se isso tivesse sido feito, a dica anterior teria o seguinte texto:

|  |
| --- |
| **if resp.isPressedIn(alvo):**  **acerto = 1**  **else:**  **acerto = 0** |

E a variável **acerto** seria salva de forma simplificada, da seguinte maneira:

|  |
| --- |
| **thisExp.addData(“resposta\_do\_participante”, acerto)** |

O comando comando **thisExp.addData()** também pode receber uma *string* como segundo argumento. Veja:

|  |
| --- |
| **if resp.isPressedIn(alvo):**  **thisExp.addData(“resposta\_do\_participante”, “o participante acertou”)**  **else:**  **thisExp.addData(“resposta\_do\_participante”, “o participante se lascou”)** |

Como atribuir entradas do teclado a uma variável?

Isso pode ser feito por meio do comando **event.getKeys()**. Basta atribuir o conteúdo digitado pelo participante uma variável. Suponha que usemos a variável **vai\_corinthians** para esse fim. Um bloco de código simples atribuir a essa variável a entrada no teclado.

|  |
| --- |
| **vai\_corinthians = event.getKeys()** |

Por exemplo, se o participante pressionar a tecla “x”, a variável **vai\_corinthians** armazenará o valor “x”. Se o participante digitar a tecla “Caps Lock”, a variável **vai\_corinthians** armazenará o valor “capslock”. Se o participante digitar a tecla “9” à direita do teclado, a variável **vai\_corinthians** armazenará o valor “num\_9”. E assim por diante.

Como tornar o cursor do *mouse* invisível ou visível durante as tarefas?

O cursor do *mouse* pode ser invisibilizado em todo o programa, em rotinas específicas ou em momentos específicos de uma rotina. A depender do cenário desejado, o código deverá ser digitado em abas distintas do componente de códigos. Suponha que o pesquisador queira invisibilizar o cursor na rotina **study\_trial**. Para isso, um componente de códigos nessa rotina será o suficiente. Na aba *Begin Routine*, o pesquisador deverá inserir o seguinte código:

|  |
| --- |
| **win.mouseVisible = False** |

Suponha que o pesquisador queira fazer com que o cursor do *mouse* volte a ficar visível na rotina **test\_trial**, que ocorre logo após o fim de **study\_trial**. Para isso, um novo componente de códigos deverá ser criado, agora dentro da rotina **test\_trial**. Na aba *Begin Routine*, o pesquisador deverá inserir o seguinte código:

**win.mouseVisible = True**