PsychoPy Builderにおける実験の動的な制御の方法

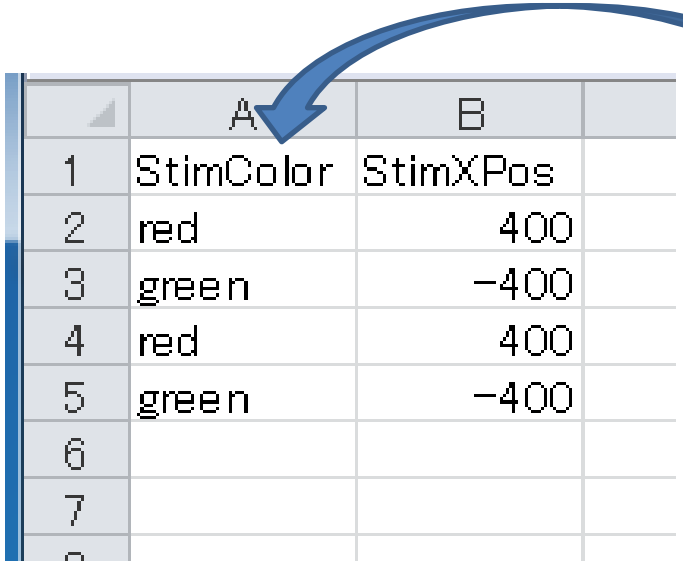
十河宏行(愛媛大学法文学部)

今日の内容

- Builderの内部変数を利用する
- ExpInfoダイアログの値を利用する
- オブジェクトのデータ属性を利用する
- if文を用いて処理を分岐する
- ルーチンを中断する
- オブジェクトのメソッドを利用する(時間があれば)

Builderの内部変数を利用する

- 変数
  - さまざまな値を代入することが出来る「箱」のようなもの。
  - Builderの条件ファイルにおけるパラメータ名は、Builder内部では変数として扱われる。



	A	B
1	StimColor	StimXPos
2	red	400
3	green	-400
4	red	400
5	green	-400
6		
7		

StimColorという名前の「変数」にred、greenといった「値」を次々と代入しながら処理を繰り返していると考える。

- Builderでは、Builderを動作させるために内部で自動的に用意される変数がある(**内部変数**)。

t	ルーチンが開始してから経過した時間
frameN	ルーチンが開始してから描画されたフレーム数
theseKeys	その瞬間に押されたキー名

ほかにもいろいろあります

- **[作業1 (sample01.psyexp)]**
  - Textコンポーネントを1個配置し、**Name**を**text_t**とする。
  - **text_t**の**Text**に **$t** と入力し、set every repeatにする。
  - **text_t**の**Stop**を空欄にする。

  - 注
    - **赤文字**はコンポーネントのプロパティを表す。
    - **Units**はすべて**norm**とする。
    - Staticコンポーネントはすべて削除するものとする。
  - ポイント
    - **プロパティ名に$が含まれないプロパティにおいて変数を参照したりする時には、入力する値に$をつける。**
    - **表示する値を常に更新し続けるにはset every repeatを選択する。**

sample01.psyexpの実行結果

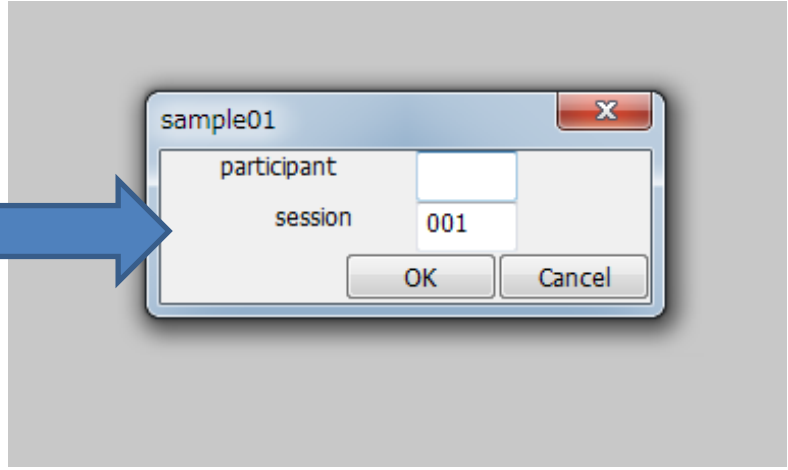
画面中央に内部変数tに格納されている値、すなわちルーチンが始まってからの時間が表示される。

ESCキーを押すと終了する。

ExpInfoダイアログの値を利用する

- ExpInfoダイアログとは、PsychoPyの実験を実行する時に最初に表示されるダイアログのこと。

このダイアログ



- ExpInfoダイアログに入力された値はexpInfoという変数にPythonの**dictオブジェクト**として格納されている。
- dictオブジェクトからは、[ ]演算子を用いて値を取り出すことが出来る。

expInfo['participant'] → ダイアログのparticipantに入力された値

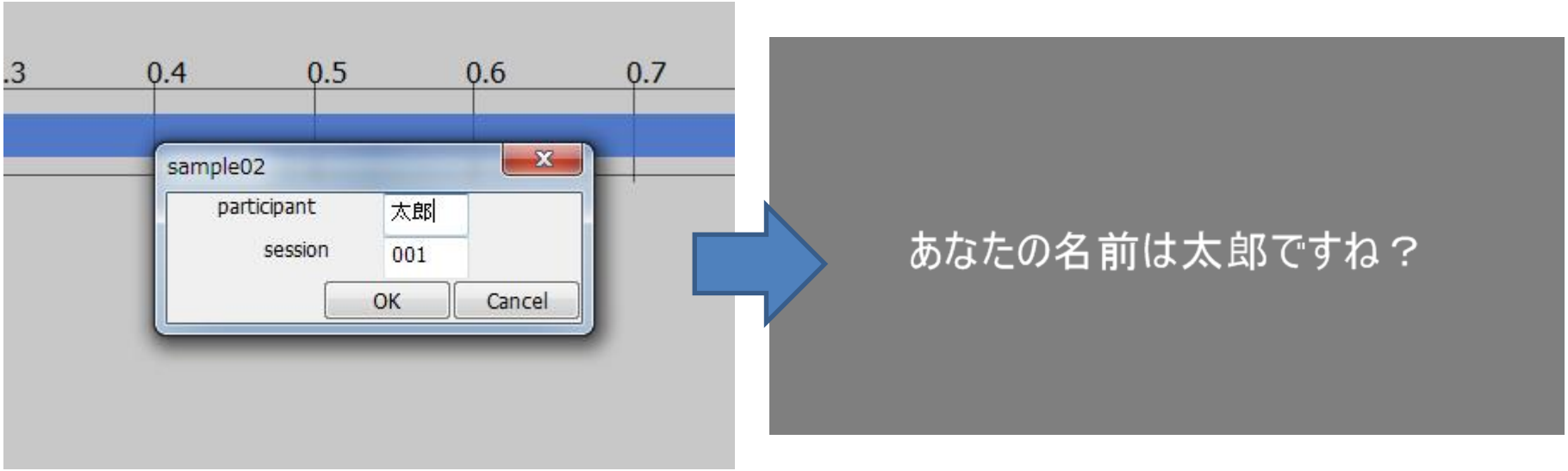
expInfo['my item'] → ダイアログのmy itemに入力された値

- **[作業2 (sample02.psyexp)]**
  - Textコンポーネントを1個配置し、**Name**を**text_name**とする。
  - **text_name**の**Stop**を空欄にする。
  - **text_name**の**Text**に 以下の式を入力する。

$u'あなたは' + expInfo['participant'] + u'ですね？'

  - ポイント
    - $が付くとPythonの式として解釈されるので、通常**Text**に入力する場合と異なり、文字列はシングルクォーテーションまたはダブルクォーテーションで囲む必要がある。
    - 日本語の文字列にはクォーテーションマークの前にuが必要(Unicode)。
    - 複数の文字列を + で連結することが出来る。

sample02.psyexpの実行結果



participantに入力した値をTextコンポーネントで表示できる。

- expInfoダイアログを用いると、試行数や刺激の大きさなどのパラメータを実行時に入力して指定することが出来る。
- expInfoダイアログを用いて刺激の大きさや位置などの数値を入力したい場合は、入力値(文字列)を数値に変換する必要がある。

```
int(  expInfo['stimulus size']  )
```

整数に変換

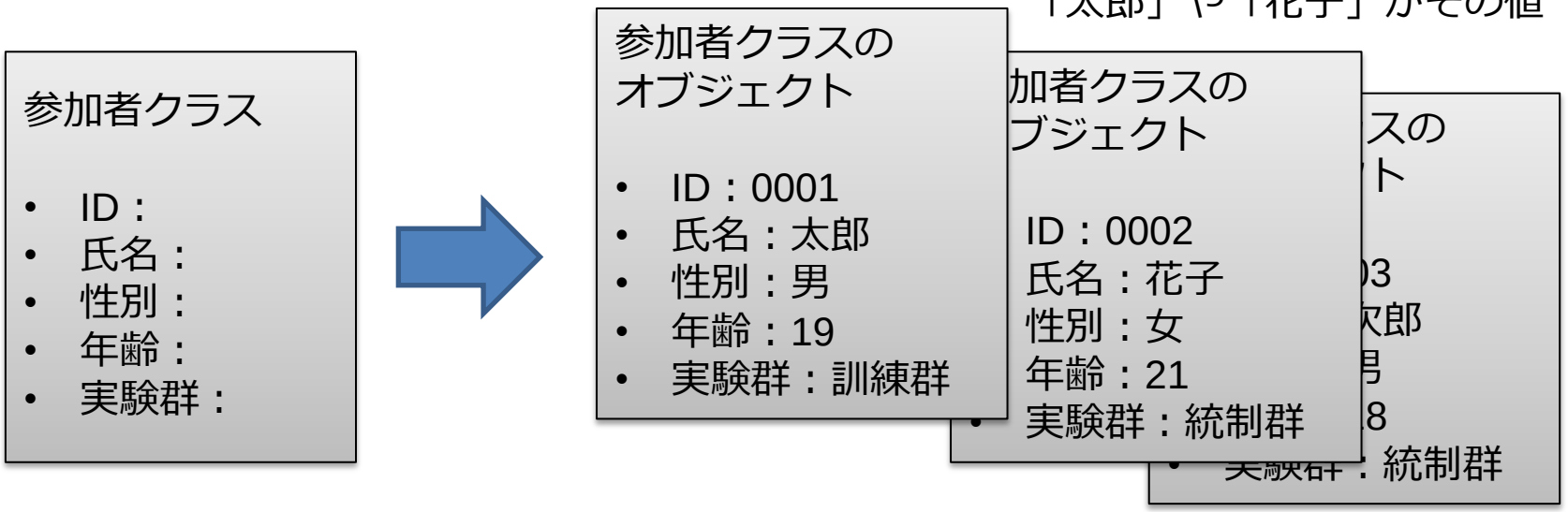
```
float(  expInfo['stimulus size']  )
```

小数に変換

オブジェクトのデータ属性を利用する

- オブジェクト
  - 構造化されたデータを扱うための仕組み
  - データの構造と操作を定義したものをクラスという
    (かなり乱暴な要約)
- データ属性
  - オブジェクトに属するデータ

「氏名」などがデータ属性で
「太郎」や「花子」がその値



- Builderでルーチンに配置しているTextやKeyboardといったコンポーネント、フローに配置しているループなどはすべてオブジェクトである。
- コンポーネントやルーチン、ループの名前は、そのままPsychoPyの内部で用いられる変数名である。
- Pythonの文法では、オブジェクトのデータ属性を参照する時には . 演算子(ドット演算子)を使う。

key_resp_2.keys 

Key_resp_2という変数に格納されたオブジェクトのkeysという属性の値

- **[作業3 (sample03.psyexp)]**
  - Textコンポーネントを1個配置し、**Name**を**text_key**とする。
  - Keyboardコンポーネントを1個配置し、**Name**を**kb**とする。
  - **text_key**と**kb**の**Stop**を空欄にする。
  - **kb**の**Allowed Keys $**を空欄にし、**Force end of Routine**のチェックを外す。
  - **text_name**の**Text**に**$kb.keys** と入力し、set every frameにする。

  - 念のため、実験設定ダイアログを開いて**Full-screen window**のチェックを外しておく。

sample03.psyexpの実行結果

キーボードのキーを押すと、そのキーに対応した「キー名」が表示される。よく使うキーのキー名を覚えておくと便利。

ESCキーを押して終了出来ない場合は、Builderの画面に戻ってStopボタンを押す(Full-screen windowでない場合)。

- **[作業4 (sample04.psyexp)]**
  - Textコンポーネントを1個配置し、**Name**を**text_trial**とする。
  - Keyboardコンポーネントを1個配置し、**Name**を**kb**とする。
  - **text_trial**と**kb**の**Stop**を空欄にする。
  - trialルーチンを繰り返すようにフローに**trials**というループを追加する。
  - **text_trial**の**Text**に以下の式を入力し、set every repeatにする。

$u'第' + str(trials.thisN+1) + u'試行'

  - ポイント:
    - trialsのデータ属性thisNには実行済みのループ回数が格納されている。
    - 数値を文字列として表示する時には、str( )で文字列に変換する。

sample04.psyexpの実行結果

スクリーンに「第n試行」(n=1,2,3…)と表示される。スペースキーなどを押すと次の試行へ進む。

if文を用いて処理を分岐する

- if文
  - Pythonにおけるプログラムの流れを制御する文のひとつ。

```
if 条件式:
    条件式が真の場合に実行する処理
else:
    条件式が偽の場合に実行する処理
```

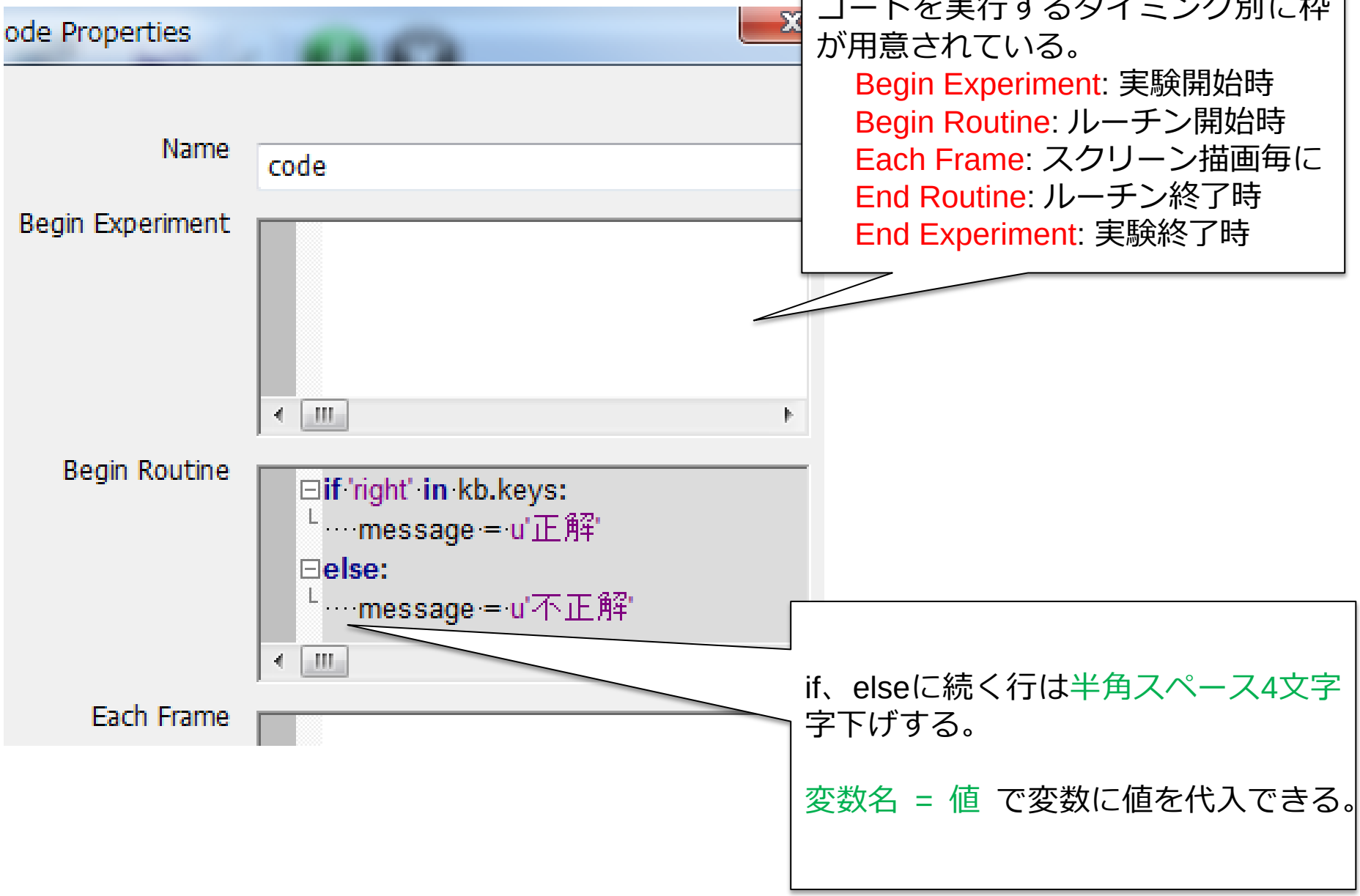
- Codeコンポーネント
  - Builderの実験に直接Pythonのコードを埋め込むコンポーネント。
  - Codeコンポーネントを用いてif文を埋め込むと条件に応じて刺激などを変化させることが出来る。

- 条件式
  - 評価した結果がTrue(真)またはFalse(偽)になる式。

A == B	AとBは等しい	not A	Aの否定
A!=B	AとBは等しくない	A and B	AかつB
A > B	AはBより大きい	A or B	AまたはB
A < B	AはB未満	A in B	AはBに含まれる
A >=B	AはB以上	など	
A <= B	AはB以下		

(rt < 1.0) and (resp == 'right') 

rtが1.0未満でrespが'right'に等しい


ode Properties
Name
code
Begin Experiment
Begin Routine
if 'right' in kb.keys:
message = u'正解'
else:
message = u'不正解'
Each Frame
コードを実行するタイミング別に枠が用意されている。
Begin Experiment: 実験開始時
Begin Routine: ルーチン開始時
Each Frame: スクリーン描画毎に
End Routine: ルーチン終了時
End Experiment: 実験終了時
if、elseに続く行は半角スペース4文字字下げする。
変数名 = 値 で変数に値を代入できる。

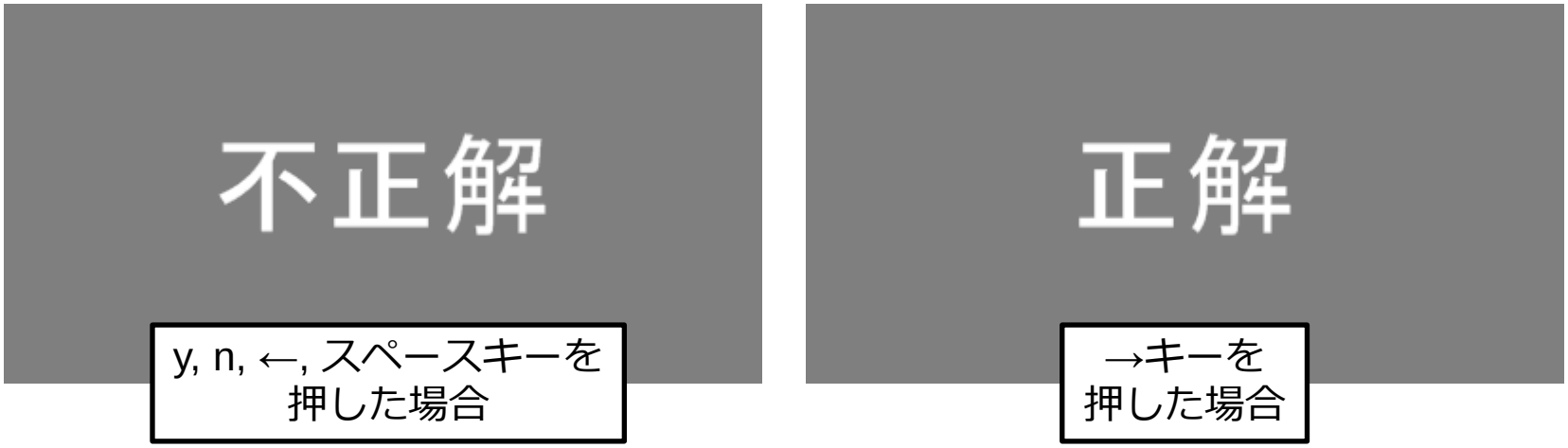
- **[作業5 (sample05.psyexp)]**
  - フローに**feedback**というルーチンを追加し、**trial**ルーチンの後に実行されるようにする。
  - trialルーチンで以下の作業をする。
    - Keyboardコンポーネントを1個配置し、**Name**を**kb**とする。
    - **kb**の**Stop**を空欄にする。
  - feedbackルーチンで以下の作業をする。
    - Codeコンポーネントを1個配置し、**Begin Routine**に以下の式を入力する。

```
if 'right' in kb.keys:
    message = u'正解'
else:
    message = u'不正解'
```

Textコンポーネントを
Codeコンポーネントより
下に配置する！

    - Textコンポーネントを1個配置し、**Name**を**text_feedback**とする。
    - **text_trial**の**Text**に **$message** と入力し、set every repeatにする。
  - ポイント
    - trialルーチンに配置してあるkbを、feedbackルーチンに配置されたCodeコンポーネントから利用できる。コンポーネントのNameは実験全体で有効。

sample05.psyexpの実行結果



灰色一色のスクリーンが表示されたら、y、n、←、→、スペースのいずれかを押す。→を押したときだけ正解と表示される。

条件ファイルで正答キーをcorrectAnsという名前で登録し、
if correctAns in kb.keysとすれば試行毎に正答キーを変更できる。

ルーチンを中断する

- より柔軟なルーチンの中断
  - Keyboardコンポーネントでは、**Allowed Keys $**に指定されたキーのどれを押しても中断されてしまう。
  - あるキーが押された場合は記録しながらルーチンを継続し、別のキーが押された場合はルーチンを中断したい場合がある。

  - Builderは、ルーチンの実行中にcontinueRoutineという内部変数にFalse(偽)という値が代入されるとルーチンを中断する機能を持っている。

- **[作業6 (sample06.psyexp)]**
  - Keyboardコンポーネントを1個配置し、**Name**を**kb**とする。
  - Polygonコンポーネントを1個配置し、**Name**を**poly**とする。
  - **kb**と**poly**の**Stop**を空欄にする。
  - **kb**の**Force end of Routine**のチェックを外す。
  - **poly**の**Position [x, y] $**に**[xpos, 0]**と入力し、set every frameにする。
  - Codeコンポーネントを1個配置し、**Begin Experiment**に以下の式を入力する。

xpos = 0

次ページに続く

- **[作業6 (sample06.psyexp)] つづき**
  - Codeコンポーネントの**Each Frame**に以下の式を入力する。

```
if 'right' in theseKeys:
    xpos += 0.05
elif 'left' in theseKeys:
    xpos -= 0.05
elif 'space' in theseKeys and t>10:
    continueRoutine = False
```

  - ポイント
    - +=, -= はそれぞれ変数に値を加える、値を引く演算子。
    - elif はelse ifのことで、複数の条件を次々と判定して処理を分岐させる時に用いる。

sample06.psyexpの実行結果

カーソルキーの左右を押すと長方形が左右に移動する。実行直後にスペースキーを押しても何も起こらないが、10秒以上経過した後にスペースキーを押すと終了する。

時間の計測に内部変数tを使っていることに注意。

オブジェクトのメソッドを利用する

- メソッド
  - オブジェクトそれ自体に対する処理を定義したもの。
  - オブジェクトのメソッドを呼び出す時には、データ属性と同様に . 演算子(ドット演算子)を使う。

poly.contains(mouse) → polyという変数に格納されたオブジェクト内にマウスカーソルがあるか否か判定する

trials.getEarlierTrial(-1) → trialsループで1回前の繰り返しの時のパラメータをdictオブジェクトとして得る

メソッドには非常にたくさんの種類がある

- **[作業7 (sample07.psyexp)]**
  - Mouseコンポーネントを1個配置し、**Name**を**mouse**とする。
  - Polygonコンポーネントを1個配置し、**Name**を**poly**とする。
  - **mouse**と**poly**の**Stop**を空欄にする。
  - **kb**の**Force end of Routine**のチェックを外す。
  - **poly**の**N Vertices $**に**3** 、**Color**に$colと入力し、set every frameにする。
  - **poly**の**Position [x, y] $**に**[0.5*cos(t), 0.5*sin(t)]**と入力し、set every frameにする。

次ページに続く

- **[作業7 (sample07.psyexp)] つづき**
  - Codeコンポーネントを1個配置し、**Begin Experiment**に以下の式を入力する。

```
col = 'green'
```

  - Codeコンポーネントの**Each Frame**に以下の式を入力する。

```
if poly.contains(mouse):
    col = 'red'
else:
    col = 'green'
```

  - ポイント
    - sin(x)、cos(x)はそれぞれxの正弦、余弦を返す。
    - A * BはAとBの積を表す。

sample07.psyexpの実行結果

三角形が反時計回りの楕円軌道を描いて移動する。マウスカーソルを操作して三角形に重ねると、三角形が赤色になる。

「PsychoPy Builderで作る心理学実験」との対応

- サンプルファイルと関連が深い章・節
  - sample01：第5章 5.4
  - sample02：第5章 5.5
  - sample03：第3章トピック3-1、第6章 6.8
  - sample04：第7章 7.4
  - sample05：第6章 6.7、6.8
  - sample06：第7章 7.3、第9章も参照
  - sample07：第8章 8.6
- その他参考になる節
  - 付録 11.4 (内部変数一覧)
  - 表6-3 (Keyboard)、表7-1、7-2 (TrialHandler)、表8-3 (Mouse)