Scratch とプログラミング思考

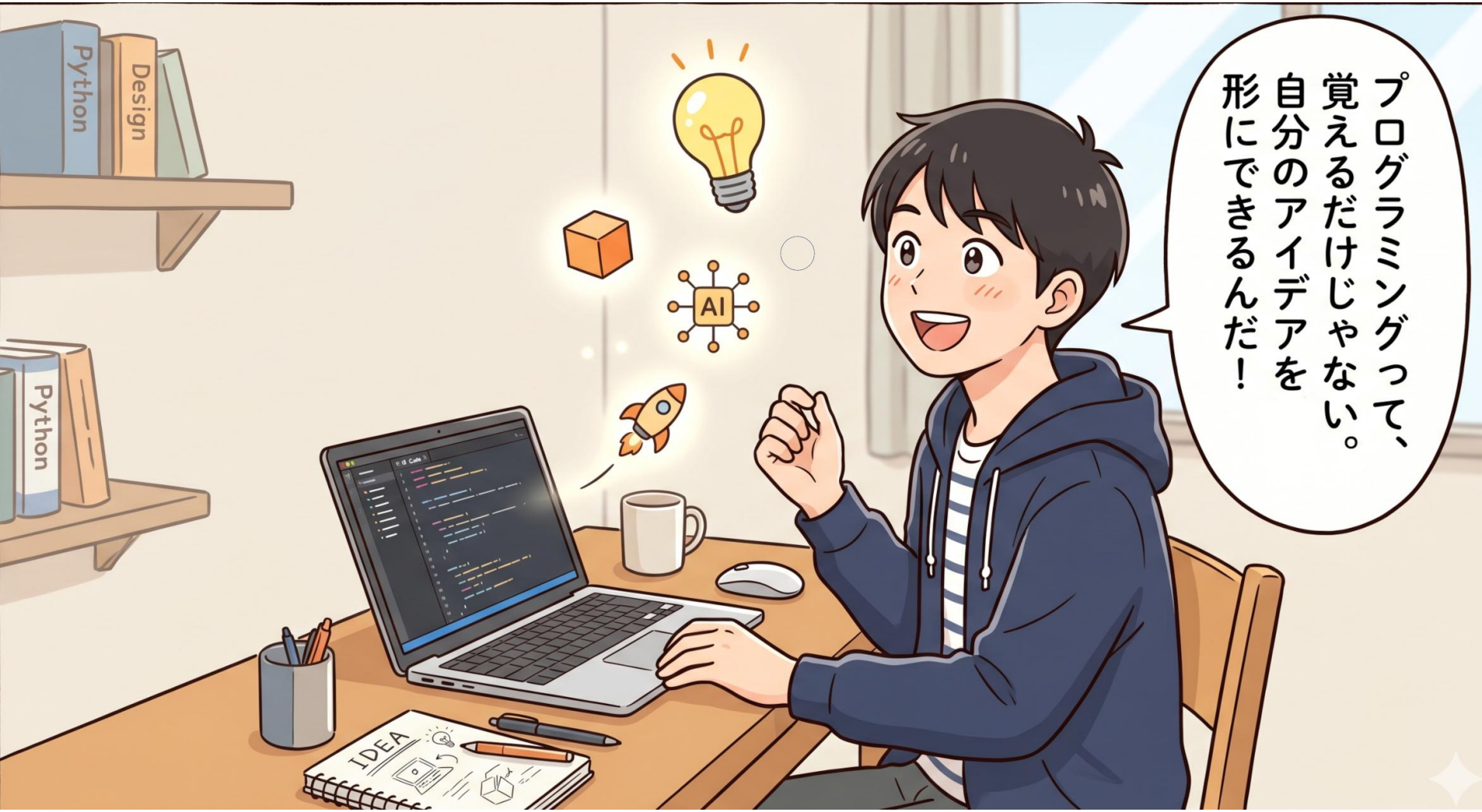
（Scratch）

URL: https://www.kkaneko.jp/pro/man/index.html

金子邦彦

資料中の図などは Google Nano Banana 2 を用いて作成

1. プログラミング
ー　人間の力を増幅する


プログラミングって、
覚えるだけじゃない。
自分のアイデアを
形にできるんだ！
Python
Design
Python
AI
IDEA

プログラミングは創造的　ー　授業の狙い

×

覚える、
問題を解くだけ

捉え直す

○

創造的な作業

プログラミング＝人間の力を増幅する営み

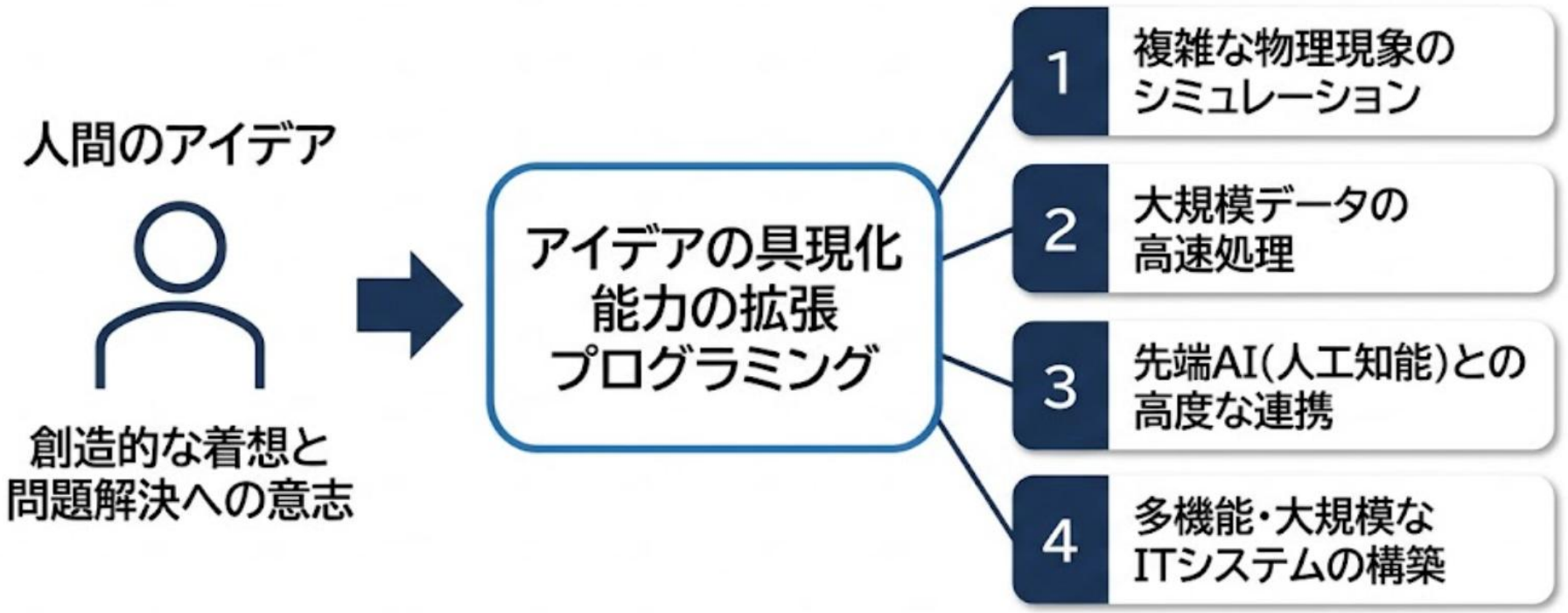
ゴール：応用へ進むための土台をつくる

説明パート
プログラミング的思考の枠組みを学ぶ

＋

演習パート
CodeCombat で楽しく体得する

プログラミングで何ができるのか



誤解:単なる『覚えて、問題を解くだけの勉強』

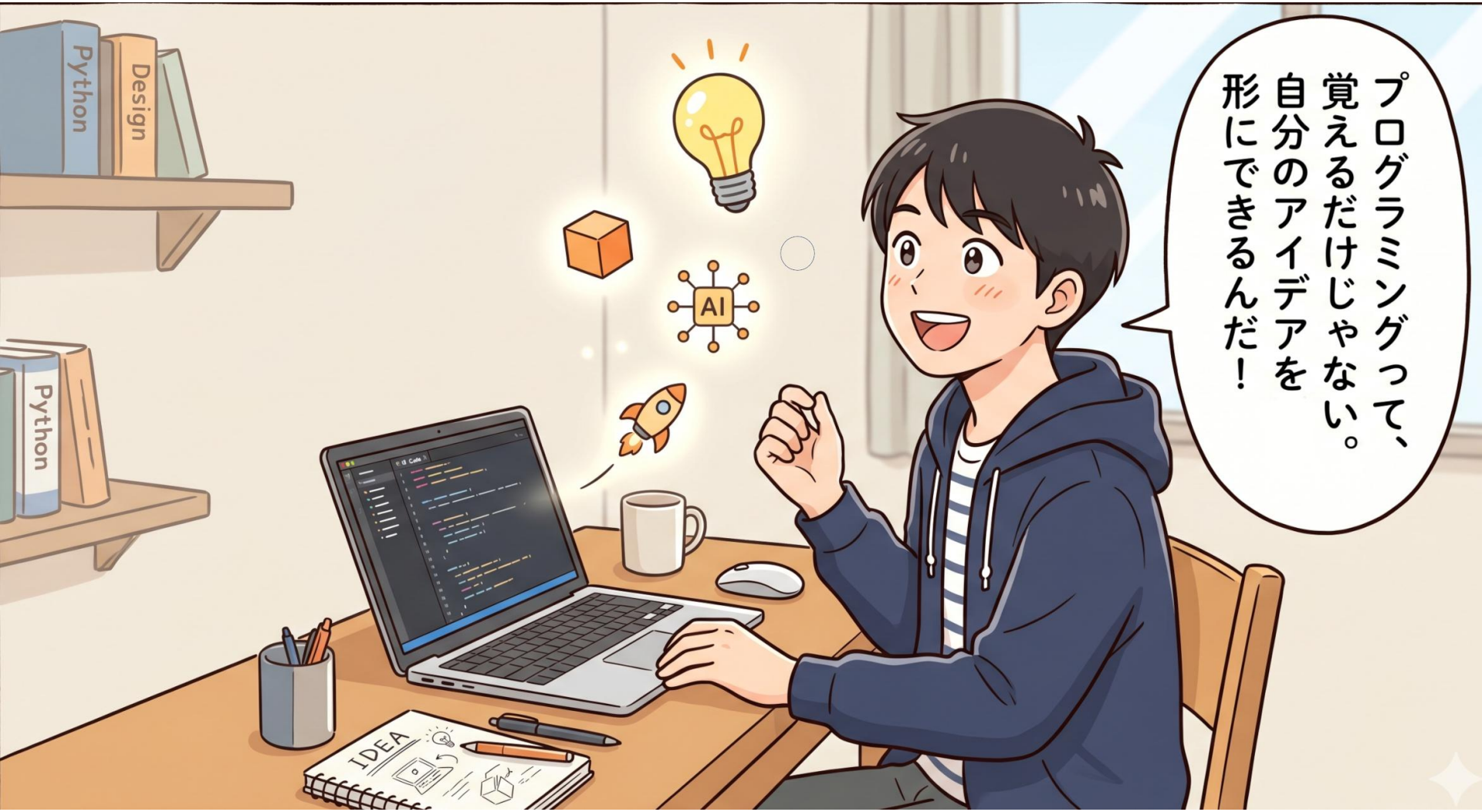
正解:『自分のアイデアを形にする』創造的な作業

プログラムとは ― 命令を書いた手順の集まり

だから複雑な作業も**自動化**・**効率化**できる

プログラミングを学ぶことで何が身につくか

7-1. プログラミングとは何か
ー　人間の力を増幅する


プログラミングって、
覚えるだけじゃない。
自分のアイデアを
形にできるんだ！
Python
Design
Python
AI
IDEA

情報工学科の学び

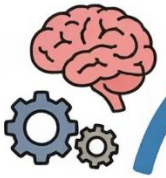
学ぶこと

コンピュータ

インターネット

デジタル活用

実践的な学びのサイクル



考える
（設計・論理）

（例：ロボットを正確に
動かすプログラミング）

書く
（プログラミング）

動かす
（実行・検証）

直す・調べる

実践による
実力習得

学習スタイル

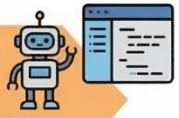
個人ワーク
（一人で取り組む）

グループワーク
（仲間と助け合う）

情報工学の面白さ

創造
（新しいアイデア）

設計
（システム全体）



具現化
（形にする・実装）

プログラミングは創造的　ー　授業の狙い

×

覚える、
問題を解くだけ

捉え直す

○

創造的な作業

プログラミング＝人間の力を増幅する営み

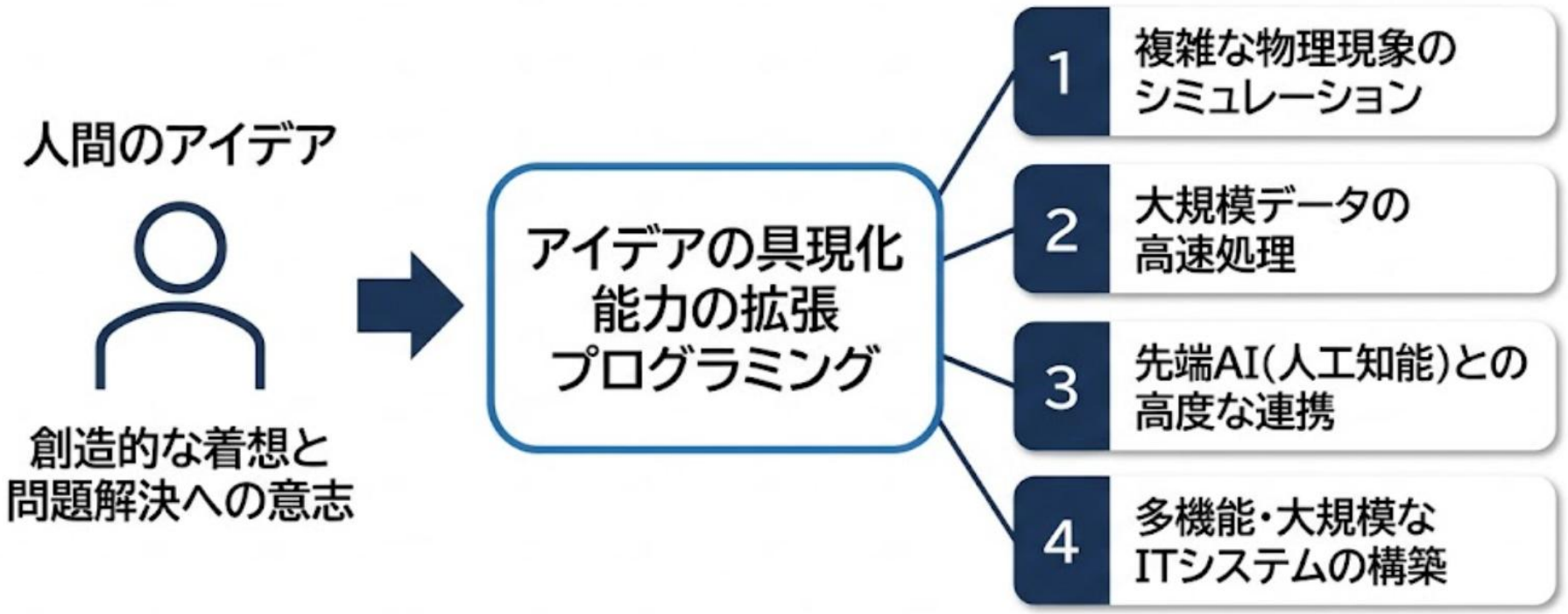
ゴール：応用へ進むための土台をつくる

説明パート
プログラミング的思考の枠組みを学ぶ

＋

演習パート
CodeCombat で楽しく体得する

プログラミングで何ができるのか



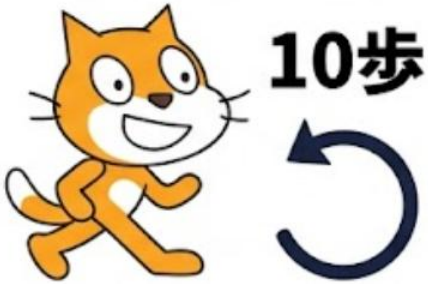
誤解:単なる『覚えて、問題を解くだけの勉強』

正解:『自分のアイデアを形にする』創造的な作業

プログラムとは ― 命令を書いた手順の集まり

プログラムは、命令を書いた手順の集まり

指示を与える → **自動で処理** → **結果を得る**



くりかえし

合計40歩進んだ!

だから複雑な作業も自動化・効率化できる

プログラミングを学ぶことで何が身につくか

これからは"AIに指示してプログラミングする"側に立つ ― LLMは使いこなす道具

プログラミング学習における LLM の活用

LLM とは

- 大規模言語モデル（Large Language Model）
- 自然言語でやり取りし、コードの生成・説明・修正を支援する AI

代表例：ChatGPT ／ Claude ／ Google Colab の AI アシスタント

特性を理解

学習パートナーとしての二面性

強み

強力な学習パートナーになる（コード生成・説明・修正を即座に支援）

注意点

誤ったコードを自信ありげに提示することがある

正しく使う

望ましい使い方

1. 生成されたコードは必ず自分で実行して確かめる
2. なぜそう書くのかを LLM に説明させる
3. まずは LLM に頼らず基本概念を体得する

LLM活用の具体例

大前提：生成されたコードはそのまま信用せず、自分で実行して結果を確かめること

活用例1：コードの生成を頼む

依頼の例
Pythonで [1, 2, 3] と [4, 5, 5] の折れ線グラフを描くコードを書いて

↓

LLMの応答
plt.plot(...) を使ったコードが返ってくる

↓

学習者の行動
Colabなどで実行し、
思いどおりのグラフが出るかを確認する

活用例2：コードの意味を説明させる

依頼の例
この1行のどこがオブジェクトで、どこがメソッドで、どこが引数か説明して

↓

LLMの応答
各部分の役割を解説してくれる

↓

学習者の行動
『なぜそう書くのか』を
説明させることで、自分の理解を固める

LLM活用の具体例 – やるべき使い方とやってはいけない使い方

エラーの原因を尋ねる

推奨される使い方

実行してエラーが出る
↓
エラーメッセージを
そのまま貼り付ける
↓
「このエラーの意味と
直し方を教えて」と質問
↓
示された直し方を理解し、
自分で修正して再実行

書き換え（リファクタリング）を頼む

推奨される使い方

- 「このコードを変数を使って書き直して」
- 「score = score + 1 のような再代入の例に書き換えて」

学んだ概念に沿った書き換えを依頼することで、概念と実コードの対応が見えやすくなる

やってはいけない使い方

避けるべき使い方

- 生成されたコードをコピペして提出するだけ
- 自分では一度も実行しない
- 意味を確認しない

強力な学習パートナーではなく、誤りを増幅する道具になってしまう

LLM は「自分で実行し、意味を確認する」使い方で初めて学習パートナーになる

7-2. プログラミング的思考と実践

ねこが進むほどスコアが増える ― この小さな世界から、プログラミングが始まる

ねこの行動と得点（スコア）

ねこが10歩進むたびにスコアが1増え、10回くりかえして10になる

10歩を10回くりかえすと、スコアは10になる

この世界を情報工学で分析

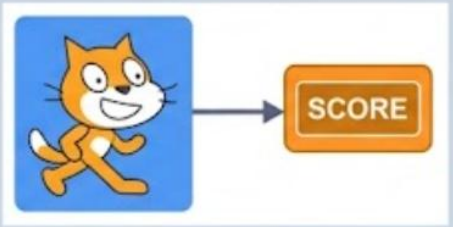
対象・構造・条件の3点で世界を捉える

対象（世界にあるモノ）

→ねこ：
この世界で
動く　主体

→スコア：
ねこの成果
果を測る量

構造（モノどうしの関係）



→ねこが前に進むこと
と、スコアが増えるこ
ことが結びついている

前進と得点が連動しているのが、
この世界のしくみ

条件（世界の決まり）

ねこは1回に
10歩ずつ進む

ねこが進むと、
スコアが1増える

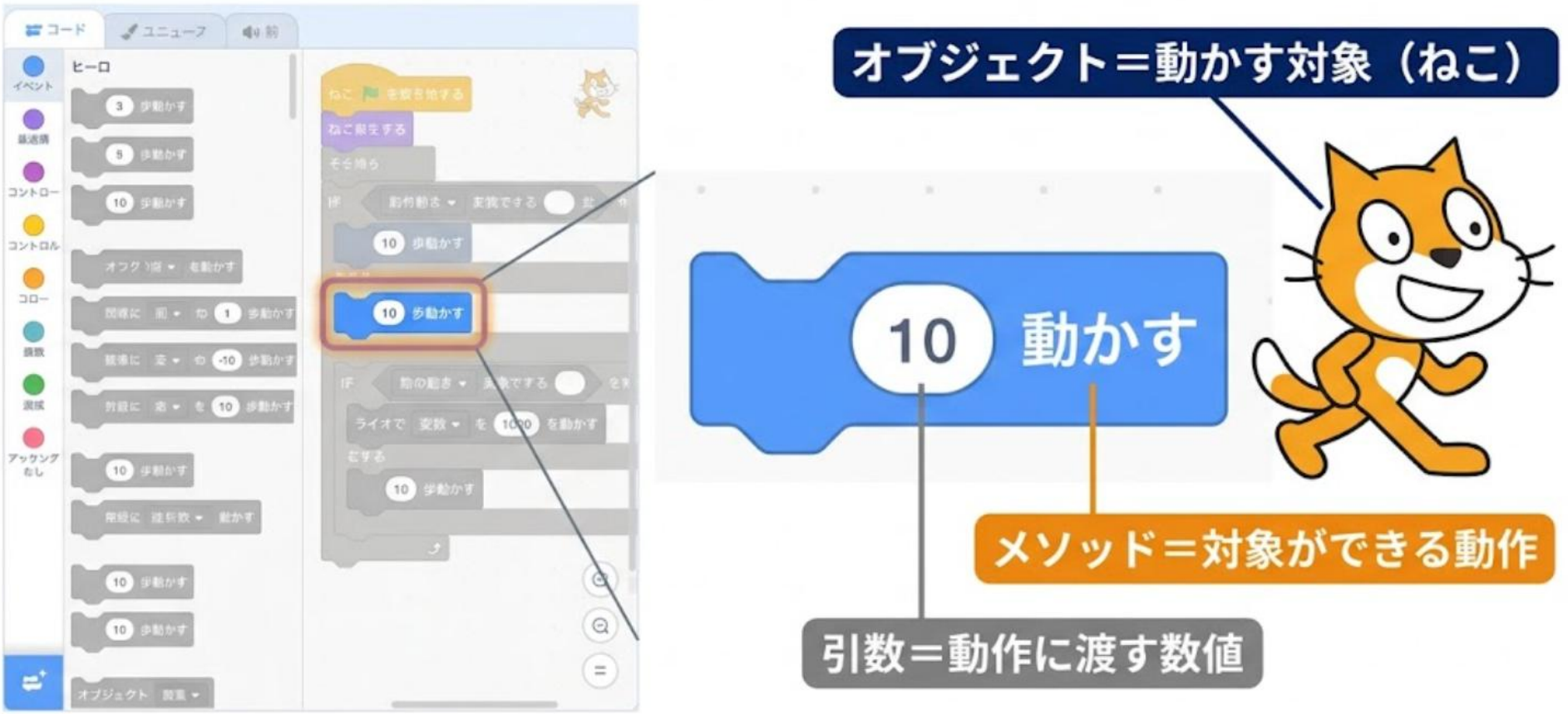
ねこの行動とスコアを
支配するルール

プログラムと『３つの道具』

このプログラムが『３つの道具』でできている

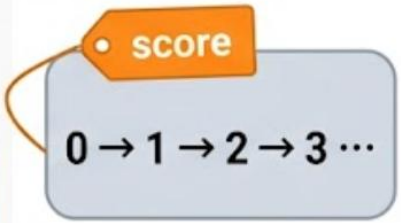
● ① オブジェクト・メソッド・引数　● ② 変数と代入　● ③ アルゴリズムの感覚

① オブジェクト・メソッド・引数



② 変数と代入

「score を 0 にする」
→ 代入：名札 score に値0を入れる（はじめの得点）



「score を 1 ずつ変える」
→ 再代入：今の score に1を足して入れ直す

③ アルゴリズムの感覚

手順1 → 旗が押されたとき （始まり1の合図）

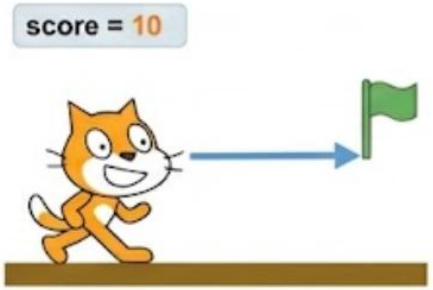
手順2 → score を 0 にする （準備）

手順3 → 10回繰り返す 10 回繰り返す （くりかえし）

手順4 → 10歩動かす+ scoreを1つ変える （くりかえす中身）

10 歩動かす

score を 1 ずつ変える



どんな順番でブロックを並べればゴールに着くかを考える力

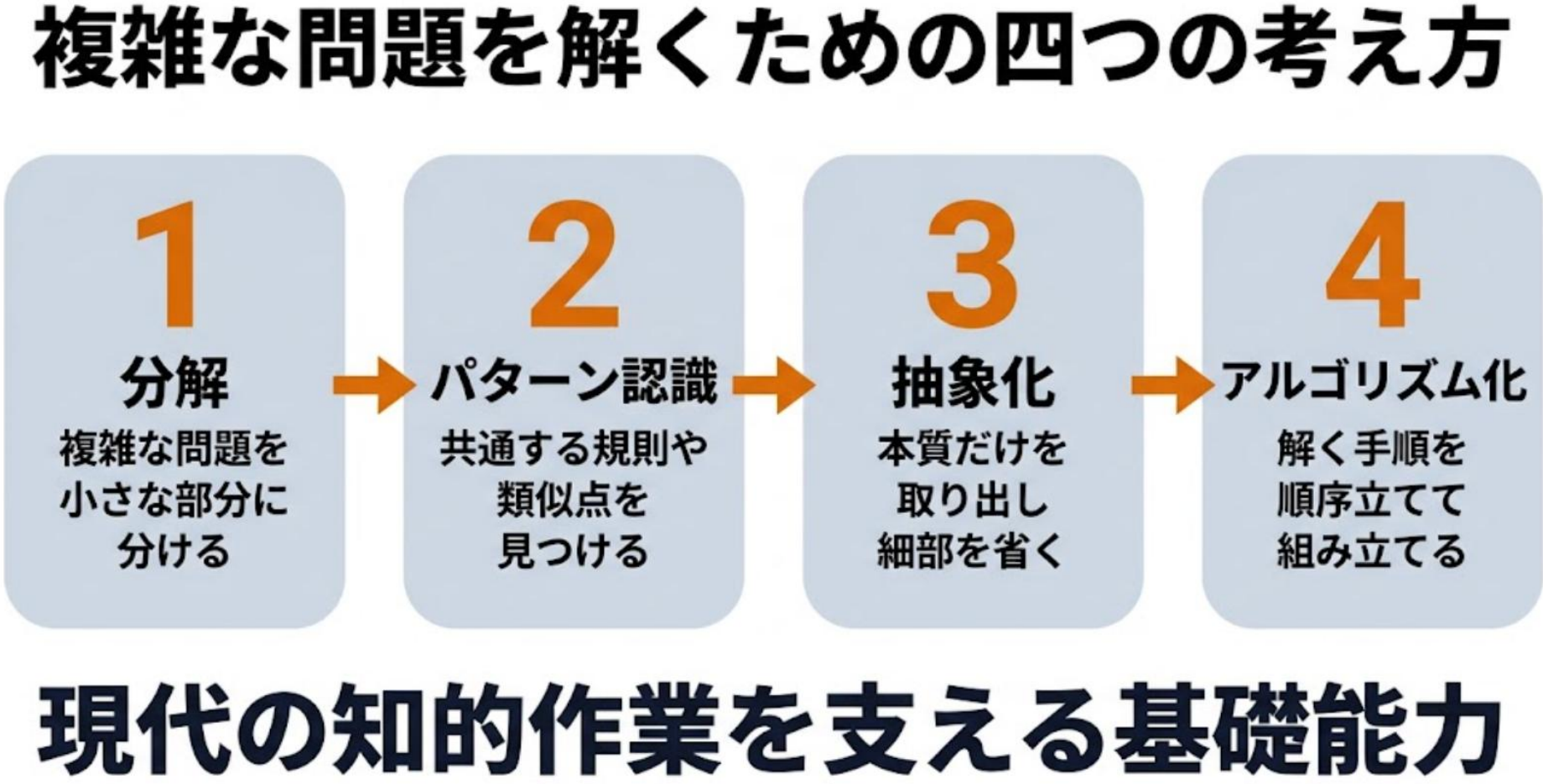
プログラムで使用される『３つの道具』

① オブジェクト・メソッド・引数
ねこ・動かす・10 のまとまり

② 変数と代入
score に値を入れ、書き換える

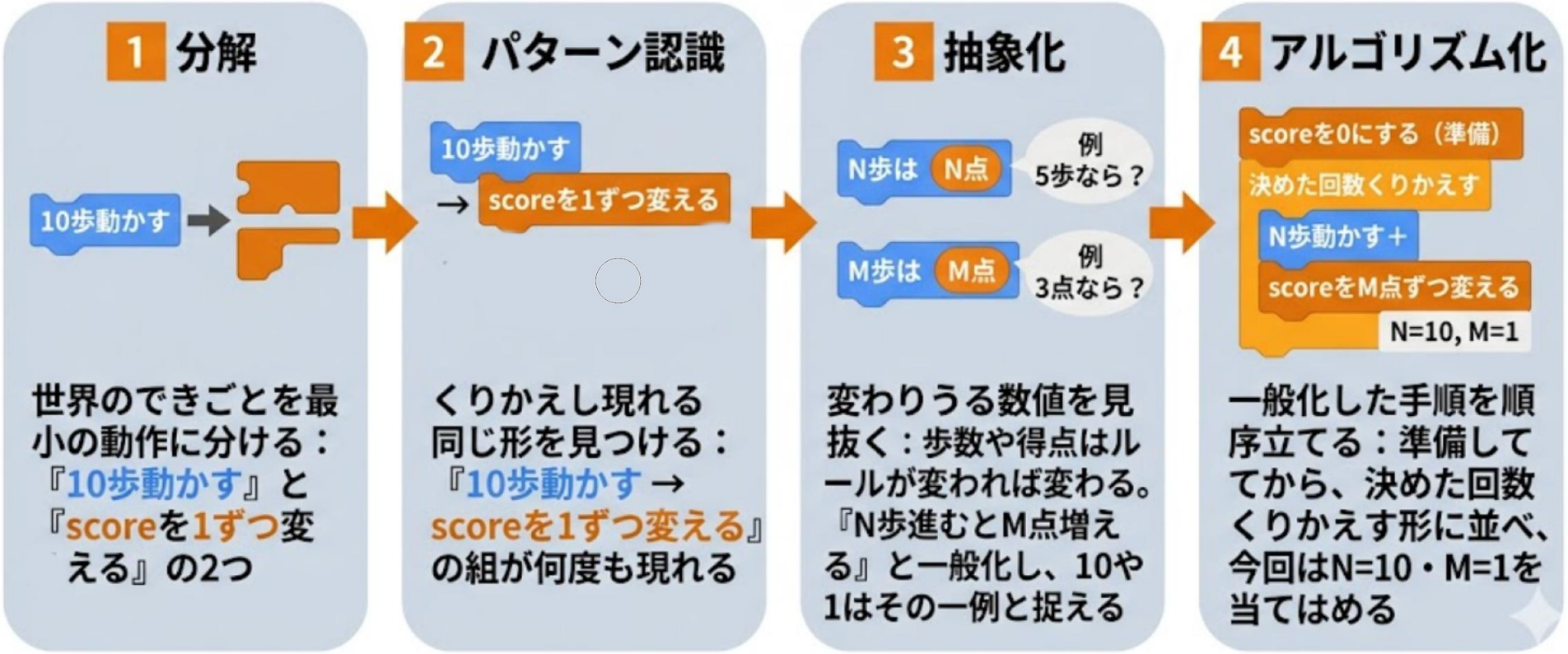
③ アルゴリズムの感覚
ブロックを正しい順に積み重ねる

プログラミング的思考

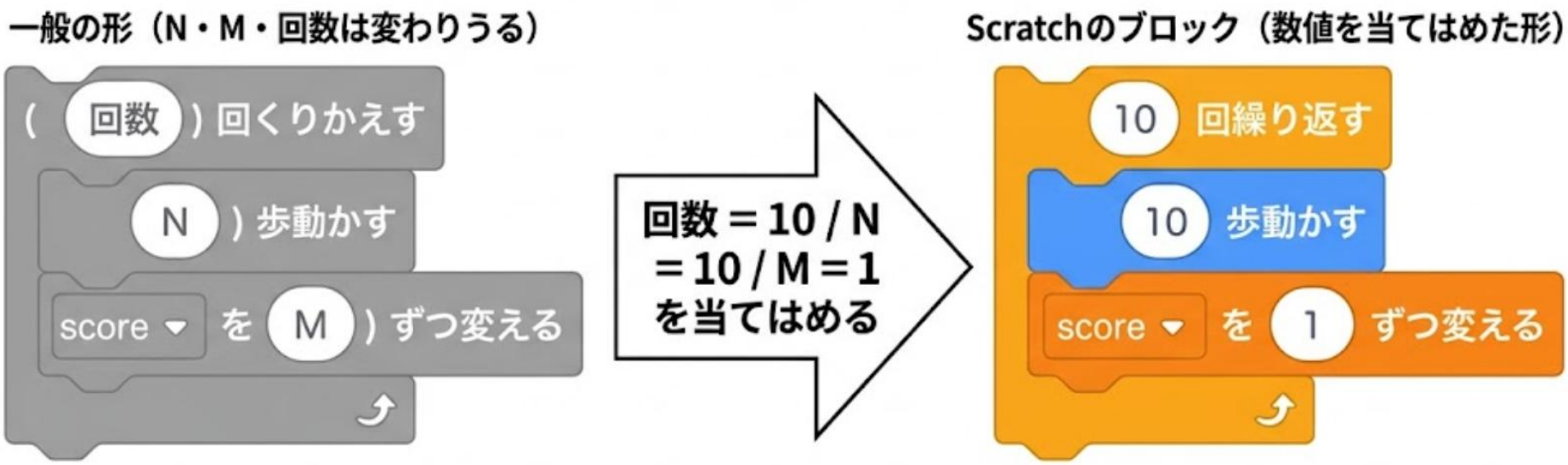

複雑な問題を解くための四つの考え方
1
分解
複雑な問題を
小さな部分に
分ける
2
パターン認識
共通する規則や
類似点を
見つける
3
抽象化
本質だけを
取り出し
細部を省く
4
アルゴリズム化
解く手順を
順序立てて
組み立てる
現代の知的作業を支える基礎能力

プログラミング的思考

『ねこが進むとスコアが増える』世界を、分解・パターン認識・抽象化・アルゴリズム化



アルゴリズム化の結果に、１０，１０，１の具体的な値を設定



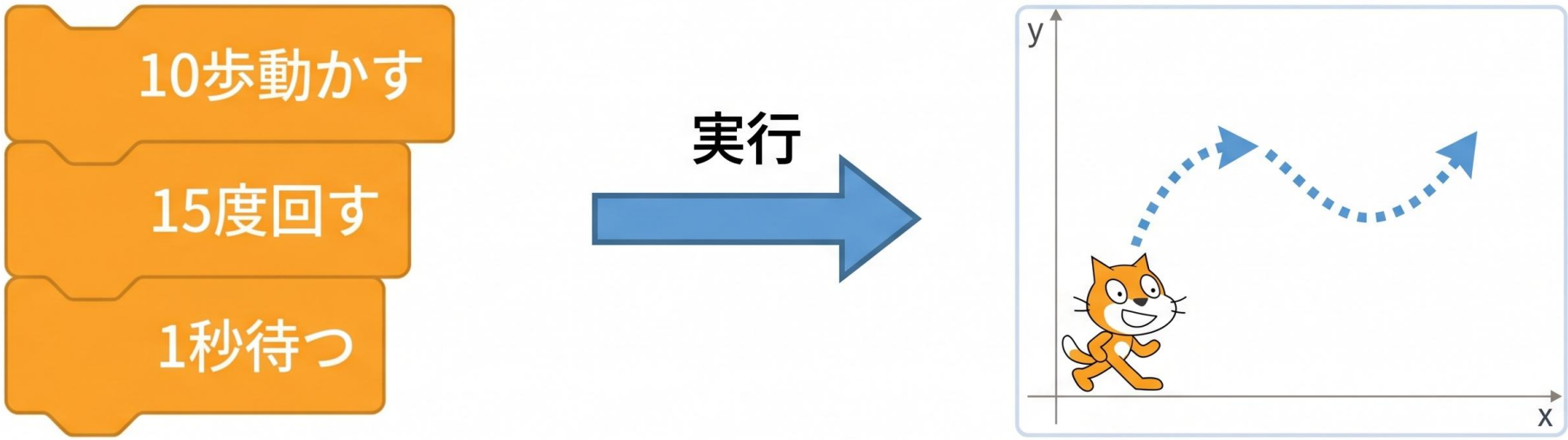
3. 変数に関する演習

ブロックを組み合わせるだけで、キャラクターを自在に操れる

Scratch：ブロックで動かすプログラミング

Scratch は scratch.mit.edu 上の Web ブラウザで動かす
(インストール不要)

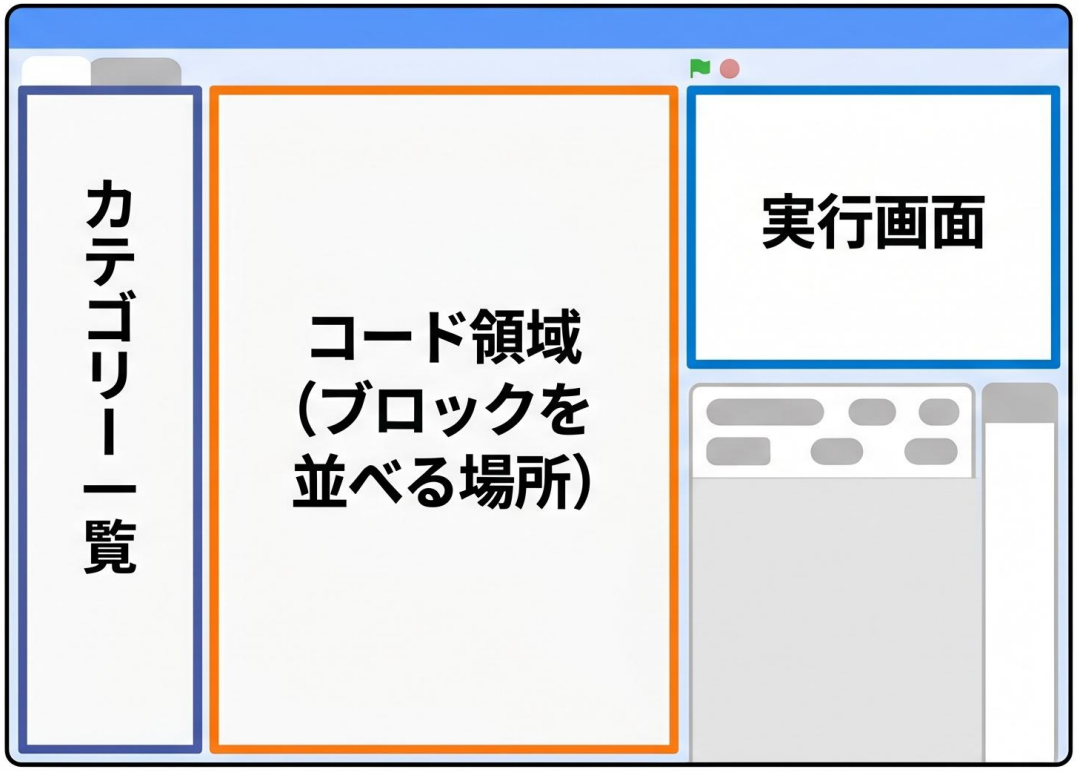
① プログラムを作る　② 命令の通りに　③ キャラクタが動く



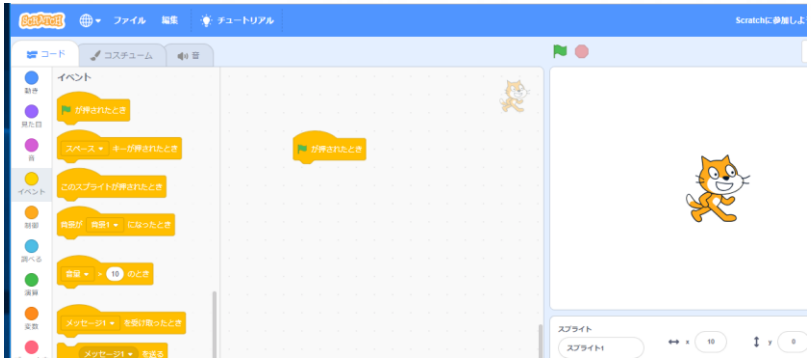
ブロックを組み合わせるだけで、キャラクタを自在に操れる

Scratchの画面構成と操作の基本

画面構成

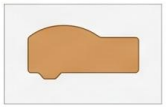


実際の画面



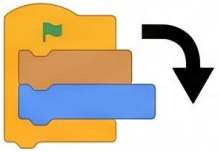
ブロック操作の基本

① ブロックを置く



② ブロックを組み合わせる

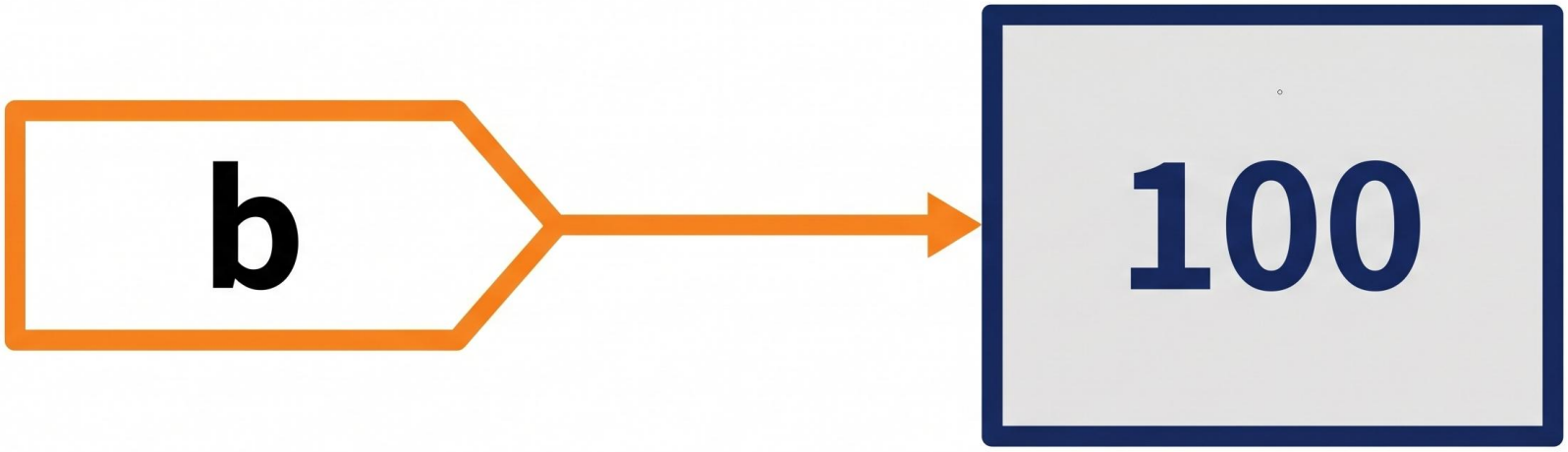
③ プログラムの起動



ブロックの動作が
順に実行される

変数

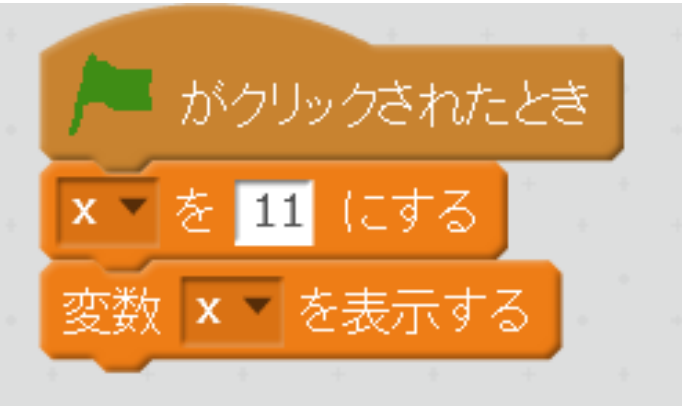
変数は『名札』、オブジェクトは『実体』。名札が実体を指し示す。



変数（名札）　　**オブジェクト（実体）**

1. **データに名前を付けて保管する**
2. **変数（名札）は実体を指し示すだけ**
3. **必要に応じて書き換えられる**

Scratch での変数



変数 x を使うような
プログラム

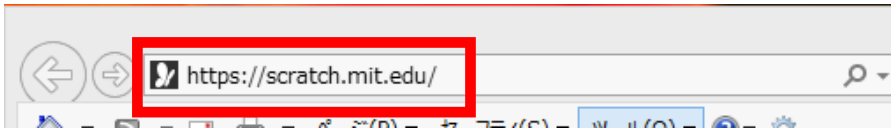
変数 x の値の表示結果

演習

- パソコンにログインする
- Webブラウザを起動する
- Webブラウザで、次のURLを開く

  https://scratch.mit.edu/

Webブラウザの起動



- 「やってみよう」をクリック

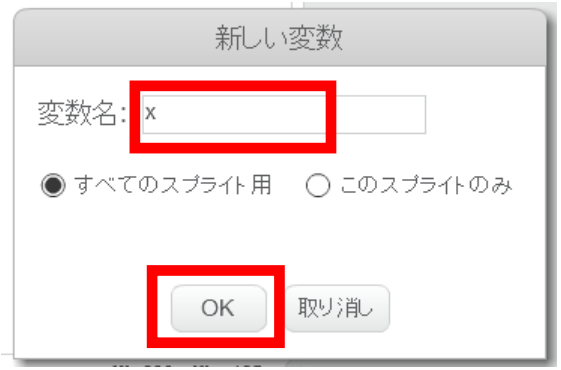
※　次ページに続く

演習

5. 「データ」をクリックし，「変数を作る」をクリック

6. 変数名を「x」．（半角の x です）

「OK」をクリック.



※　次ページに続く

演習

7. 「x を 0 にする」と「変数 x を表示する」を置き、この２つを組み合わせる

※　次ページに続く

演習

8. 「イベント」をクリック. 「 がクリックされたとき」を組み合わせる

組み合わせる

※　次ページに続く

演習

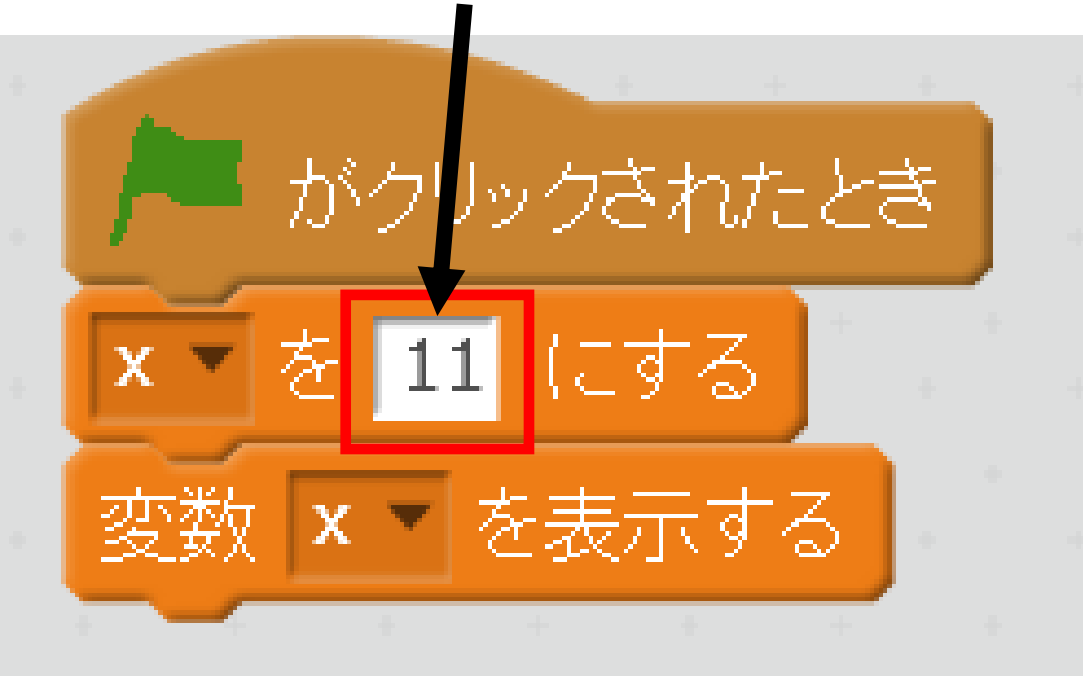
9. 実行ボタンをクリックすると，「x 0」のように表示されるので確認する

※　次ページに続く

演習

10. 「x を 0 にする」を「x を 11 にする」に書き換えてみる

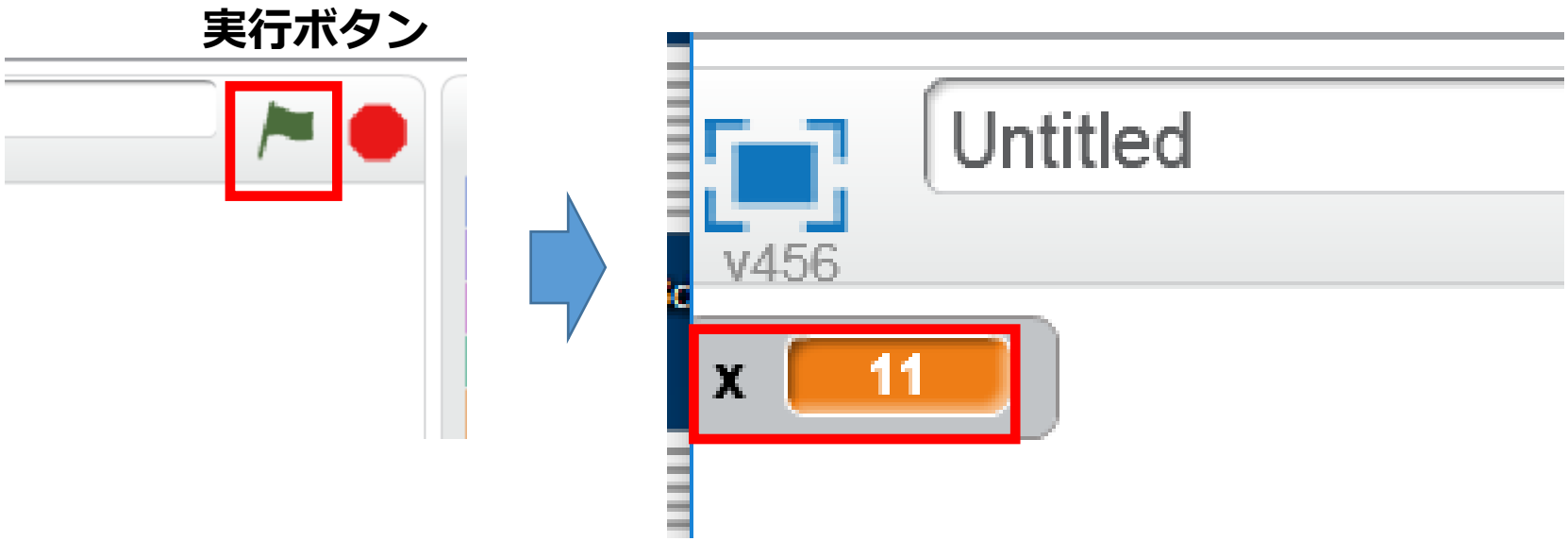
※　半角の「11」　（全角の１１では動かない）



※　次ページに続く

演習

11. 実行ボタンをクリックすると，「x 11」のように表示されるので確認する



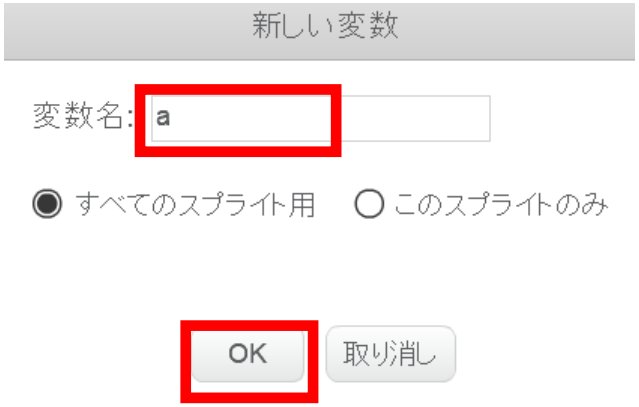
※　次ページに続く

演習

新しい変数 a を追加し，値を 100 に設定したい.

12. 「データ」をクリックし，「変数を作る」をクリック

13. 今度は，変数名を「a」．（半角の a ）
「OK」をクリック.

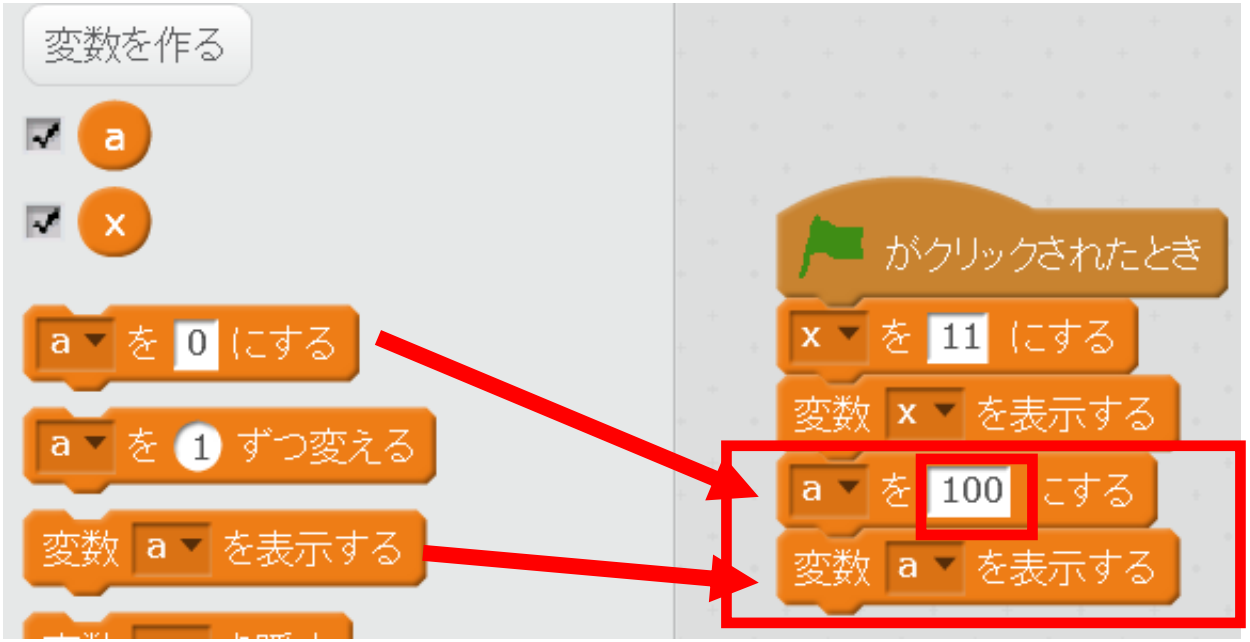


※　次ページに続く

演習

14. 「aを0にする」と「変数aを表示する」を組み合わせる.

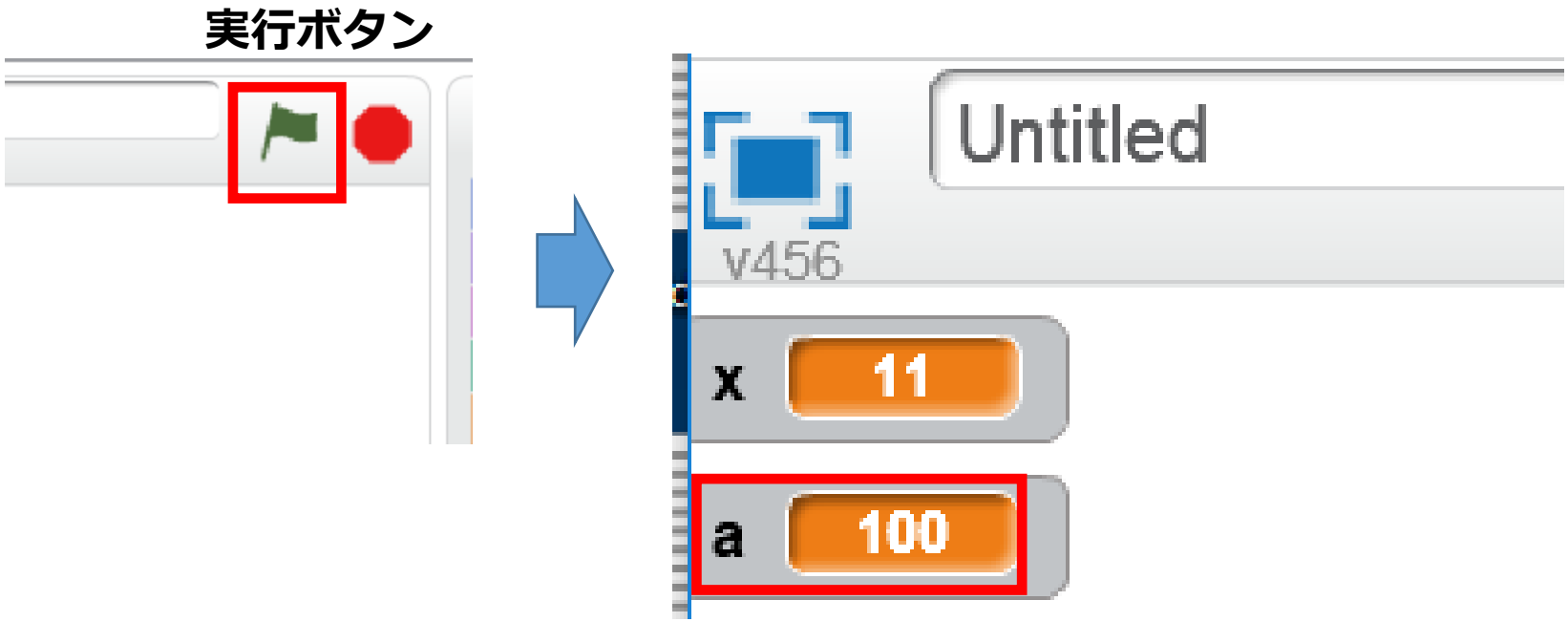
そして、「0」のところを「100」に書き換える（半角の「100」です）



※　次ページに続く

演習

15. 実行ボタンをクリックすると，「x 11」，「a 100」のように表示されるので確認する



※　次ページに続く

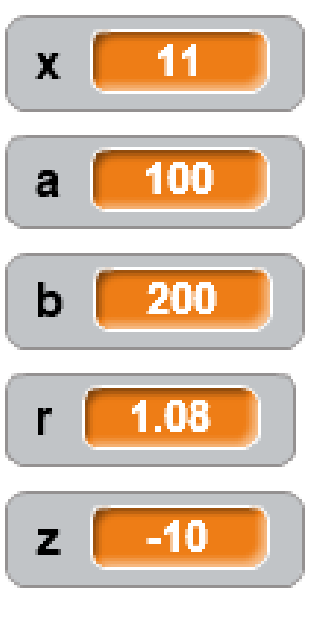
演習問題

◆新しい変数 b を追加し，値を 200 に設定しなさい

◆新しい変数 r を追加し，値を 1.08 に設定しなさい

◆新しい変数 z を追加し，値を -10 （マイナス10）に設定しなさい

◆下の図、左のように組み立てなさい



実行結果を確認しなさい

4. 式に関する演習

変数が役に立つのは

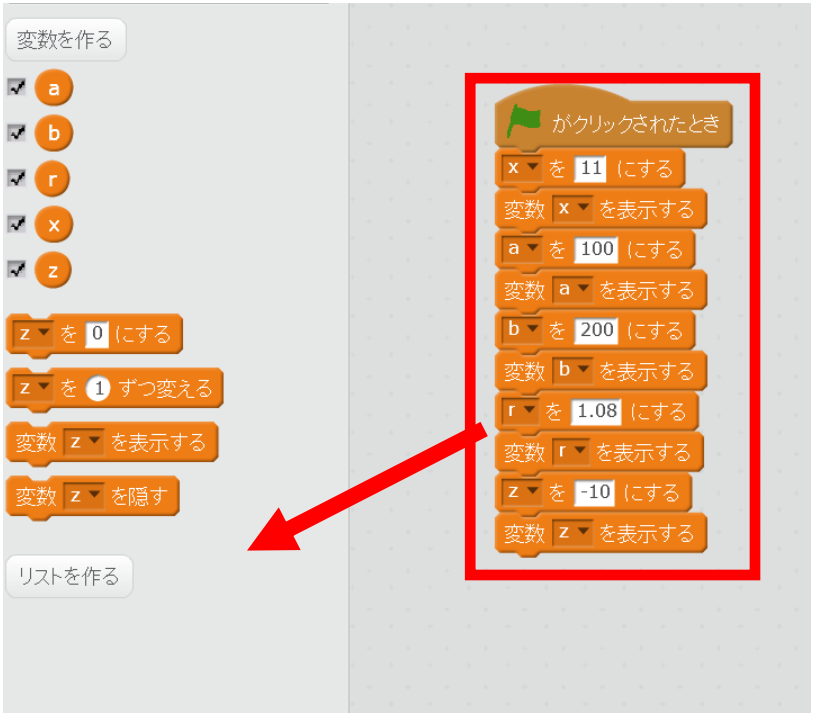
◆ データを記憶したいとき

◆ 同じような計算などを，値を変えながら何度も行いたいとき

演習

1. さきほど作成したブロックは不要なので，ブロックを

マウスの右ボタンを押しながら，中央エリアにドラッグする.



ドラッグすると
消える

※　次ページに続く

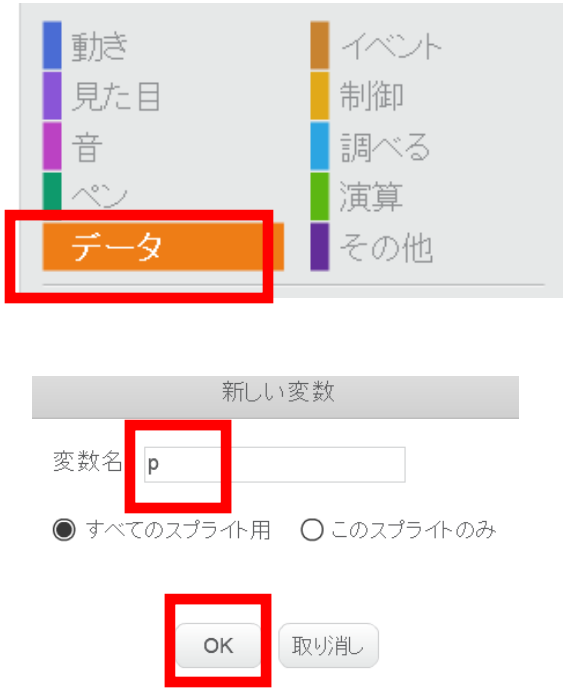
演習

2. 「イベント」をクリック．下の図のようにブロックを置く．

※　次ページに続く

演習

3. 「データ」をクリック.
新しい変数 p を作る



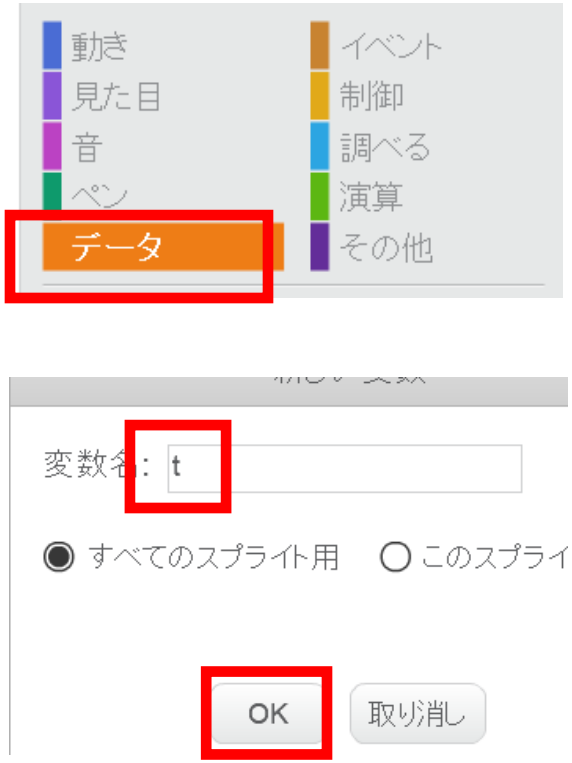
4. 下の図のようにブロックを組み合わせる.

※　次ページに続く

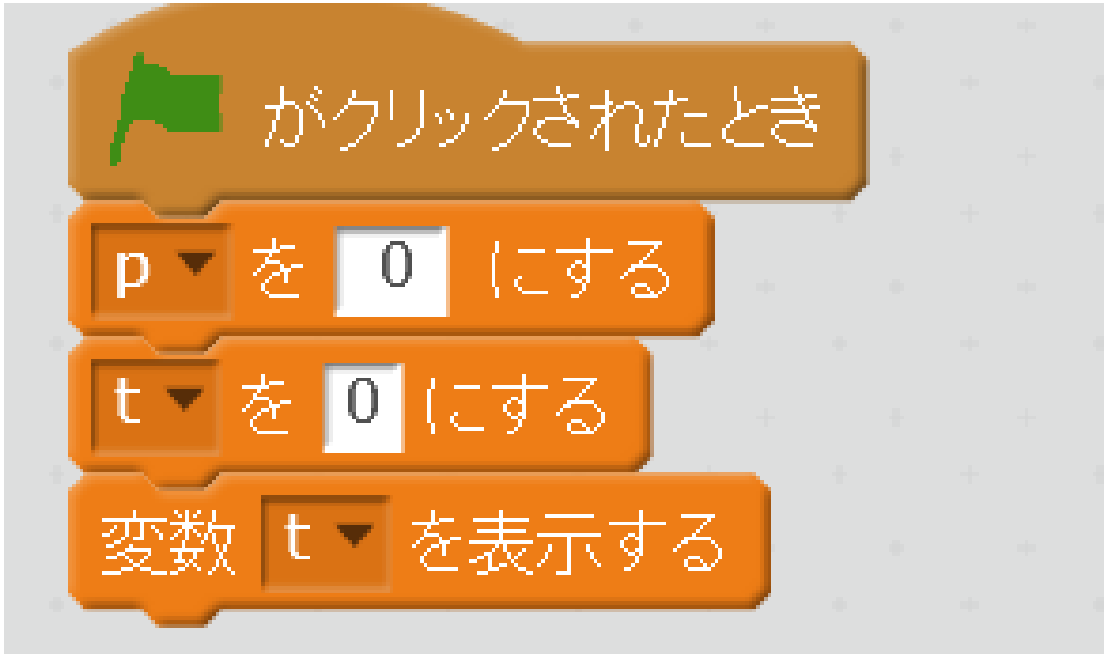
演習

5. 「データ」をクリック.

新しい変数 t を作る



6. 下の図のようにブロックを組み合わせる.

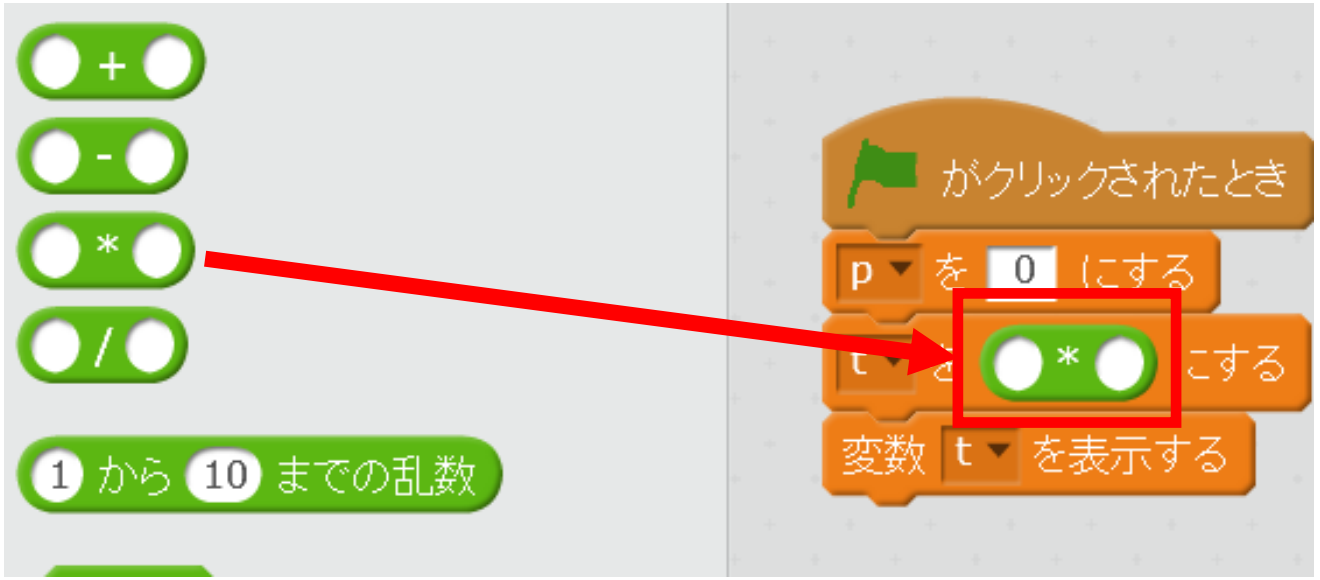


※　次ページに続く

演習

7. 「演算」をクリック.

8. 「○ * ○」のブロックを,
下の図のように組み合わせる.



※　次ページに続く

演習

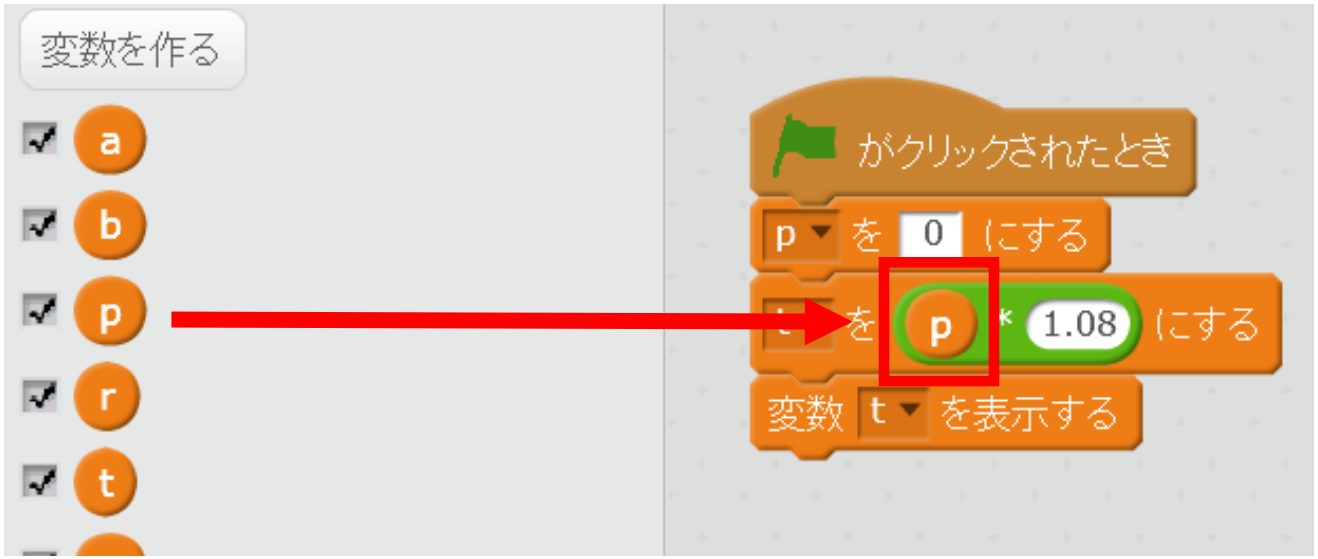
9. 「1.08」のように書き換える．「1.08」も半角

※　次ページに続く

演習

10. 「データ」をクリック.

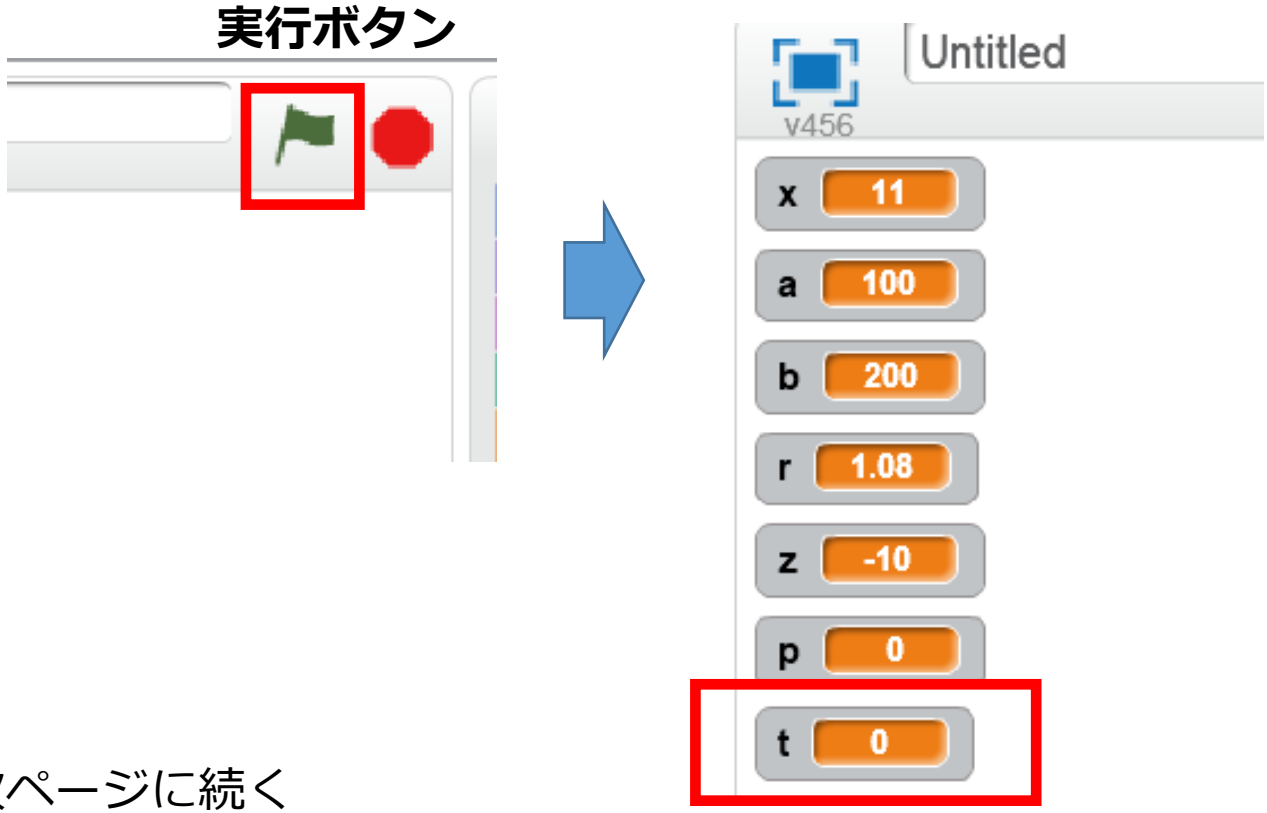
11. 下の図のように，変数 p のオレンジ色のブロックをはめ込む



※　次ページに続く

演習

12. 実行ボタンをクリックすると，「t 0」のように表示されるので確認する



※　次ページに続く

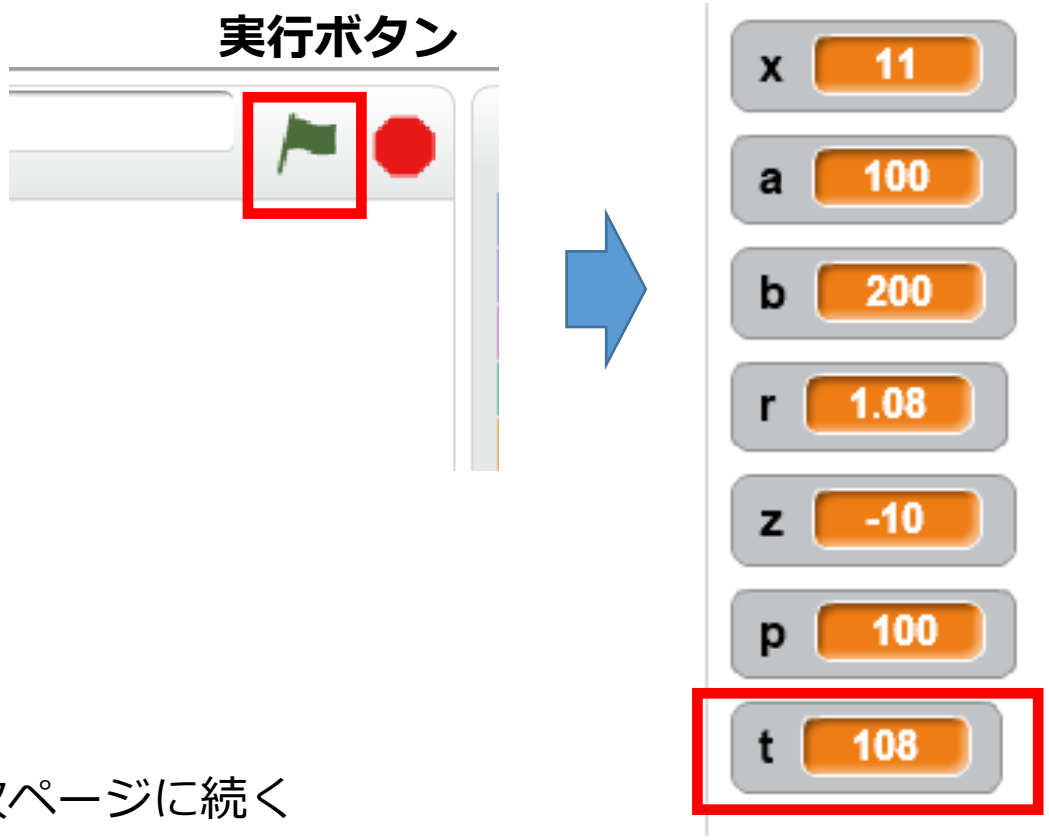
演習

13. 変数 p の値を 100 に変えて，100 * 1.08 の値を求めてみる

※　次ページに続く

演習

14. 実行ボタンをクリックすると，「t 108」のように表示されるので確認する



※　次ページに続く

演習課題

変数の値を変えて，同じ計算を再実行

◆ 変数 p の値を 1000 に変えて，1000 * 1.08 の値を求めなさい

◆ 変数 p の値を 2000 に変えて，2000 * 1.08 の値を求めなさい

◆ 変数 p の値を 4000 に変えて，4000 * 1.08 の値を求めなさい

※　次ページに続く

演習課題

２割引きの価格を求めたい

◆ 右のようにプログラムを作りなさい

◆ 変数 p の値を 100 に変えて，変数 t の値を確認

◆ 変数 p の値を 4500 に変えて，変数 t の値を確認

◆ 変数 p の値を 8000 に変えて，変数 t の値を確認

演習課題

正方形の面積を求めたい

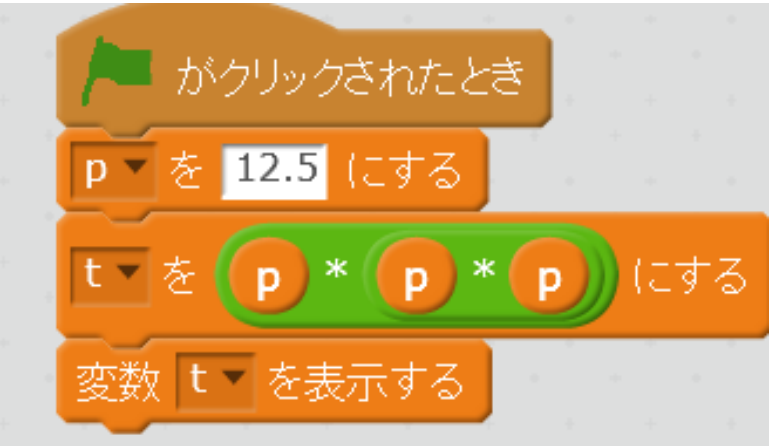
- 下のようにプログラムを作りなさい

- 変数 p の値を 10 に変えて，変数 t の値を確認
- 変数 p の値を 25 に変えて，変数 t の値を確認
- 変数 p の値を 12.5 に変えて，変数 t の値を確認

各自への演習課題

立方体の体積を求めたい

- 下のようにプログラムを作りなさい



- 変数 p の値を 10 に変えて，変数 t の値を確認
- 変数 p の値を 25 に変えて，変数 t の値を確認
- 変数 p の値を 12.5 に変えて，変数 t の値を確認

5. 条件分岐に関する演習

条件分岐

変数や式の値によって，結果が変わる

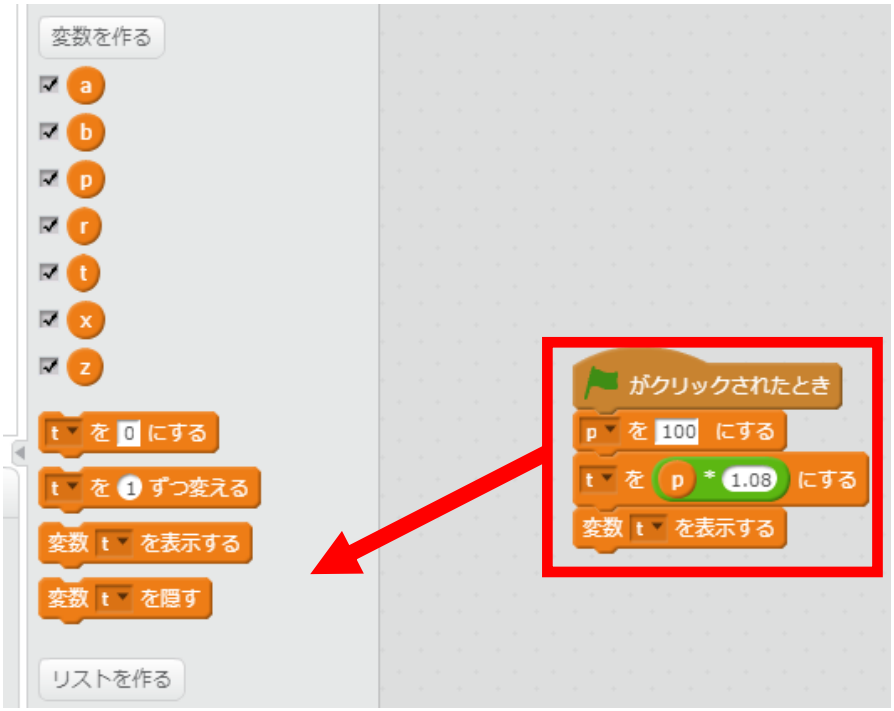
例えば

age の値が 20未満　→ 100 yen

20以上　→ 200 yen

演習

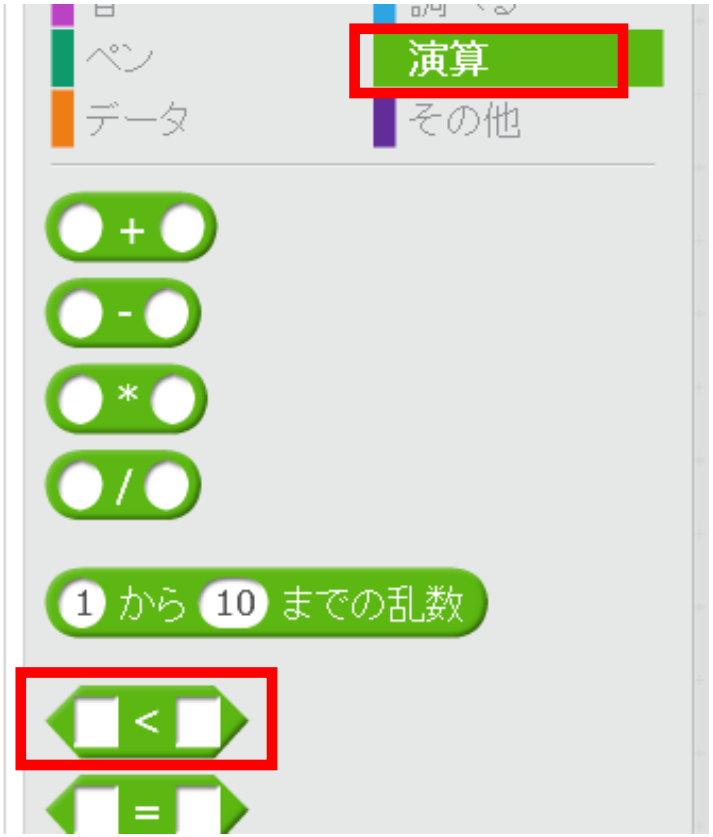
- 今まで作成したブロックは不要なので，ブロックを
- マウスの右ボタンを押しながら，中央エリアにドラッグする.



ドラッグすると
消える

演習

- 次のブロックを使う



演習

次の２つの変数を作る

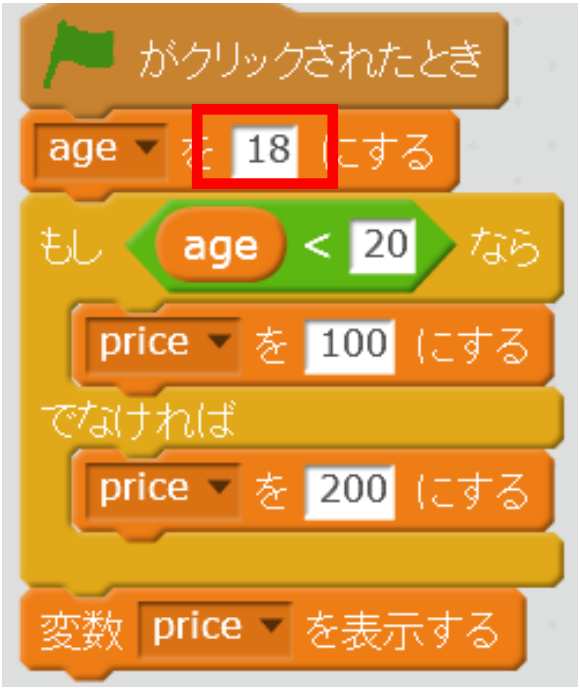
- age
- price

演習

ブロックを下のように組み立てる

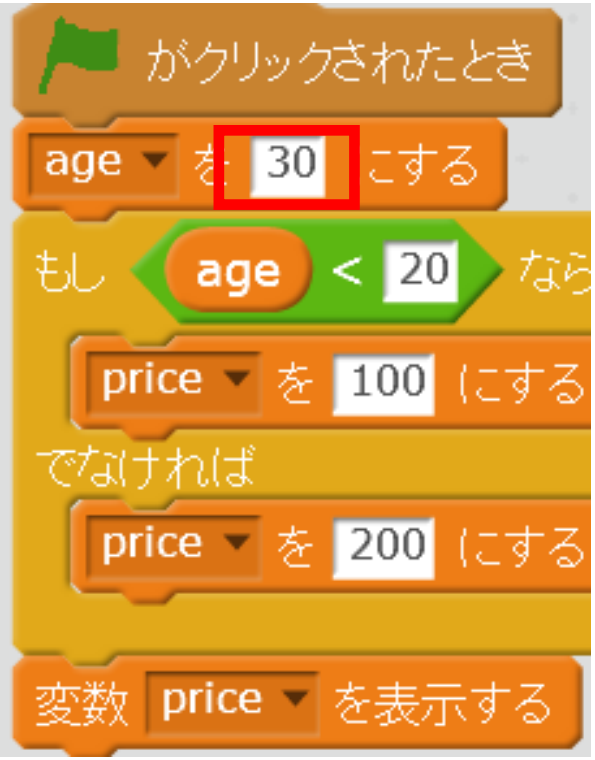
演習

- 変数 age の値を 18 にして，実行ボタンを押すと，price の値が 100 になることを確認



演習

- 変数 age の値を 30 にして，実行ボタンを押すと，price の値が 200 になることを確認



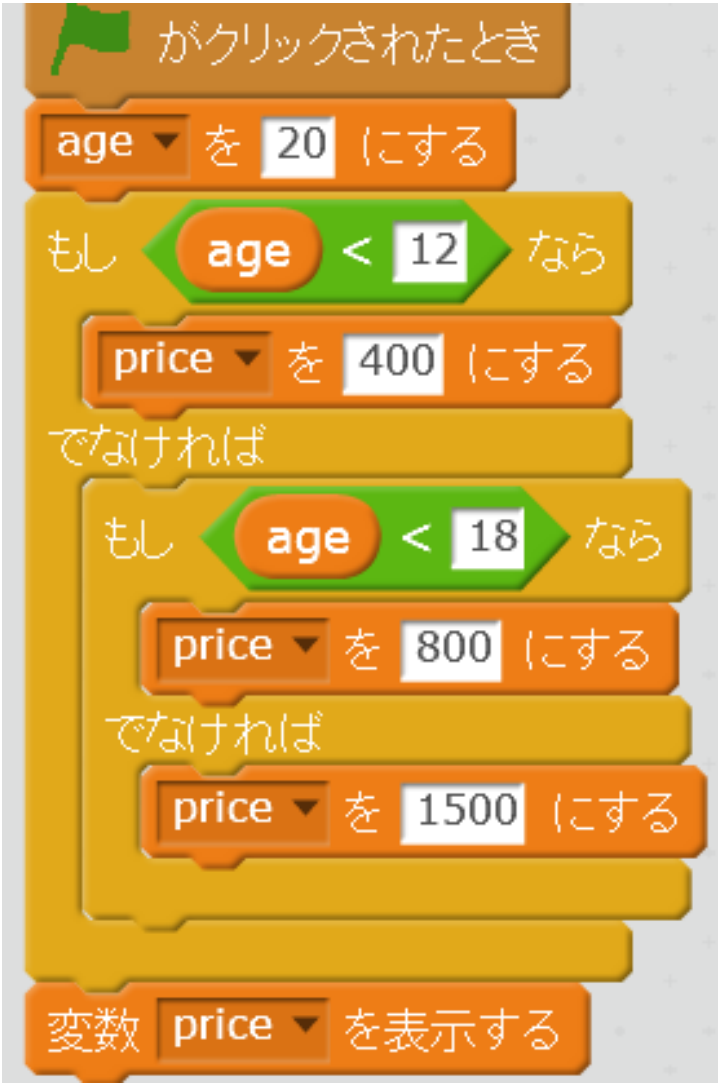
各自への演習問題

ある映画館は

- 12歳未満　400円
- 12歳以上18歳未満 800円
- 18歳以上　1500円

• 右のようにブロックを
組み立てなおしなさい

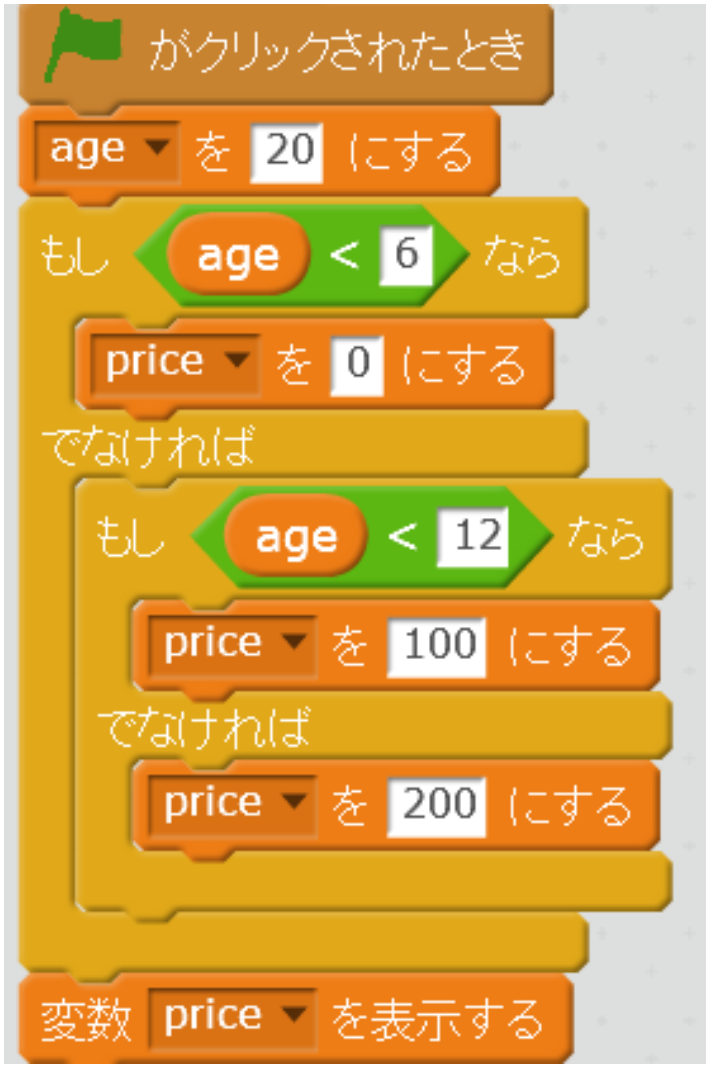
• そして，変数 age の値を
10, 15, 20 と変えて，実行



演習問題

あるバスは

- 6歳未満　　0円
- 12歳未満 100円
- 12歳以上 200円



• 自分で考えて，ブロックを組み立てなおしなさい

• そして，変数 age の値を 5, 10, 15, 20 と変えて，実行してみなさい

5. 繰り返しに関する演習

繰り返し

- 同じような処理を繰り返すこと.
- 変数の値を変えながら，繰り返すのが定石

演習

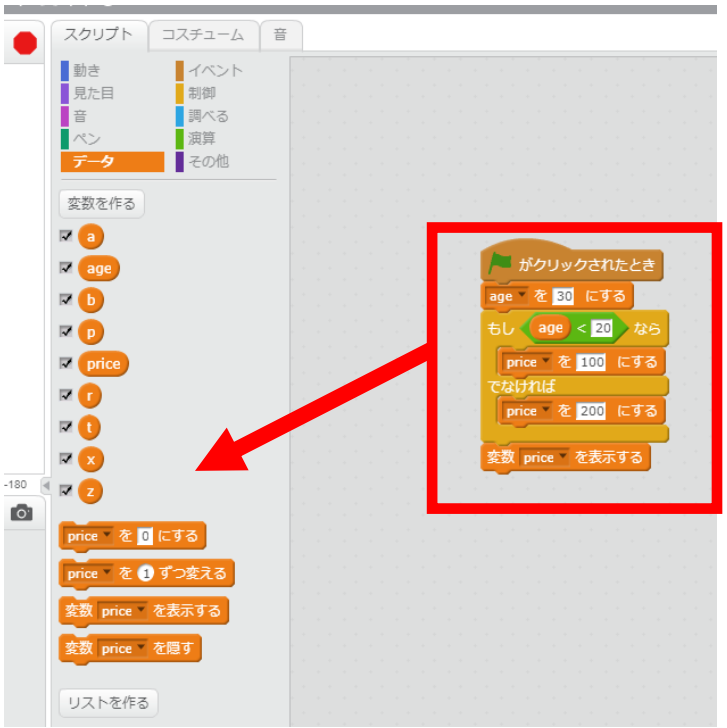
- 新しいブロックを使う

キャラクタを動かす

繰り返す

演習

- 今まで作成したブロックは不要なので，ブロックをマウスの右ボタンを押しながら，中央エリアにドラッグする.



ドラッグすると
消える

※　次ページに続く

演習

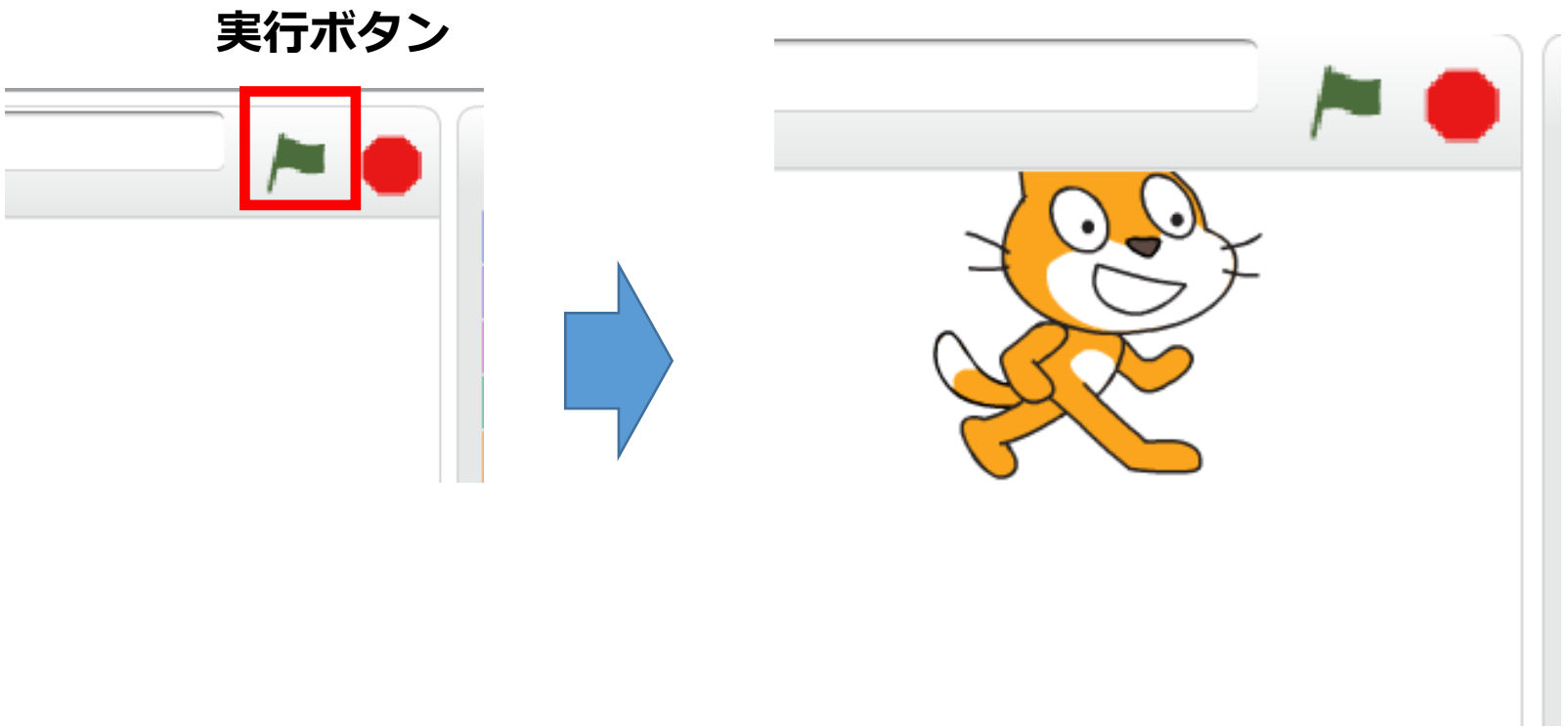
次の２つの変数を作る

- c
- d

演習

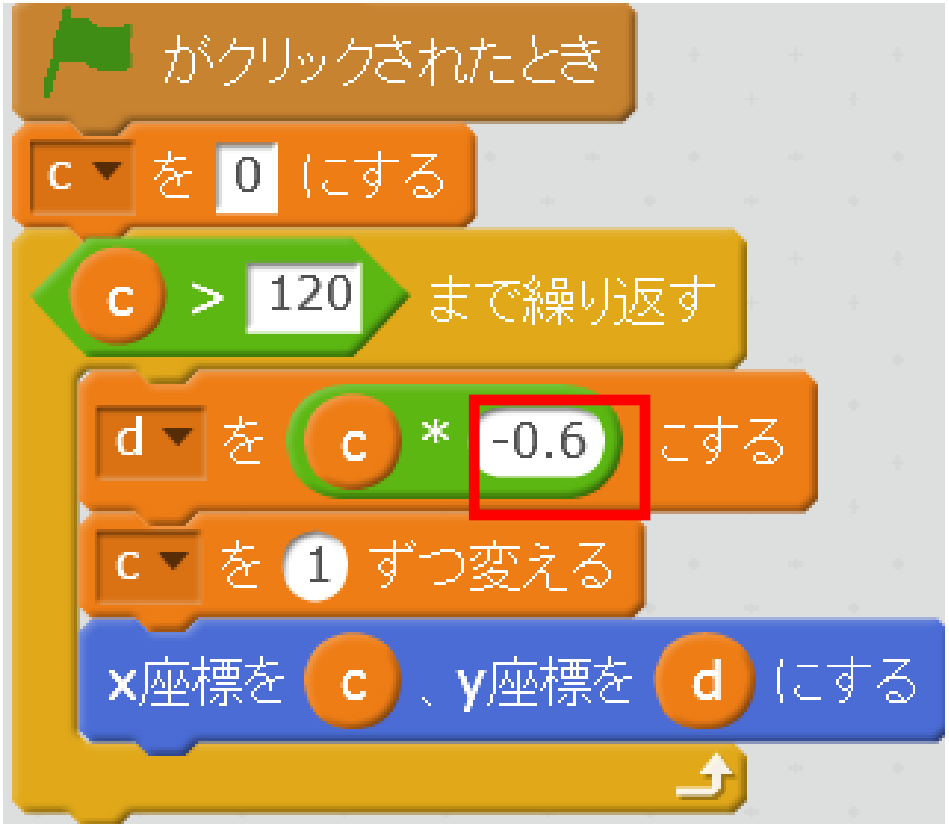
- ブロックを下のように組み立てる

- 実行ボタンを押すとキャラクタが動くことを確認

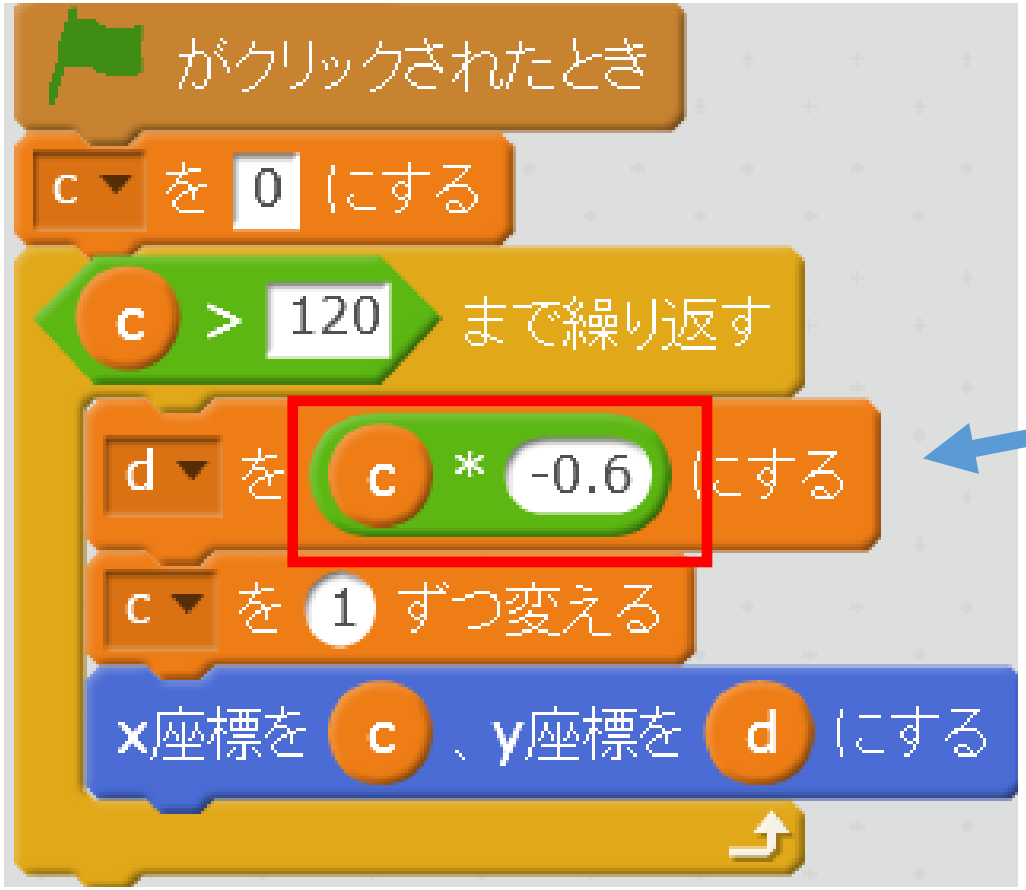


演習

- 「-0.6」のように書き換えて，もう１度実行してみなさい



まとめ



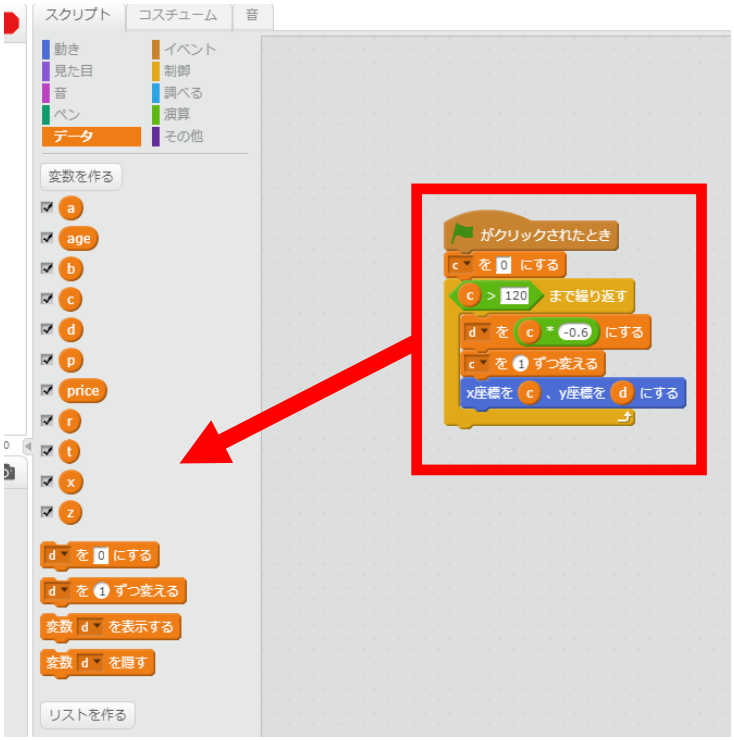
変数 c の値が変わっても，
いつも
　d の値は c の **-0.6** 倍にしたい

演習

- 今度は，次のブロックを使う

演習

- 今まで作成したブロックは不要なので，ブロックをマウスの右ボタンを押しながら，中央エリアにドラッグする．



ドラッグすると
消える

演習

次の２つの変数を作る

- xx
- vx

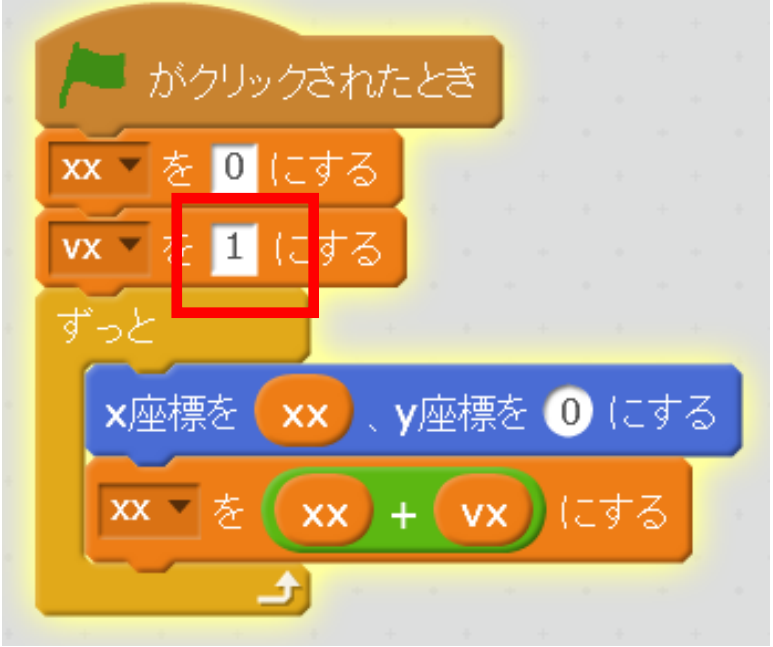
演習

「ねこ」のキャラクタを，

- 速さ： 1

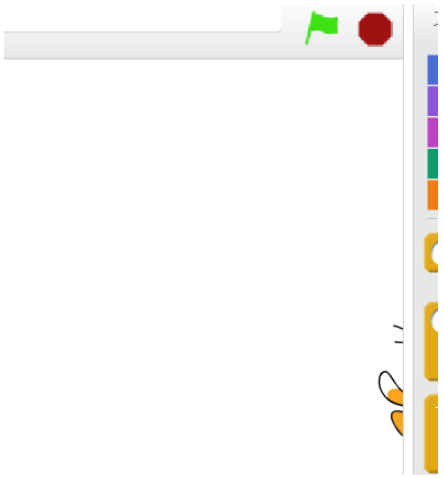
に設定して，動かしていく．

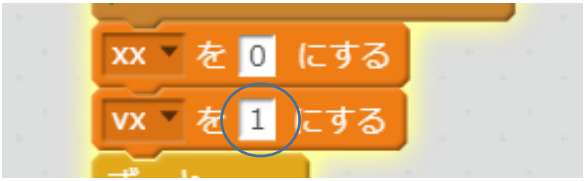
- 一度，右のようにブロックを組み立てなおしなさい



- 実行ボタンを押すとキャラクタが動くことを確認（キャラクタが右端まで行くと，プログラムが自動で止まる）

実行ボタン





「vx を **4** にする」、「vx を **0.1** にする」、「vx を **-1** にする」など、いろいろ変えてみなさい