1. 情報工学の世界

（コンピューターサイエンス）

金子邦彦

資料中の図などは Google Nano Banana 2 を用いて作成

「コンピューターサイエンス」第１回の内容

コンピュータの基礎と情報工学の全体像

無料ソフトウェア　著作権

オープンソース
ライセンス
改変の自由

所有権
利用のルール

情報工学と身近な技術の結びつき

情報工学

ゲーム開発

スマートフォンアプリ

Scratchによるビジュアルプログラミング

プログラミング的思考の学び

問題解決力

論理的思考

創造力

ゴール → 手順 → 解決

本授業の図は生成AIである Google Nano Banana 2 を用いて作成

自己紹介

金子邦彦 （かねこくにひこ）（福山大学工学部）

【研究領域】
データベース応用，データベース基盤技術，高度データ利用
【実績】
・学術論文等：２７編，査読付き国際会議：７６編，その他講演多数
・教科書等：３
・授業担当経験：のべ２４科目
・科学研究費：のべ１１件 概算のべ数千万円 他大学との共同多数
・共同研究，受託研究など：のべ１０件 概算のべ一億円 国際共同研究あり
・学部生，大学院生の指導経験多数

詳しくはhttp://www.kkaneko.jp/index.html

人工知能，**画像処理**，**３次元コンピュータグラフィックス（VR含む）**，**Webシステム**，**知的システム**や**社会システム**の成功には，
データベースが必要という気持ちで進めている

1-1 コンピューターサイエンスで学ぶこと


EFIWON ƁIƁAA
Hokkaido
Aizuwakamatsu
Sendai
Tokyo
Osaka
Kumamoto
Kagoshima
UTokyo
かり
かり

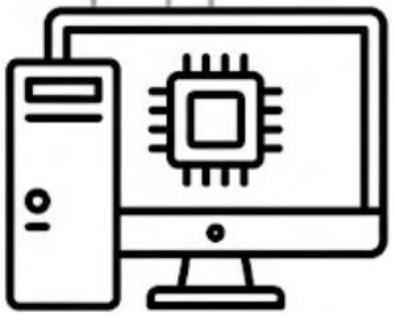
コンピュータと人間の協働

人工知能
知的な能力を持つ

ネットワーク化
生活・社会を変革する

プログラムで動作
作成は創造的な活動

協働
知的能力を増幅する



人間

コンピューター

コンピュータの基礎を学ぶことは、楽しくエキサイティング！

コンピューターサイエンスで学ぶこと①

コンピュータは万能ではない

計算には誤差が生じる

$0.1 + 0.2 \neq 0.3$

数値を有限桁でしか表せないため、計算結果に小さな誤差が積み重なることがある

バグを自分で全部は見つけられない

プログラムが正しいかを完全に自動で判定することは原理的にできない

コンピューターサイエンスで学ぶこと②

プログラミングの基礎

パソコンやスマートフォンのアプリも、すべてプログラムで動いている

- ブラウザ
- 地図アプリ
- ゲーム

コンピューターサイエンスで学ぶこと③

情報化社会から人工知能化社会へ

情報化社会の基盤

収集　処理　共有

コンピュータとネットワークにより、情報の収集・処理・共有が容易になった

情報が価値を持つ時代

情報そのものが価値を持つようになった

モノ・労働 ＋ 情報 → 新たな価値

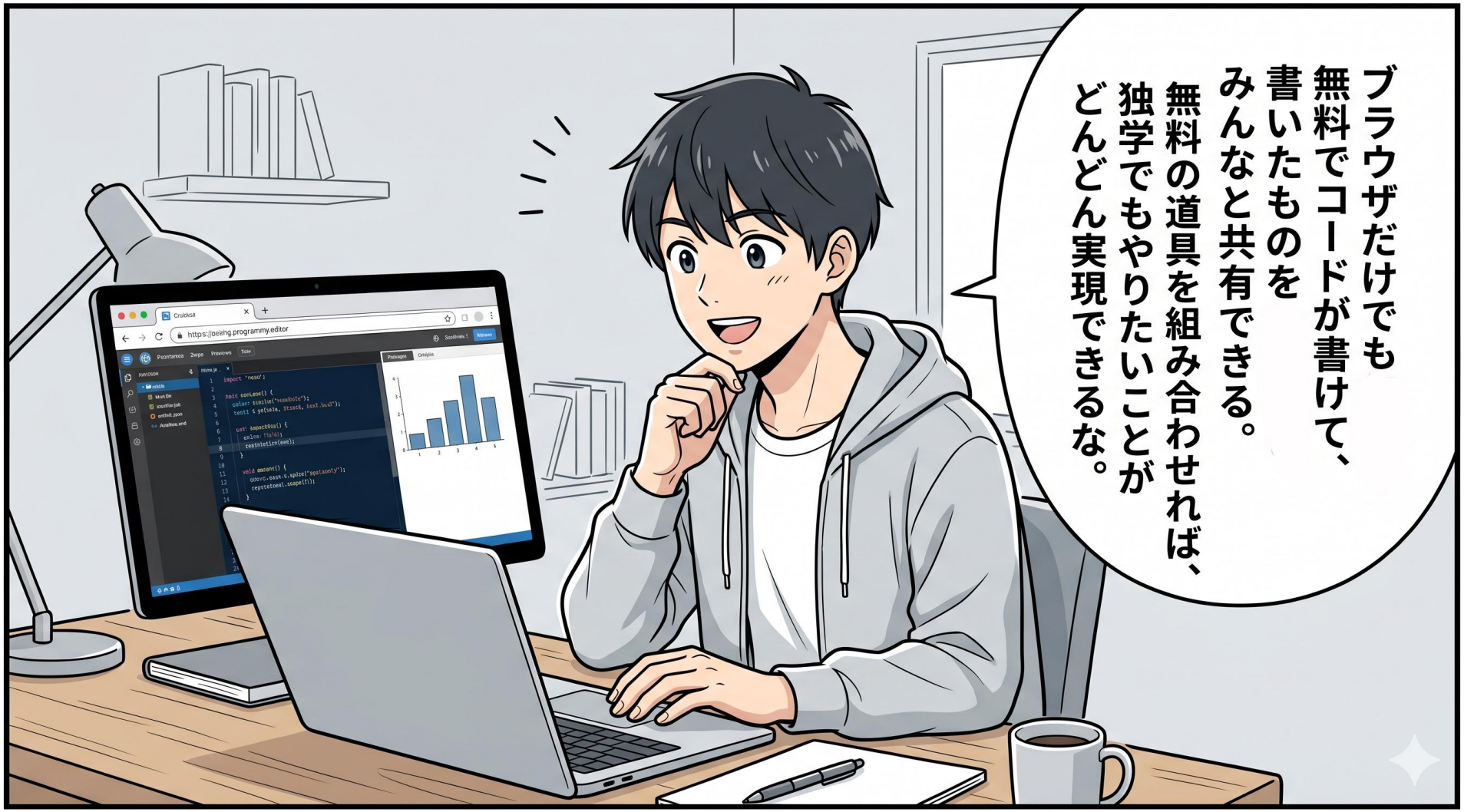
人工知能化社会へ

人工知能が身近に普及し、社会全体が大きく変化つつある

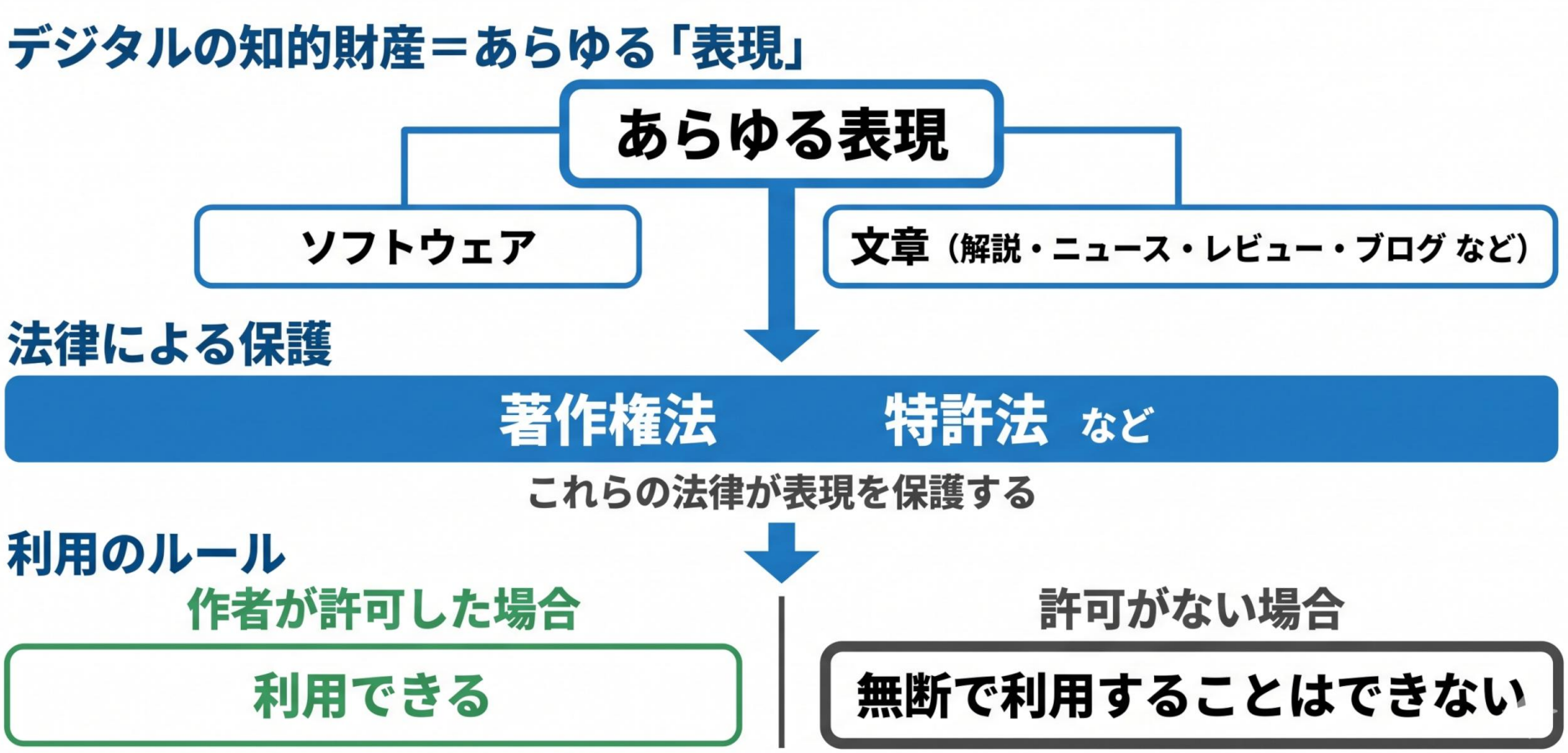
生活
仕事
社会

技術の進展により、社会の中心が『モノ』から『情報』、
さらに『知能』へと移り変わっている

1-2 無料ソフトウエア，無料データ


ブラウザだけでも
無料でコードが書けて、
書いたものを
みんなと共有できる。
無料の道具を組み合わせれば、
独学でもやりたいことが
どんどん実現できるな。

デジタルの知的財産と法律による保護



無料で使えるソフトウェアとデータ

Database Lab.

無料ソフトウェア（フリーウェア）

無償で配布されるソフトウェア

オープンソースソフトウェア（OSS）

ソースコードが公開され、
自由に利用・改変・再配布できる

※『自由（オープン）』とは利用・改変・再配布の自由のこと。必ずしも無料を意味しない。本資料では無料で使えるものを『無料ソフトウェア』と呼ぶ

やりたいことができる

社会が発展する

経験を共有できる

人類全体の発展に役立つ財産になる

無料ソフト・無料データを使うときのマナー

守るべき3つのマナー

1. 利用条件を確認する
2. 不具合は自己責任
   （そう定められていることが多い）
3. 著作権を尊重する

見極めの視点

注意すべきもの

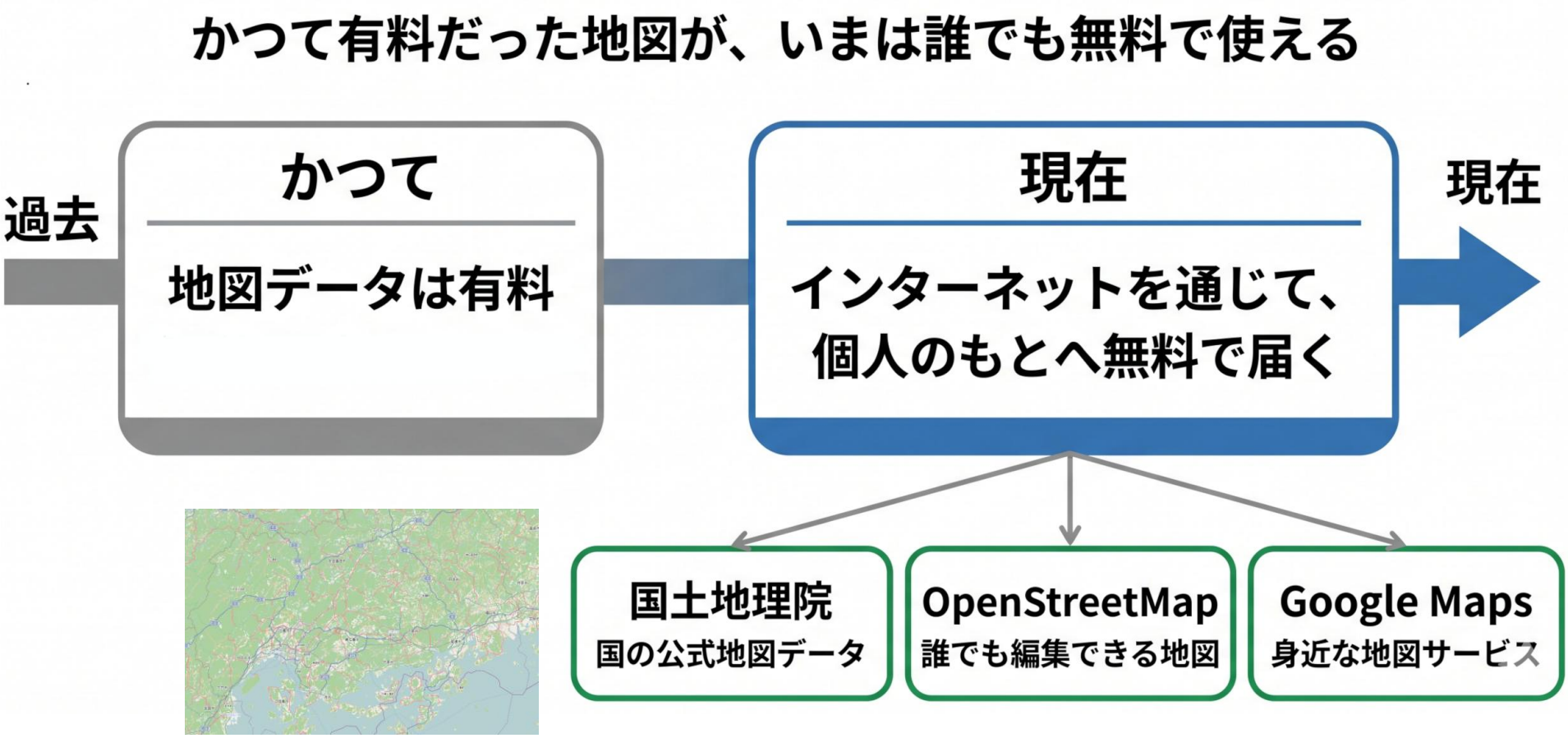
＝「無料を装った詐欺的なソフト・データ
（役に立たない／悪意で誘導する）」

問題ないもの

＝「広告収入や善意で提供されるもの」

大切なのは、提供元と利用条件を必ず確認すること

無料で使える地図データ



無料の地図（広島県部分を抜粋）

1-3 情報工学の世界
― ゲームの中の情報工学

情報工学科の学び

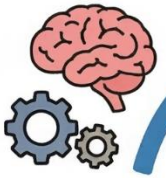
学ぶこと

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

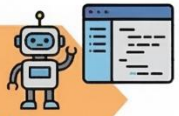
個人ワーク
（一人で取り組む）

グループワーク
（仲間と助け合う）

情報工学の面白さ

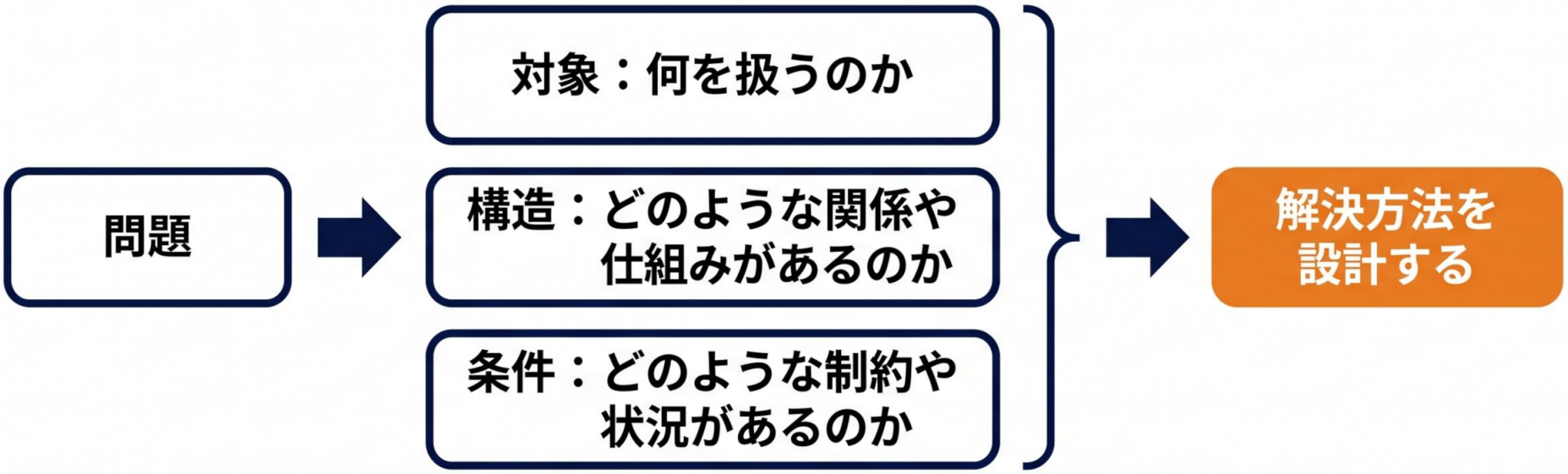
創造
（新しいアイデア）

設計
（システム全体）



具現化
（形にする・実装）

情報工学：問題を解決するための「見極める力」



例：キャラクタの自動の動き

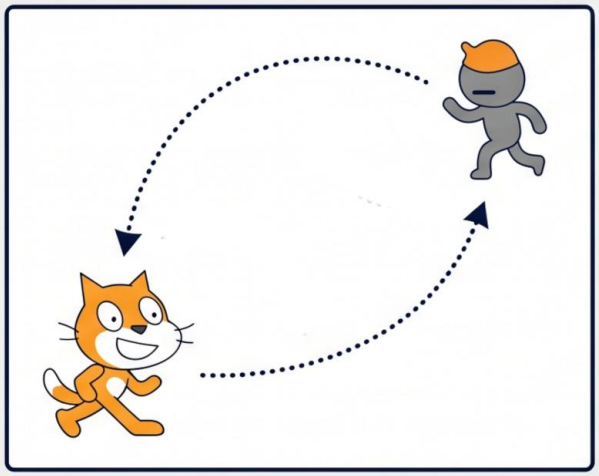
対象 → キャラクタ
構造 → 位置と速度の関係、移動のルール
条件 → 画面の範囲、障害物、速度の上限

やみくもにプログラムを書くのではなく、まず対象・構造・条件を整理してから設計する。
これが情報工学で求められる「見極める力」。

ゲームにおけるキャラクタの自動行動

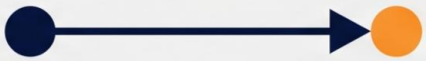
キャラクタはプログラムで動く ― 位置・速度・ルールの組み合わせ

対象：キャラクタ



構造：位置と速度の関係

次の位置 ＝ 現在の位置 ＋ 速度

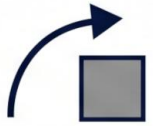


移動ルール
1. キャラクタの速度を計算
2. 速度に応じて位置を更新

条件

画面の範囲
枠からは
み出さない

障害物
ぶつからない
よう避ける

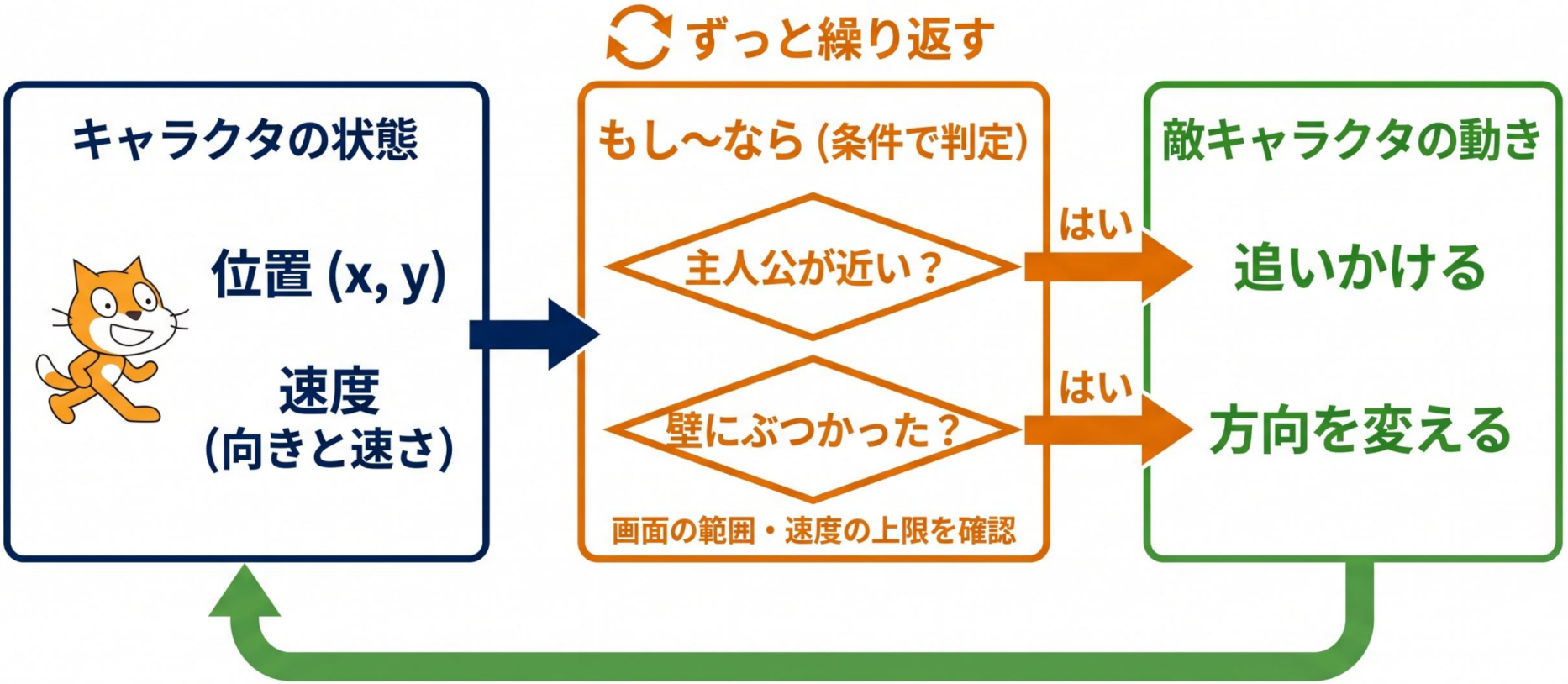


速度の上限
速くなり
すぎない

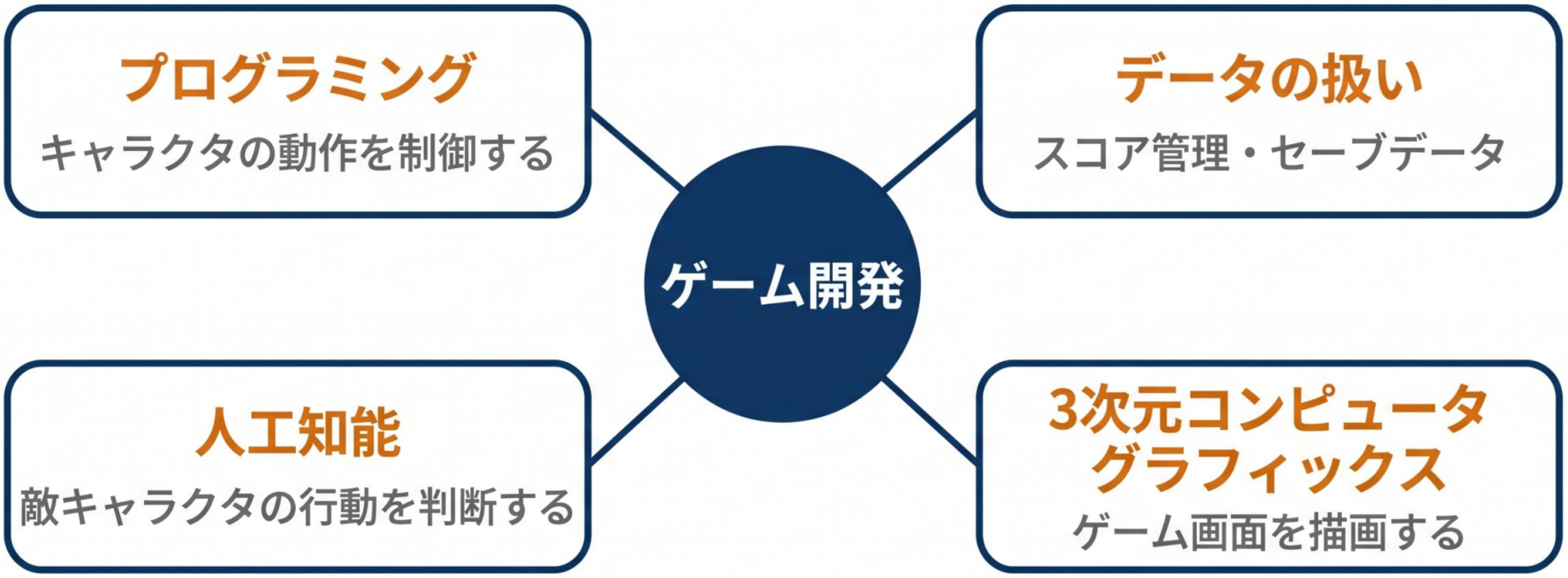
単純なルールの繰り返しで、追いかける・避けるといった自然な動きが生まれる

ゲームのキャラクタを動かす仕組みには、情報工学の考え方(問題を見極めて手順に分解する)が凝縮されているため、入門の題材として扱う

キャラクタの動きを制御する仕組み



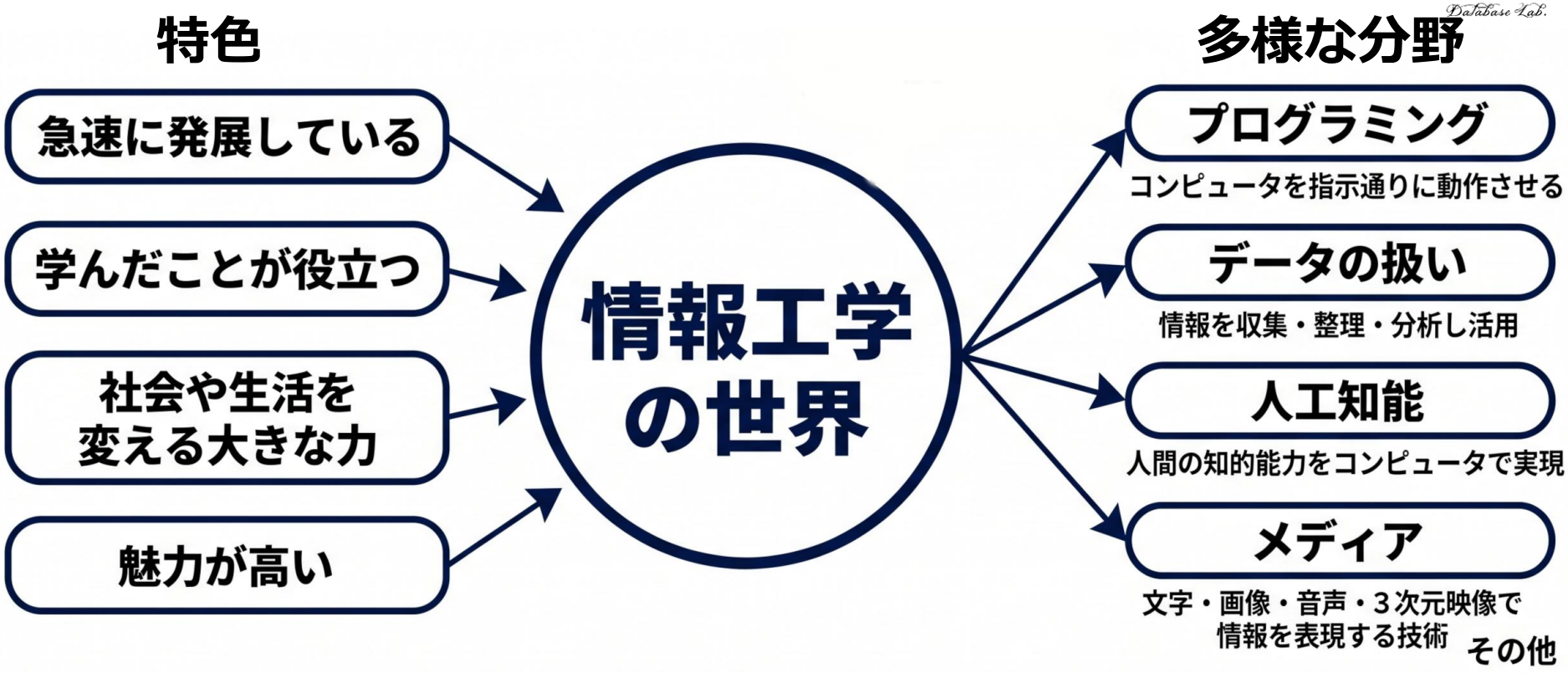
ゲーム開発を支える情報工学の分野



ゲームは情報工学のさまざまな分野が組み合わさって成り立っている

プログラミングの第一歩：Scratchでキャラクタの動きをプログラムで制御してみよう

情報工学の世界



重視する力　創造力 ／ 発想力 ／ デザイン力 ／ 実行力

新しいアイデア・アプローチが歓迎される
問題解決には、対象・構造・条件を見極める力が必要

1-4 プログラミングの創造性

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

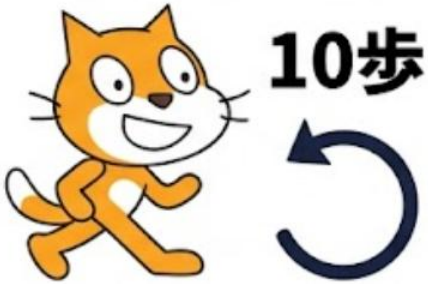
＋

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**



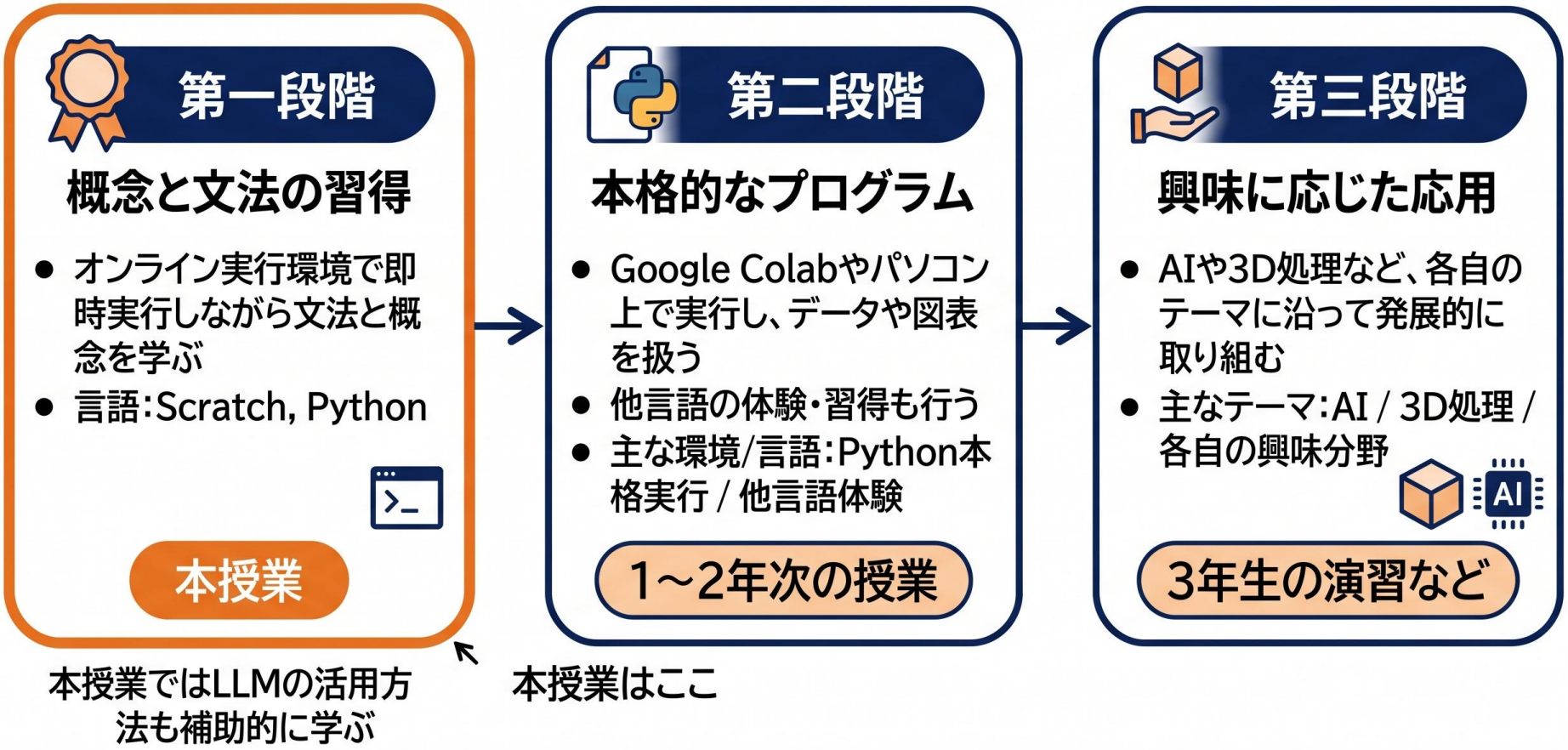
くりかえし

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

本授業と今後の学び


第一段階
概念と文法の習得
● オンライン実行環境で即時実行しながら文法と概念を学ぶ
● 言語：Scratch, Python
本授業
第二段階
本格的なプログラム
● Google Colabやパソコン上で実行し、データや図表を扱う
● 他言語の体験・習得も行う
● 主な環境/言語：Python本格実行 / 他言語体験
1～2年次の授業
第三段階
興味に応じた応用
● AIや3D処理など、各自のテーマに沿って発展的に取り組む
● 主なテーマ：AI / 3D処理 / 各自の興味分野
AI
3年生の演習など
本授業ではLLMの活用方法も補助的に学ぶ
本授業はここ

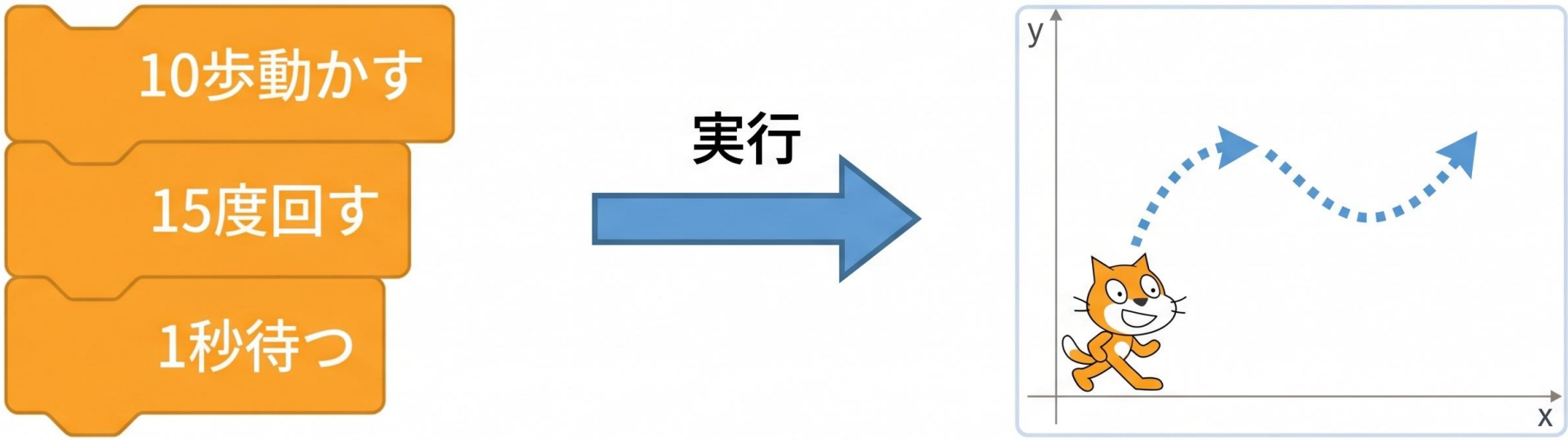
1-5 Scratch プログラミング

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

イベント駆動

何かが起きたときにプログラムが開始する仕組み

イベント（きっかけ）

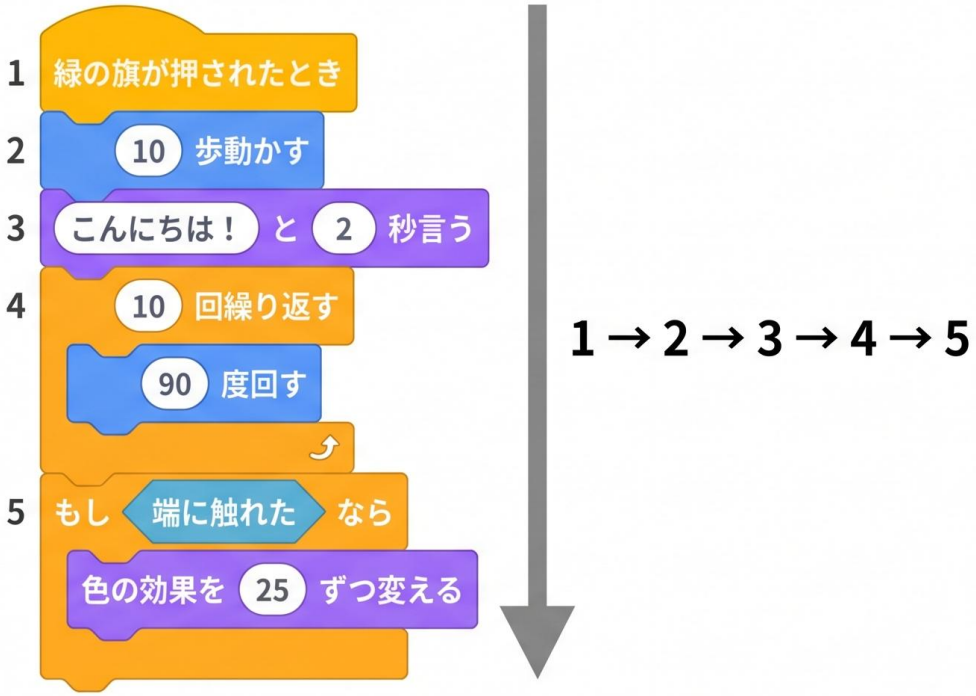
- 旗ボタンをクリック
  - 最も基本的なイベント
- キーが押される
- スプライトがクリックされる

イベントによるプログラム開始

Scratchでは『イベント』カテゴリの『旗が押されたとき』ブロックを使い、旗ボタンのクリックを合図に開始できる。

順次実行

上から下へ順番に実行される

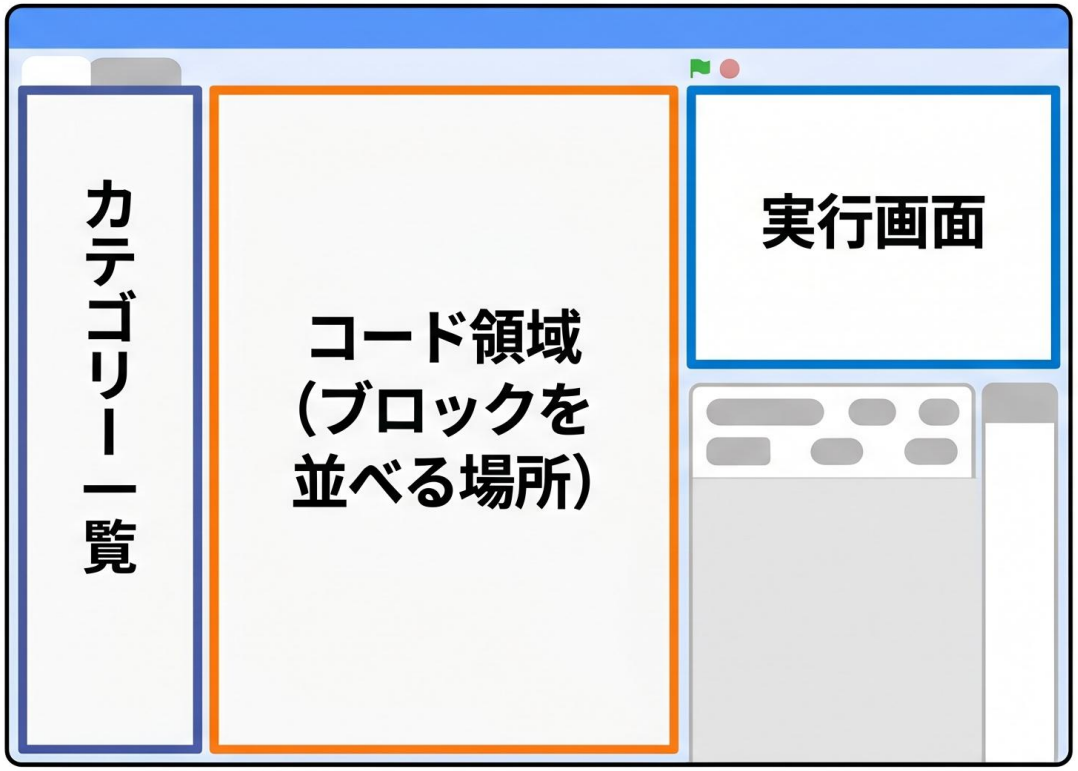


ブロックを上から下へ合体させると、合体した順番どおりにキャラクターが動く。
ブロックの並び順が、そのまま実行の順序になる。これが順次実行の基本。

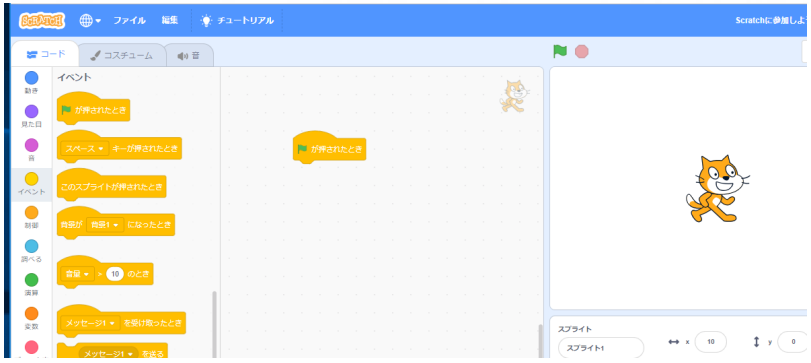
※ この例には後で学ぶ『繰り返し』『条件分岐』も含まれるが、ここでは上から下へ順番に実行される点に注目する。

Scratchの画面構成と操作の基本

画面構成

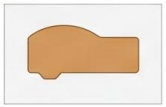


実際の画面



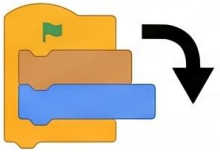
ブロック操作の基本

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

ブロック操作の基本

Scratch はブロックを並べて組み合わせることでプログラミングを行う

① ブロックを置く

使いたいブロックを画面に置く

② ブロックを組み合わせる

上から下へ順番につなげる

③ プログラムの起動

ボタンを押して動かす

組んだ順に実行される

ブロック操作の基本：① ブロックを置く

左側のカテゴリー一覧から**種類を選ぶ**

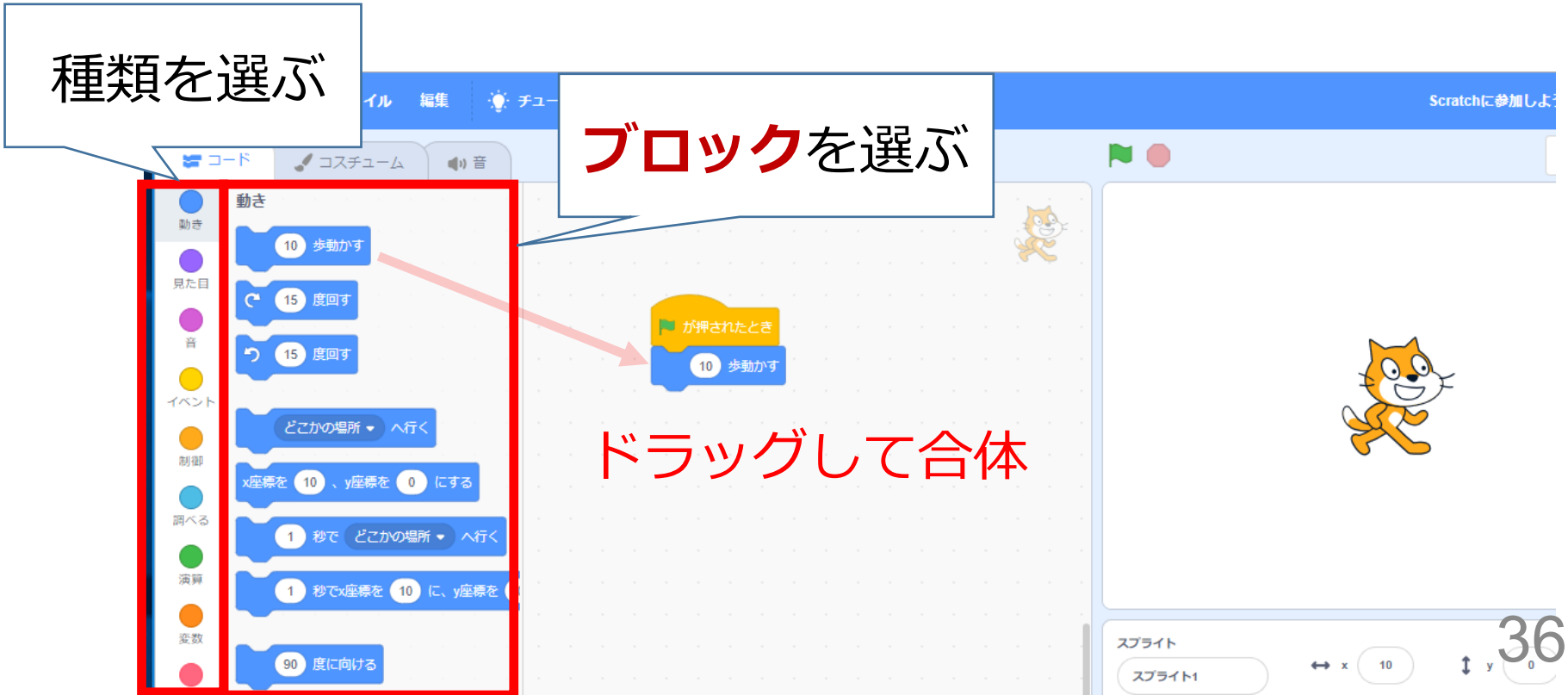
→ **ブロックを選ぶ**

→ 中央のコード領域へ**ドラッグして置く.**

ブロック操作の基本：② ブロックを組み合わせる

種類を選ぶ → ブロックを選ぶ

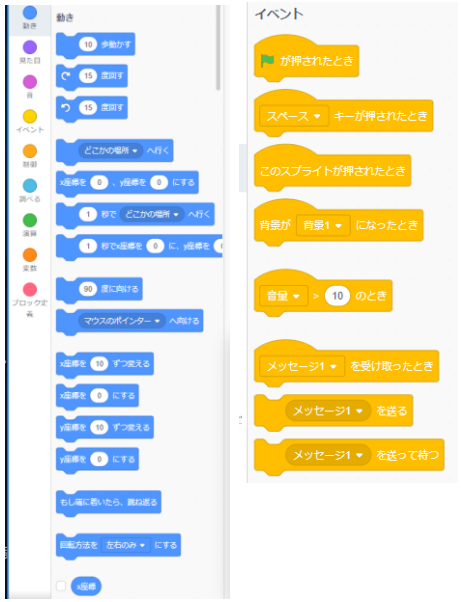
→ 既存のブロックの近くへドラッグして**合体させる**．ブロック同士がぴったり接続されること．



ブロックの形状と合体ルール

- Scratchのブロックには形状の違いがあり，**同じ形状同士で合体（接続）できる仕組み**になっている．

（例）例えば，**六角形のブロック**（条件判定ブロック）は**六角形の穴にはめ込んで使う**．形が合わないブロックは合体できない．



たくさんの種類のブロック

六角形ブロック

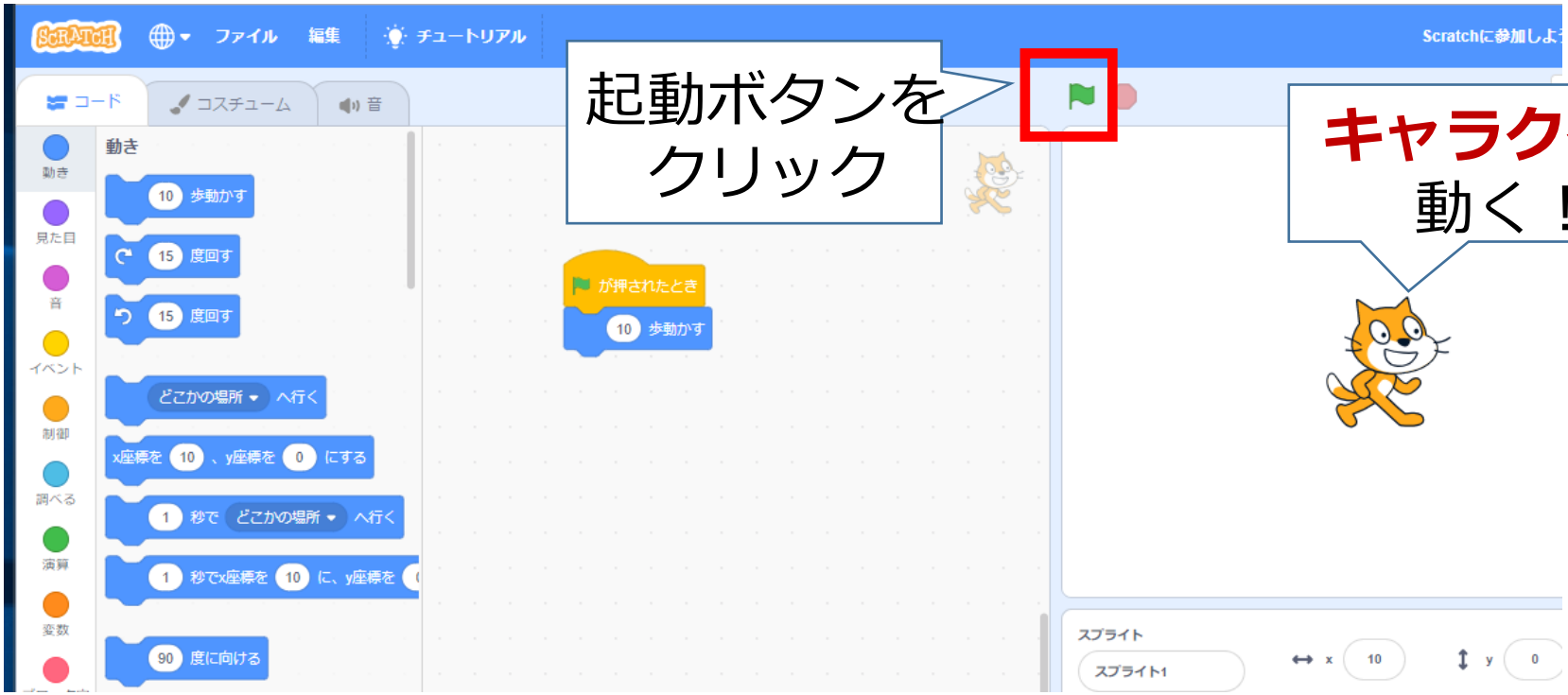
六角形の穴

同じ形のブロックは合体できる

ブロック操作の基本：③ プログラムの起動

起動ボタン（旗ボタン）をクリック

→ キャラクタが動く.



ブロックの削除

- 不要なブロックは，**右クリックメニュー**で「**ブロックを削除**」．
- ブロックを誤って配置しても，右クリックメニューからいつでも**削除できる**ので，自由に試行錯誤してよい．

Scratchの良さ

Scratch はブロックを並べて組み合わせることでプログラミングを行う

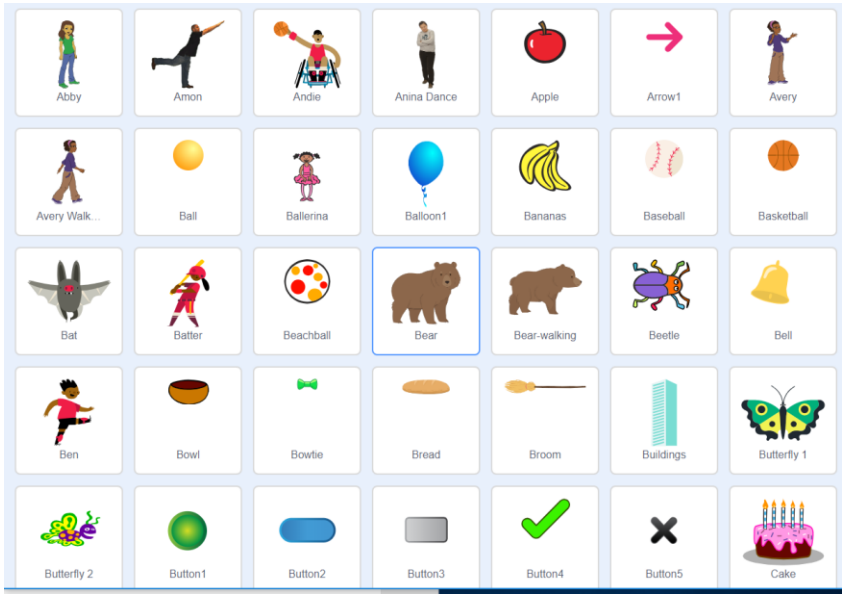
1-6 Scratchのキャラクタ

Scratch のキャラクタの基本

- キャラクタは自由に増やせる
- 各キャラクタは独立したプログラムを持つ
- 新しく追加したキャラクタには最初プログラムがない → 自分で組み立てる

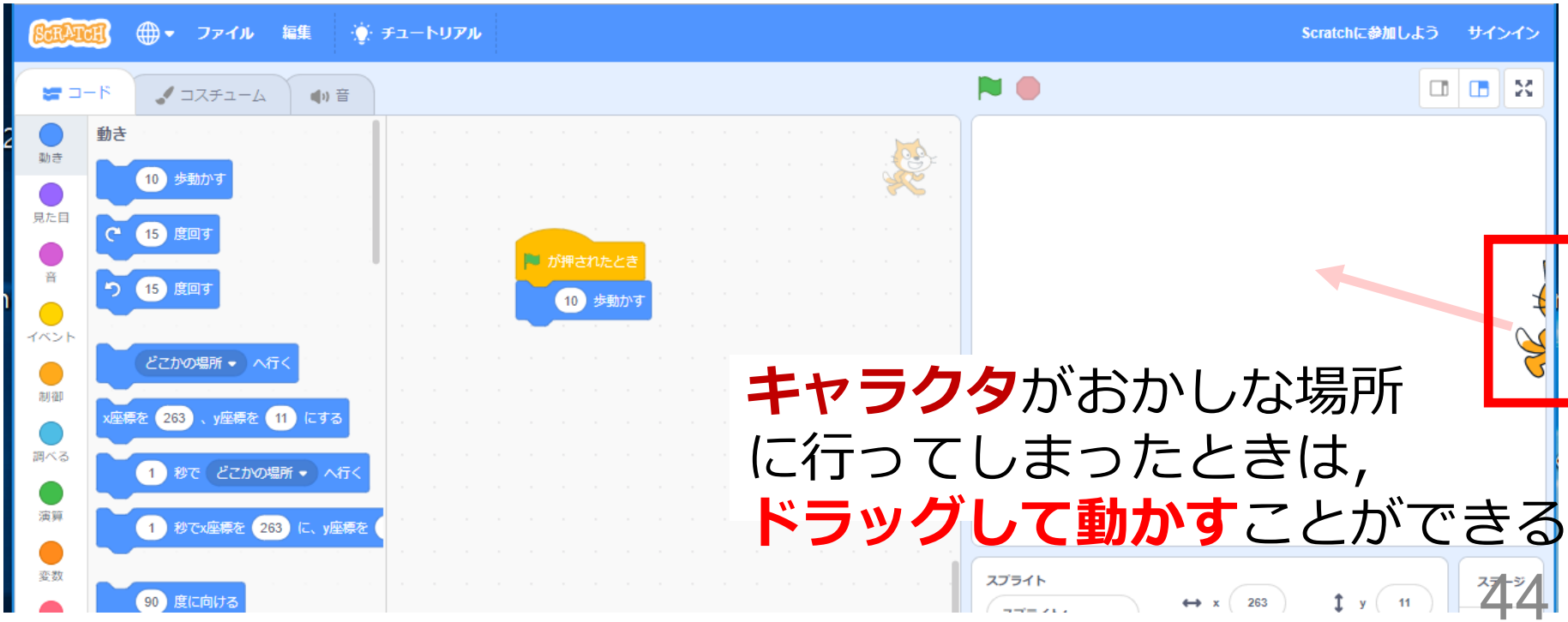
スプライト（キャラクタ画像）

- **スプライト**は，**キャラクタの画像データ**のこと．
- 画面右下の「**スプライトを選ぶ」ボタン**をクリックすると，多数のスプライトの一覧が表示され，そこから**好きなキャラクタを選んで追加**できる



キャラクタの強制移動

- **キャラクタ**はマウスでドラッグ（左ボタンを押しながら）して**位置を変更できる．プログラム実行中でもOK.**
- **キャラクタ**がおかしな場所に行ってしまったときに有効



キャラクタがおかしな場所に行ってしまったときは，**ドラッグして動かす**ことができる

キャラクタの削除

キャラクタの削除：スプライト一覧に表示される
キャラクタの右上の「x」をクリック

削除したいキャラクタ
の「x」で
「**削除**」

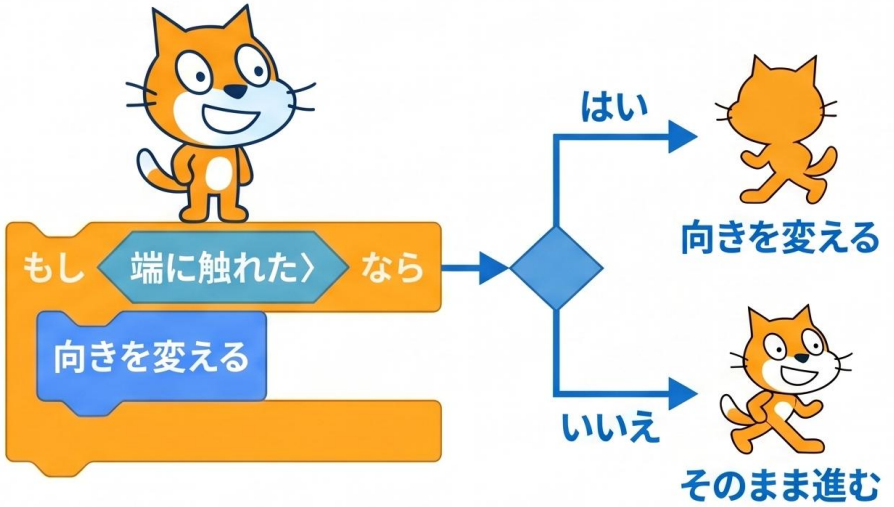
1-7 Scratchのキャラクタの制御

Scratch の「制御」ブロックの使い方

① 繰り返し

同じ命令を決めた回数だけ自動でくり返す

② もし…なら、…する（条件分岐）



条件が成り立つときだけ、決めた命令を実行する

制御 ① 繰り返し

「**制御**」カテゴリの**「ずっと」ブロック**を**合体**.

- **「ずっと」の中に入れたブロック**は，強制停止するまで**何度も繰り返し実行される**．キャラクタを自動的に動かし続けることができる．※止めるときは赤い丸ボタン
- 繰り返しの外に置いたブロックは，1回だけ実行される.

キャラクタが自動で
動き続けるようになる

制御 ② もし・・・たら，・・・する（条件分岐）

「**動き**」カテゴリの**「もし端に着いたら，跳ね返る」ブロックを合体**.

- **端に着いたら跳ね返るようになる**．「もし端に着いたら，跳ね返る」ブロックは，キャラクタがステージの端に到達したかどうかを判定.
- 「もし○○なら△△する」のような**条件に応じた動作の切り替え**が**条件分岐**.

動き

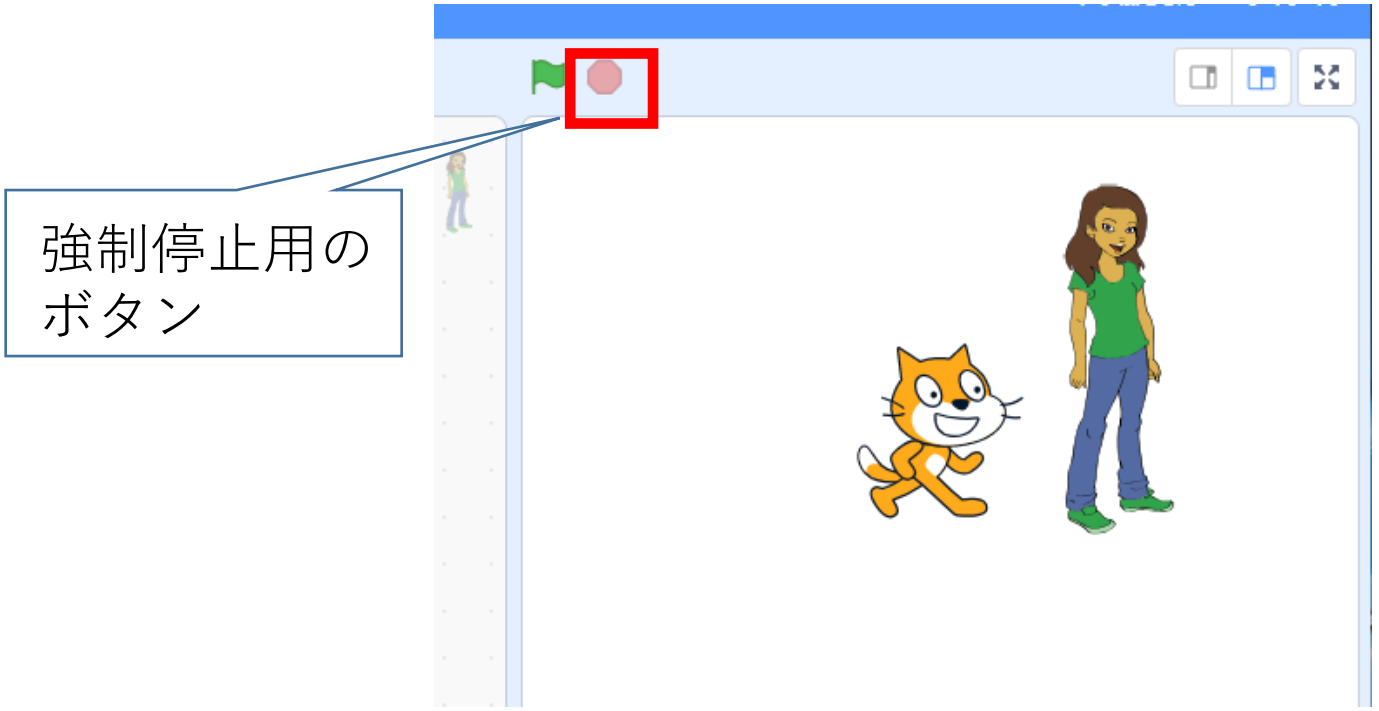
もし端に着いたら，跳ね返る

ドラッグ

端に着いたら
跳ね返るようになる

プログラムの強制停止

- 実行中のプログラムは，**強制停止用のボタン（赤い丸ボタン）**のクリックにより**停止**できる
- 繰り返しを使ったプログラムでは，この強制停止ボタンが停止手段となる．

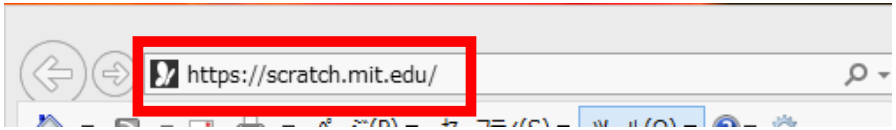


演習

演習① Scratch の開始・ブロック操作

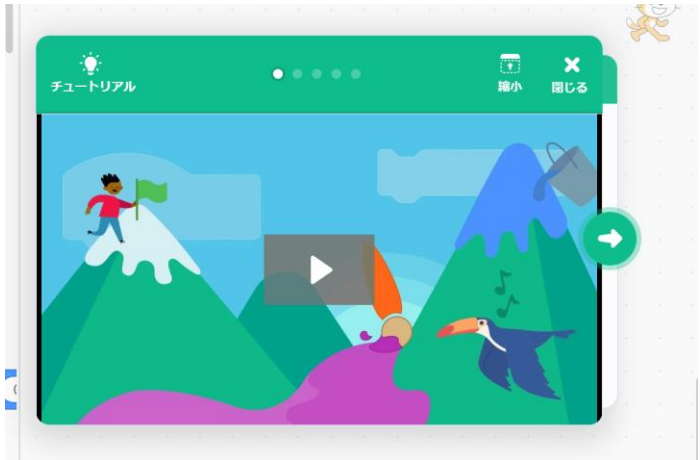
1. **Webブラウザを起動**
2. Webブラウザで，次のURLを開く

https://scratch.mit.edu/



3.「**作ってみよう**」をクリック

次の説明ビデオを視聴しないときは
右上の「×閉じる」をクリック



4.「**イベント**」をクリック

6.「**動き**」をクリック

5. 

を**ドラッグ**（左ボタンを押しながら移動し，左ボタンを離す）

7. 

を**ドラッグ**し 

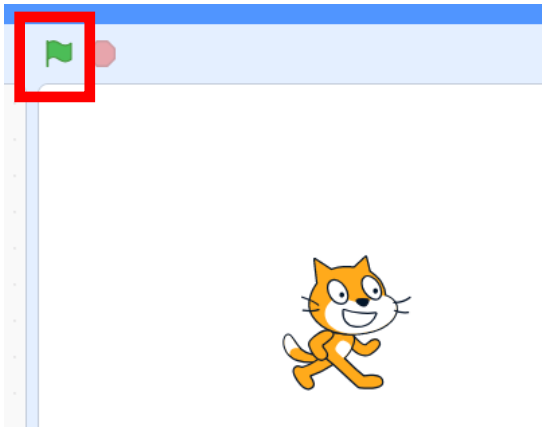
と合体

組み合わせて合体

イベント

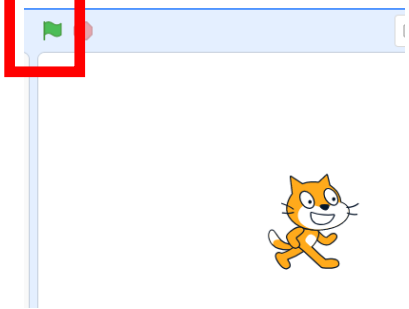
動き

8.  **（旗）ボタン**をクリックすると**キャラクタ**が，
少し右に，動く



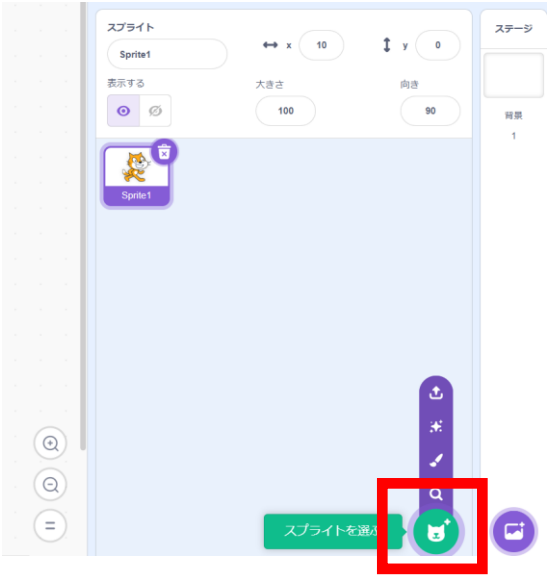
Scratch の『歩』は画面上の長さの単位

9.  **（旗）ボタン**を数回クリックしてみよう

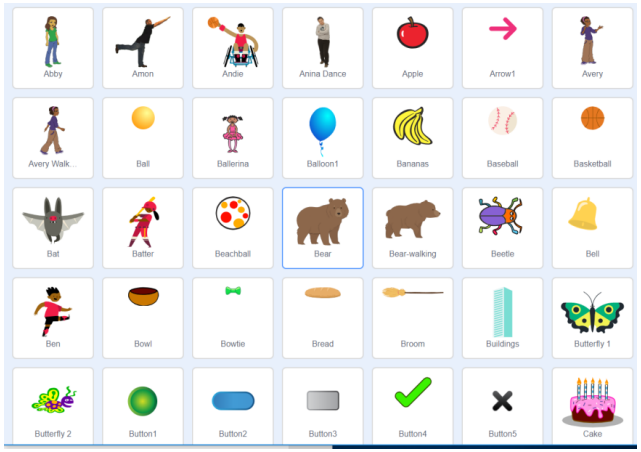


演習 ② キャラクタ・スプライト

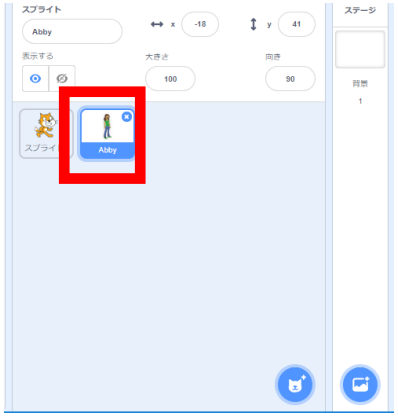
1. 「**スプライトを選ぶ**」をクリック



2. 好きな**キャラクタ**を選ぶ



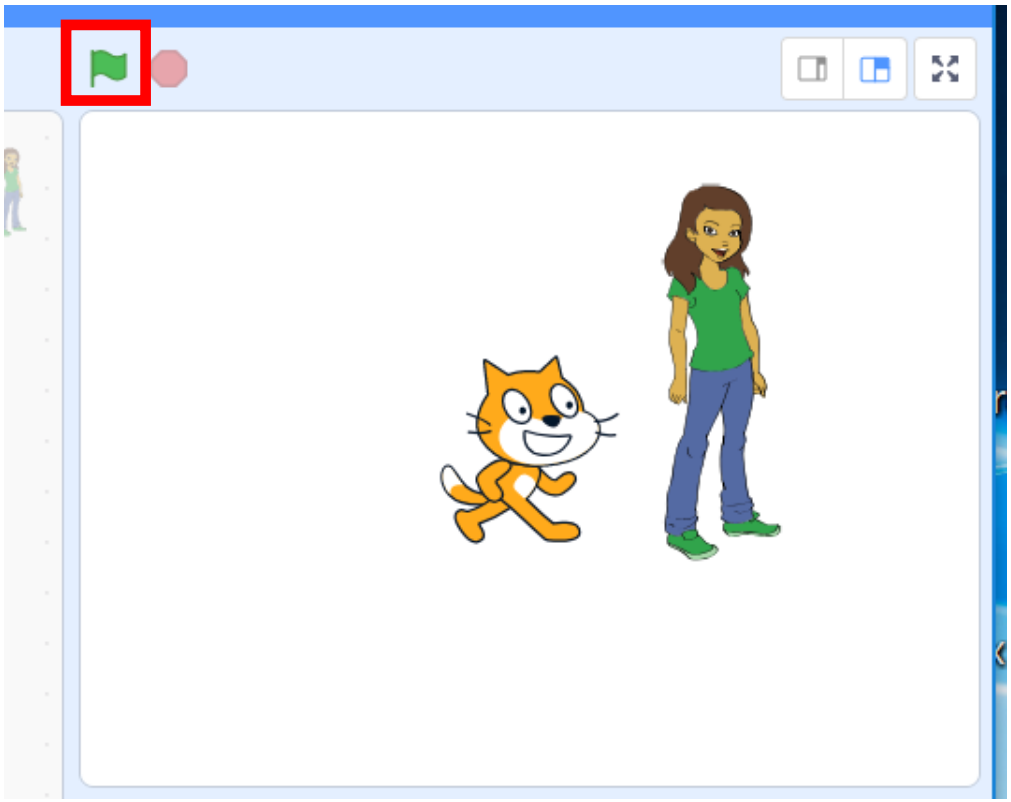
3. 右下の「スプライト」で，新しいキャラクタを選んでから.



4. 演習①と同様に「**旗が押されたとき**」と「**10歩動かす**」のブロックを組み立てる.

Scratch の『歩』は画面上の長さの単位

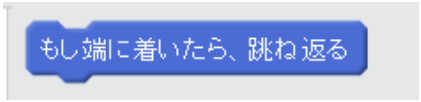
5.  **（旗）ボタン**をクリックするとキャラクタが動く．何度かクリックしてみよう



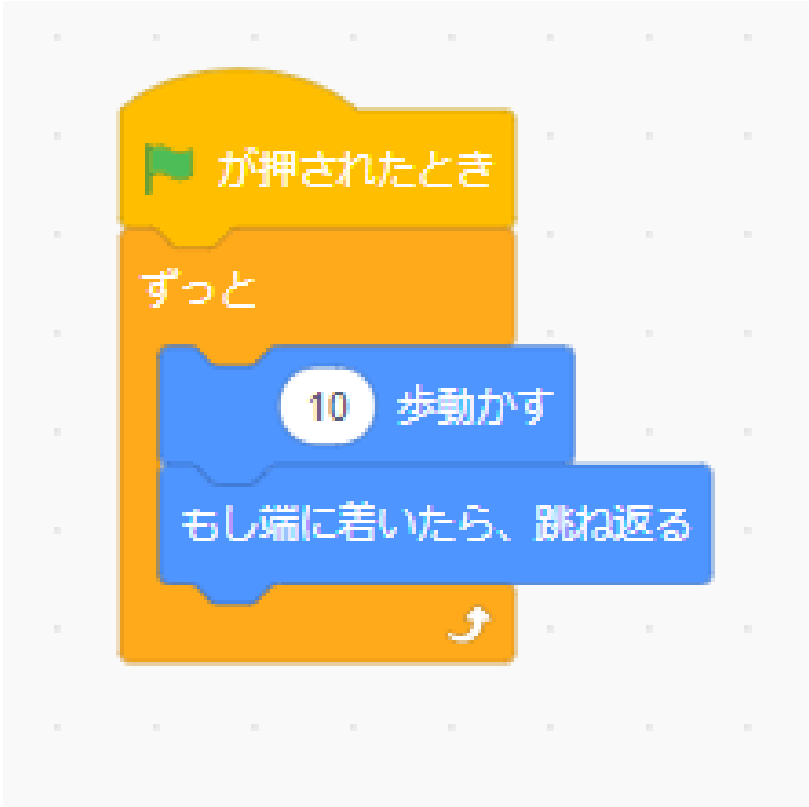
演習③ キャラクタの制御

1．　新しいキャラクタが選ばれていることを確認

2．　「**制御**」を選び  **（ずっと）のブロックをドラッグ，既存のブロックと合体**

3.「**動き**」を選び 

（もし端に着いたら，跳ね返る）ブロックをドラッグ，「**ずっと**」の中に合体



4.  **（旗）ボタン**をクリックすると**キャラクタ**が**動く**.

 **（強制停止）ボタン**をクリックすると**止まる**

5.（**強制停止**）**ボタン**をクリックして**止めてから**

「**15度回す**」を「**ずっと**」の前に**加える.**

動き始めるときに，**ななめに傾く**ようになる

回転は時計回り

6.  **（旗）ボタン**をクリックすると**キャラクタ**が**動く**.

 **（強制停止）ボタン**をクリックすると**止まる**

動き始めの瞬間に
１５度傾く

演習④　自由制作（余裕のある人向け）

各自で工夫する．

- いろいろな動きを試す．
- 複数のキャラクタを同時に動かす．
- たくさんの種類のブロックを試す．同じ形のブロックは合体できる（六角形ブロックは六角形の穴にはめ込む）．