Q&A(よくある質問とサンプルプログラム)

ここでは、ドリトルを使ってプログラムを作っている方から寄せられた質問などのうち、 問い合わせが多いものについてサンプルプログラムを例示します。

Q1:描いた図形をアニメーションで動かしたい

かめたで描いた図形は、「図形を作る」の命令で部品化できます。 部品化した図形は、タイマーの処理を利用することで一定間隔で移動することができます。 ここでは、棒という図形を描いて、これを右から左に動かしています。(対応□JAVA版/JS版)

```
//////step1 動かしたい図形を作る////////
かめた＝タートル！つくる。
「かめた！20 歩く 90 右回り　100　歩く　90 右回り」！2　繰り返す。
棒＝かめた！（青）　図形を作る。
棒！200 100 位置。

//////step2 図形を動かす//////
時計＝タイマー！作る。
時計！「棒！-20 0 移動」実行。
```

このプログラムをJS版ドリトルで動作確認する

Q2:かめた（タートル）で図を描いた後、線を描かずにかめたの位置を変更したい

かめたは「ぺんなし」の命令を利用することで、線を描かずに移動するようになります。 なお再び線を引かせたい場合は「ぺんあり」の命令を利用します。(対応□JAVA版/JS版)

```
//////step1 かめたを動かして迷路を描く////////
かめた＝タートル！作る。
かめた！１００ 歩く ９０ 左回り １００ 歩く ９０ 右回り
        ２００ 歩く ９０ 右回り ５０ 歩く ９０ 右回り
        １４０ 歩く ９０ 左回り １００ 歩く ９０ 右回り
        １６０ 歩く。
迷路＝かめた！図形を作る。

//////step2 かめたの位置を迷路の入り口に移動する//////
かめた！ぺんなし。
かめた！９０ 右回り ２５ 歩く ９０ 右回り。
```

このプログラムをJS版ドリトルで動作確認する

Q3:図形にぶつかったら、かめたを消したい

かめたの「衝突」の命令に、ぶつかった相手を消す命令を書くことで、 図形にぶつかった瞬間に、衝突

した対象の図形などを消去する動作を実現できます。(対応□JAVA版/JS版)

```
//////step1 図形を描く////////
かめた＝タートル！作る。
「かめた！100 歩く　90 右回り」！4　繰り返す。
四角＝かめた！（赤）図形を作る。
四角！250 50 位置。

//////step2 かめたの移動//////
時計＝タイマー！作る。
かめた！ぺんなし。
時計！「かめた！10 歩く」実行。

//////step 3 図形にぶつかったらかめた消滅//////
かめた：衝突＝「自分！消える。」。
```

このプログラムをJS版ドリトルで動作確認する

Q4:ゲームオーバーとゲームクリア表示をしたい

ゲーム中に、失敗した場合などに「ゲームオーバー」と表示し、 成功した時には「ゲームクリア」と表示したい場合があります。

このときには、フラグ（変数）と、タイマーの処理である「最後に実行」を利用することで これらを実現することができます。

下記の例では、開始から5秒間以上　かめたが図形にぶつからなければ「ゲームクリア」と表示し、 開始から五秒以内に図形にぶつかったら「ゲームオーバー」と表示します。

衝突により「：ゲーム状態」というフラグ（変数）の中身が変わることがポイントです。 衝突時に「時計」で実行中の動作を中断しているため、ぶつかった瞬間に「最後に実行」の内容が実行されます。 したがって、衝突するとすぐにゲームオーバーの文字が表示されます。

(対応□JAVA版/JS版)

```
//////step1 図形を描く////////
かめた＝タートル！作る。
「かめた！100 歩く　90 右回り」！4　繰り返す。
四角＝かめた！（赤）図形を作る。
四角！250 50 位置。

//////step2 かめたの移動//////
時計＝タイマー！作る。
時計！5秒　時間。
時計！「かめた！10 歩く」実行。

//////step 3 操作ボタン//////
操作ボタン＝ボタン！”右”作る。
操作ボタン：動作＝「かめた！２０　右回り」。

//////step 4 図形にぶつかったらかめた消滅＆ゲーム終了をチェック//////
：ゲーム状態＝””。
かめた：衝突＝「
```

```
    ：ゲーム状態＝”ゲームオーバー”。
    自分！消える。
    時計！中断。
□□

//////step 5 時計が中断したときの処理を判断して実行//////
時計！「
    「：ゲーム状態＝＝”ゲームオーバー”」！なら「
  ラベル！”ゲームオーバー”作る
    」そうでなければ「
  ラベル！”ゲームクリア”作る
    」実行。
」最後に実行。
```

このプログラムをJS版ドリトルで動作確認する

Q5:図形を塗る色を自分で作りたい

色オブジェクトを使うことによって、図形を塗るときの色を自由に作ることができます。 色の指定はRGBカラーを利用できます。 この例では、三角形を描き、桜色で塗っています。(対応□JAVA版/JS版)

```
/////////Step1:図形を描く/////////
かめた＝タートル！作る。
三角＝「かめた！60 歩く　120 右回り」！3　繰り返す　図形を作る。
/////////Step2:図形を塗る/////////
桜色 ＝ 色！255 200 200 作る。　//　赤　緑　青　の順番で混ぜる光の色を指定(0-255)
三角！（桜色）塗る。
```

このプログラムをJS版ドリトルで動作確認する

Q6:異なる動きを順番に実行するアニメーションを作りたい

”タイマー！「・・・」実行”の命令は、指定された回数（あるいは時間）だけ「」内に書かれた内容を繰り返します。一つのプログラムの中に”タイマー！「・・・」実行”の命令が2つ以上あるときは、一つ目の内容を指定した回数だけ繰り返した後に、次の繰り返す内容が実行されます。 下記は、こちらを利用して星を動かすプログラムです。(対応□JAVA版/JS版)

```
/////////Step1:図形を描く/////////
かめた＝タートル！作る。
星＝「かめた！60 歩く　144 左回り」！5　繰り返す　図形を作る。
星！（黄色）塗る。
/////////Step2:図形を異なる動きで順次うごかす/////////
時計＝タイマー！作る。
時計！ 0.1秒　間隔　2秒　時間。
時計！「星！5 5 移動する。」実行。
時計！「星！5 -5 移動する。」実行。
時計！「星！10 右回り。」実行。
時計！「星！消える。」実行。
```

このプログラムをJS版ドリトルで動作確認する

Q7:図形を異なるタイミングで動かしていくアニメーションを作りたい

タイマーのオブジェクトを複数作り、それぞれの実行タイミングを設定することで、 図形を動かす命令の実行タイミングを変えることができます。(対応JAVA版/JS版)

```
/////////Step1:図形を描く/////////
かめた＝タートル！作る。
三角＝「かめた！60 歩く　120 左回り」！3　繰り返す　図形を作る。
三角！（赤）塗る。
星＝「かめた！60 歩く　144 左回り」！5　繰り返す　図形を作る。
星！（黄）塗る。
/////////Step2:図形を異なるタイミングで動かす/////////
星時計＝タイマー！作る。
星時計！ 1秒　間隔　10秒　時間。
星時計！「
　星！10 右回り。
　星！-1 -1 移動する。
」実行。
三角時計＝タイマー！作る。
星時計！ 0.1秒　間隔　15秒　時間。
三角時計！「
　三角！ 1 1 移動する。
」実行。
```

このプログラムをJS版ドリトルで動作確認する

Q8:マウスカーソルを追いかけるアニメーションを作りたい

マウスのカーソル位置を取得して、その位置に図形を移動することができます。 下記は、描いた星マークをマウスのカーソル位置に移動するプログラムです(対応JAVA版)

```
/////////Step1:図形を描く/////////
かめた＝タートル！作る。
大星＝「かめた！60 歩く　144 左回り」！5　繰り返す　図形を作る。
大星！（黄色）塗る。
/////////Step2:マウスの位置に合わせて図形を動かす/////////
時計＝タイマー！作る。
時計！100秒　時間　0.1秒　間隔。
時計！「
　大星！(マウス！横の位置？) (マウス！縦の位置？)　位置。
」実行。
```

Q9:図形の中心位置に別の図形を配置したい

描いた図形の中心位置を取得し、その位置に別の図形を移動するようにプログラムすることで、アニメーション作成時に常に同じ位置に重なって表示する図形を書くことができます。 このサンプルでは、「三角」の図形の中心位置を取得し「点」図形をその位置に移動しています。(対応JAVA版)

```
/////////Step1:図形を描く/////////
```

```
かめた＝タートル！作る。
三角＝「かめた！60 歩く　120 左回り」！3　繰り返す　図形を作る。
三角！（黄色）塗る。
点＝かめた！10 円　図形を作る。
点！（赤）塗る。
座標表示＝ラベル□"X□---　□Y□---"□作る。
/////////Step2:三角の中心座標を取得して点図形を移動する////////
時計＝タイマー！作る。
時計！10秒　時間　0.1秒　間隔。
時計！「
　三角！ 2 -2 移動する。
　三角のX座標＝（三角□getx□□
　三角のY座標＝（三角□gety□□
　点！（ 三角のX座標）(三角のY座標)　位置。
　座標表示□("X□" □　（三角のX座標）　+ " □Y□" + (三角のY座標)　) 書く。
□
」実行。
```

Q10:図形に当たったら跳ね返るアニメーションを作りたい

タートルオブジェクトの「衝突」の命令に、「タートル：跳ね返る」を設定することで、 タートルが図形と衝突した時に、跳ね返ったような動作を追加することができます。(対応□JAVA版/JS版)

```
かめた＝タートル！作る。
かめた！15 線の太さ　。
円＝かめた！150 円　図形を作る　0　150 移動する。
かめた：衝突＝タートル：跳ね返る。
かめた！（乱数(360)）向き　ペンなし。
時計＝タイマー！作る。
時計！0.05秒　間隔　100秒　時間。
時計！「
　　かめた！ 5 歩く。
」実行。
```

このプログラムをJS版ドリトルで動作確認する

Q11:放物線を描く動きをしたい

図形の位置を、数式と繰り返し回数を利用して求めることで、 放物線を描く軌道の動きを実現することができます。(対応□JAVA版/JS版)

```
かめた＝タートル！作る　（赤）　線の色。
かめた：初期角度　＝ 45度。
かめた：初期位置x□　□□かめた!横の位置？。
かめた：初期位置y□□ □□かめた!縦の位置？。
かめた：初速度　＝　70。
時計＝タイマー！作る　8秒　時間　0.01　間隔。
時計□□□t0 □
　　t1 = t0 /10 .
□□□x0 = かめた：初期位置x□
□□□y0 = かめた：初期位置y□
```

```
      v0 = かめた：初速度。       // 初速度
      s0 = かめた:初期角度。      // 打ち上げ角度
□□□x位置□(x0 )  □(v0) * (t1) * cos( s0 ) □
□□□y位置□(y0) + (v0) * (t1) * sin( s0) - 9.8 * (t1) * (t1) □
      かめた□□x位置 □□y位置）位置。
  」実行。
```

このプログラムをJS版ドリトルで動作確認する

Q12:ボタンを押したら動きを変えたい

ボタンを押したことを契機に、別の動きを実現したい時はフラグ（変数）をうまく使うと実現できます。この例では、かめたの動き方を、ボタンを押す前後で変化させます。(対応□JAVA版/JS版)

```
直進状態＝はい。
動作変更ボタン＝ボタン！”動作変更”　作る。
動作変更ボタン：動作＝「
        ：直進状態　＝いいえ。
□□

かめた＝タートル！作る。
時計＝タイマー！作る。
時計！60時間。
時計！「
  かめた！5歩く。
  「（：直進状態）＝＝　いいえ」！なら「かめた！５　右回り。　」実行。

」実行。
```

このプログラムをJS版ドリトルで動作確認する

Q13:WebAPIやWebクライアントオブジェクトが通信できない場合の対処方法

学校のネットワークでは，プロキシサーバを通してインターネットと通信している場合があります□WebAPIやWebクライアントで通信ができない場合は，下記の設定を行い，通信ができるか試してみてください。

- dolittle.batをテキストエディタ（メモ帳等）で開く
- プロキシの設定を追加する（色囲み部分）

色囲み部分の意味は下記の通りです．利用するネットワーク環境に合わせてご変更ください。

```
 -Dhttp.proxyHost= *** □// ***の部分に利用しているプロキシ(http通信用)のドメインを設定してください.
 -Dhttp.proxyPort=8080 □// 8080の部分は実際に利用しているプロキシ(http通信用)のポート番号を設定してください.
 -Dhttps.proxyHost= *** □// ***の部分に利用しているプロキシ(https通信用)のドメインを設定してください.
 -Dhttps.proxyPort=8080 □// 8080の部分は実際に利用しているプロキシ(https通信用)のポー
```

ト番号を設定してください.

なおVer..40以降では，ドリトルのプログラム上からhttp/https通信用プロキシ設定をできるように機能変更を予定しています。

- http通信のプロキシ設定については Ver3.31時点においても，**システムproxy**を用いることで，ドリトルコード上からでも可能です。

Q12:双方向通信（サーバーオブジェクト）による通信ができない場合の対処法

学校のネットワークでは，各コンピュータがファイアーウォールを利用している場合，この設定内容によっては， サーバー機能による通信が遮断されてしまう場合があります.

この場合には2つの対応方法が考えられます.

■ファイアーウォールの設定を変更し，通信を遮断しないようにする.

ドリトルのサーバー機能は通信時に2020ポートを利用します.
ご利用のファイアーウォールの設定（通信ルールの設定）にて，2020ポートの送受信を許可してください.

なお，ドリトルのサーバ機能については，ローカルエリアのみの通信で完結しています.
2020ポートを利用して，インターネットに接続することはありません.
ファイアーウォールの設定時，2020ポートの通信許可は利用するネットーワークのセグメントのみに限定いただいてOKです.

■ファイアーウォールの設定は変更せず，ドリトル側で通信可能なポートを指定する.

ドリトルでは,
システムyyyサーバーポート。
の命令を 「サーバ“xxx”接続.」の命令より前に記載することで，通信で利用する通信ポートを指定することができますyyyの部分は利用が許可されている通信ポートに置き換えてください）

また，学習者に通信ポートを意識させたくない（上記の命令を書かせたくない）場合は，初期設定のファイルを作成することで，起動時に通信ポートの設定が完了した状態からプログラム作成をスタートすることができます.

【初期設定ファイルの作成手順】

1. dolittle.jar(dolittle.bat)と同じフォルダ上に，startup.iniという名前のテキストファイルを作る.
2. 作成したstartup.ini内に以下の記述をして保存するXXX部分は通信に利用するポートの番号を記載してください）

システムXXXサーバーポート。

■サーバー機能を利用する際の補足

- ドリトルのサーバー機能は「server の項目にチェックを入れたときのみ有効となります.
  ドリトルの未起動時,終了後，およびserver の項目にチェックを入れていない場合については通信を行いません.
  （通信自体が有効になりません）
- ドリトルのサーバー機能による通信は，LANのみで簡潔しインターネットへの接続を必要としません.

コンピュータ室のセグメントのみ通信を許可する設定をしたコンピュータ上でも動作可能です．

- 利用可能な通信ポートの設定やセキュリティポリシーは，組織ごとに異なります．
  コンピュータの設定状況は，利用可能なポートなどのについては，ネットワーク管理者または管理会社等にご確認ください．