

# pd-9. プログラミングの基礎、 AIシステムの基礎

(Python による ICT システム)

URL: <https://www.kkaneko.jp/de/pd/index.html>

金子邦彦



1. Pythonを使った人工知能プログラミング
2. プログラミングの楽しさ、Pythonの基礎、ChatGPTの活用法、AIプログラムの例
3. 実践的なコード例を交えて、プログラミングとAIの基礎を説明
4. プログラミング、AIへの興味を持ち、将来のITエンジニアとしてのキャリアをイメージしましょう

# アウトライン

1. 人工知能とプログラミングを学ぶ楽しさ
2. Python の基礎
3. 対話型AI（ChatGPT）の活用
4. AIのプログラム例



# 1. 人工知能とプログラミング を学ぶ楽しさ

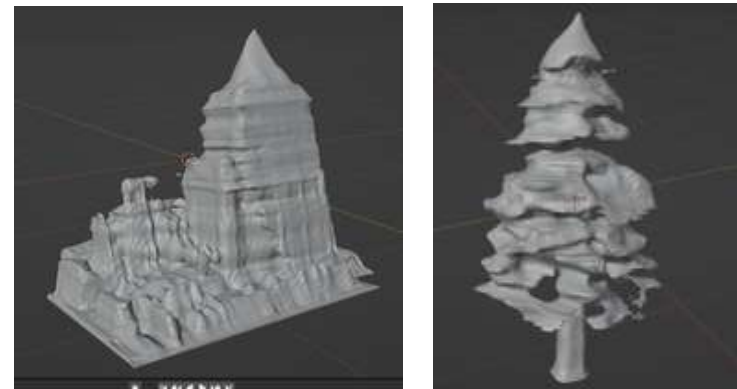
# 人工知能を学ぶ楽しさ



**人工知能**：人間の思考を模倣し、超えることを目指す挑戦。  
人工知能は、現在進行形の最先端技術であり、未来に向けてもその発展が続く、刺激と興奮に満ちた分野である。



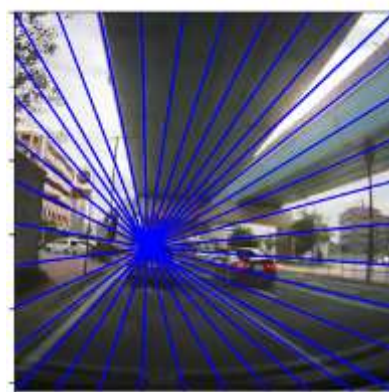
顔情報処理



3次元データの合成



物体検出、指定されたキーワードは  
AIには初見（ゼロショット）



消失点推定



テキスト検出

# プログラミングの楽しさと達成感



## ・ 楽しさ

- ・ **未来の技術を学ぶことは楽しい。**
- ・ プログラミングは**クリエイティブな行為**。
- ・ 視覚的なプログラムを書くことで、ゲーム感覚をもって楽しみながら学習することも可能。

## ・ 達成感

- ・ **自分のアイデアを形にすることで得られる達成感**
- ・ 自分でデザインし、問題が生じたときは自分で解決していく。
- ・ **自分の手でプログラムを完成させるプロセスは、大いに充実感をもたらすもの。**

- **プログラミング**は**人間の力を増幅**し、私たちができることを大幅に広げる
- **シミュレーション、大量データ処理、AI連携、ITシステム制作**など、さまざまな活動で、**プログラミング**は役立つ
- **プログラミングはクリエイティブな行為**
- さまざまな**作業を自動化**したいとき、**問題解決**したいときにも役立つ





## 2. Python の基礎

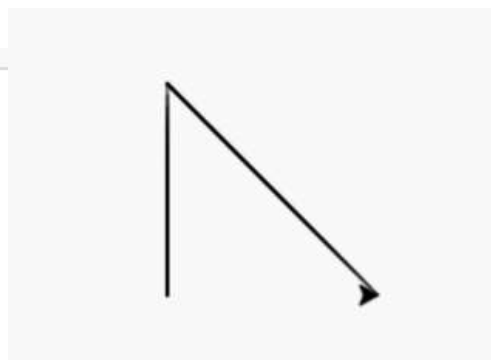
ビジュアルに結果を得るプログラムで、まずは楽しもう！



## ① Python でのグラフィックス

**プログラミングのクリエイティブな側面を実感し、ITエンジニアとしての自信を深める**

```
1 import turtle
2 t = turtle.Turtle()
3 t.goto(0,100)
4 t.goto(100,0)
```



視覚的な結果を得る  
プログラム

## ② 人工知能による画像認識

**データから新しい価値を創造できるAIエンジニアに興味を深め**



画像が犬なのか、  
猫なのか  
を判別するAI を動作

# オブジェクトとメソッド



- **オブジェクト** : コンピュータでの**操作や処理の対象となるもの**

**hero.moveDown()**

**hero**      **オブジェクト**  
**moveDown()**   **メソッド**  
間を「.」で**区切っている**

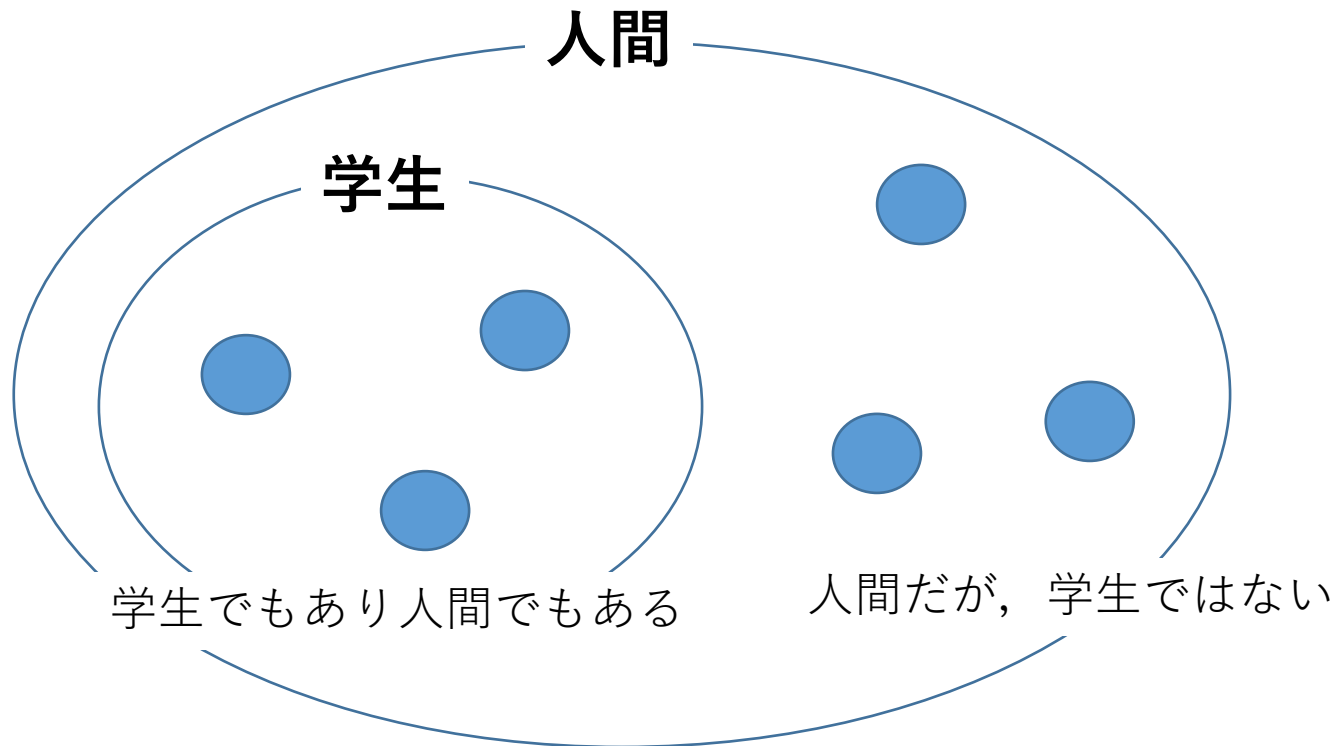
- **メソッド**: **オブジェクト**に属する機能や操作. オブジェクトがもつ能力に相当する
- **引数** : **メソッド**が行う操作の詳細に関する情報, **メソッド**呼び出しのときに、引数を指定できる

**hero.attack("fence", 36, 26)**

# クラスとオブジェクト



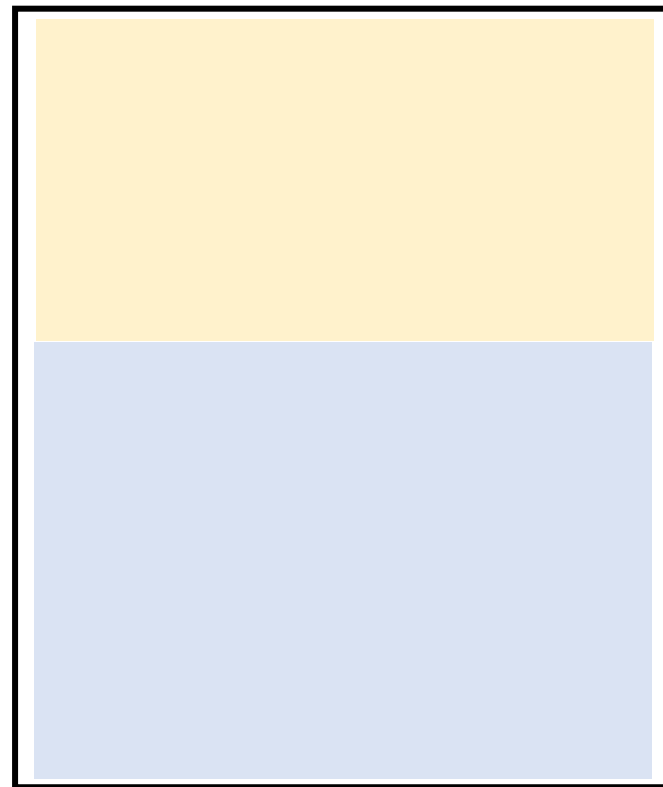
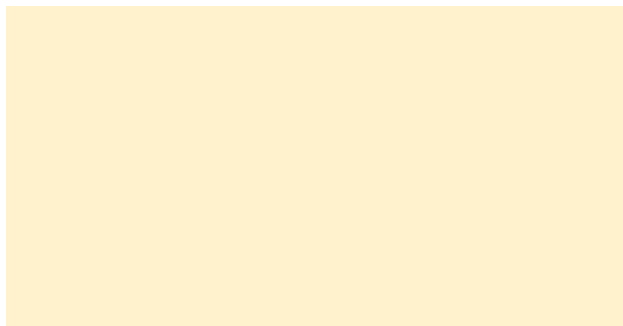
クラスは、同じ種類のオブジェクトの集まりと考えることができる



# インポート



インポートにより  
ファイルを丸ごと取り込む



**Python プログラムのファイル**  
(関数、変数、クラスなどの  
オブジェクトについて記載)

**Python プログラムのファイル**

# モジュールのインポートの方法



単一の Python プログラム  
ファイル = モジュール

```
def foo(x):  
    return x + 3
```

インポート



拡張子「.py」のファイル名で  
ファイルに保存

(ファイル名を **bar.py** とする)

ファイル名は何でもよい

import bar でインポート

```
import bar  
print(bar.foo(100))
```

bar.foo は bar モジュール内  
の foo オブジェクトの意味

Python プログラムのファイル

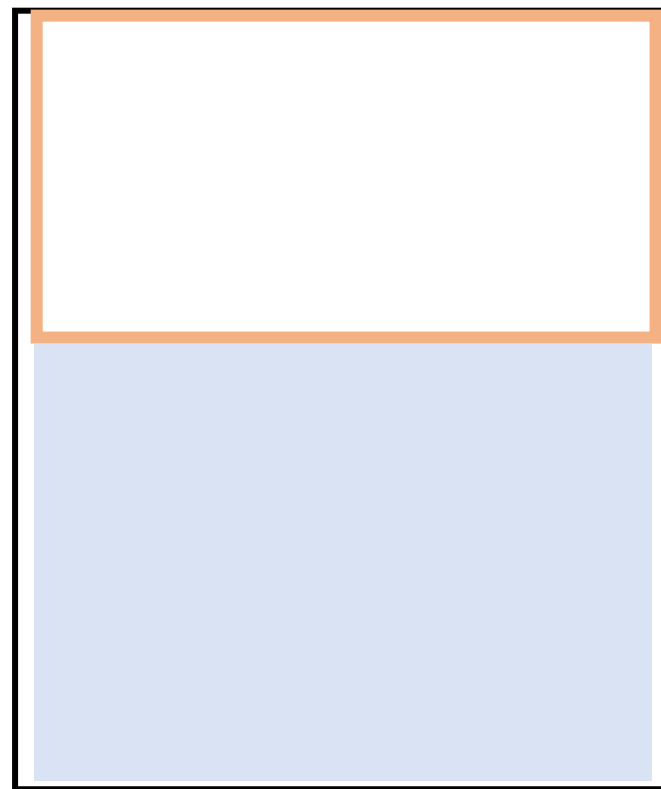
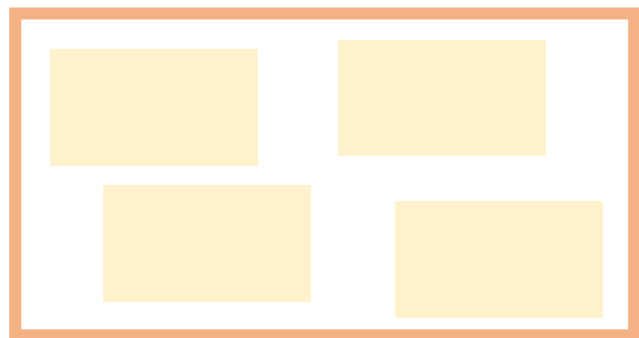
実行結果として  
103 が表示される

```
>>> import bar  
>>> print(bar.foo(100))  
103
```

# パッケージのインポート



インポートにより、**パッケージのファイル**を丸ごと取り込む



複数の Python プログラム  
を**パッケージ**にすることも  
可能

Python プログラムのファイル



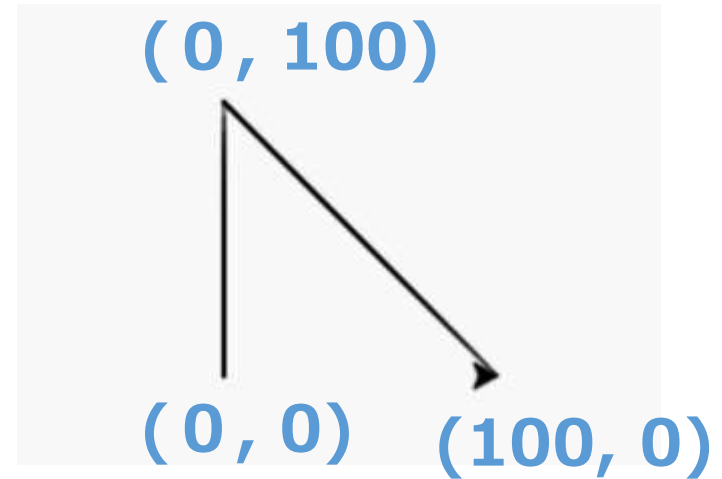
# タートルグラフィックス



カーソルを使って絵を描く

```
1 import turtle
2 t = turtle.Turtle()
3 t.goto(0,100)
4 t.goto(100,0)
```

タートルグラフィックスの機能をインポートする「import turtle」が必要



# タートルグラフィックス



```
1 import turtle
2 t = turtle.Turtle()
3 t.goto(0,100)
4 t.goto(100,0)
```

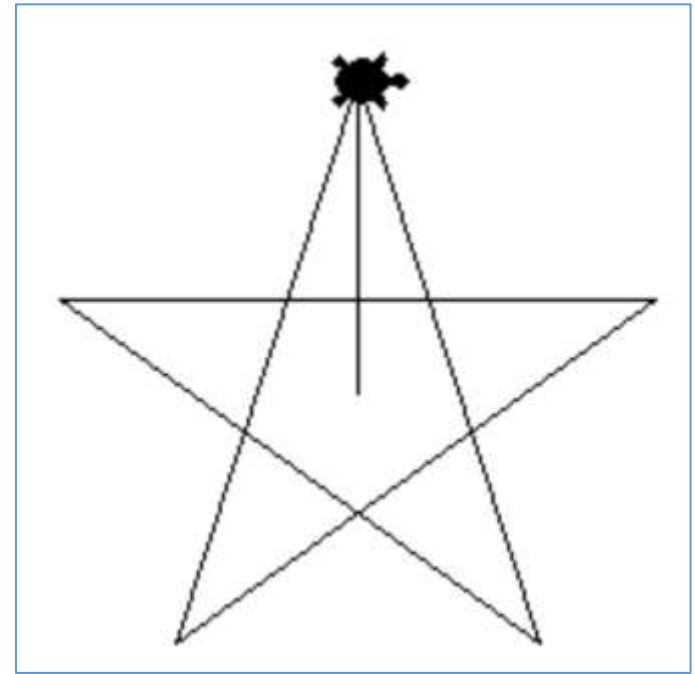
オブジェクト    メソッド

- **メソッド**は、オブジェクトが持つ機能を呼び出すためのもの
- 「**goto**」は指定した座標への移動

## 主なメソッド

- |                                     |               |
|-------------------------------------|---------------|
| • <b>goto</b> ( <横方向の値> , <縦方向の値> ) | <b>移動</b>     |
| • <b>forward</b> ( <移動量> )          | <b>前進</b>     |
| • <b>backward</b> ( <移動量> )         | <b>後退</b>     |
| • <b>right</b> ( <角度> )             | <b>右回りに回転</b> |
| • <b>left</b> ( <角度> )              | <b>左回りに回転</b> |

```
1 import turtle
2 t = turtle.Turtle()
3 t.goto(0, 100)
4 t.goto(58, -80)
5 t.goto(-95, 30)
6 t.goto(95, 30)
7 t.goto(-58, -80)
8 t.goto(0, 100)
```



Python プログラム

実行結果

**基本を押さえる**（パッケージ、オブジェクト生成、メソッド） ことで、プログラミングを楽しめるようになるろう

# Python の基礎① モジュール



```
1 import turtle
2 t = turtle.Turtle()
3 t.goto(0, 100)
4 t.goto(58, -80)
5 t.goto(-95, 30)
6 t.goto(95, 30)
7 t.goto(-58, -80)
8 t.goto(0, 100)
```

- **モジュール**は、特定の機能を実現するための Python プログラムファイル
- 「**import turtle**」は**turtle モジュール**を現在のプログラムに**インポート**
- 「**turtle.Turtle()**」は**turtle モジュール内の Turtle** の機能を**呼び出し**

# Python の基礎② クラスとオブジェクト



```
1 import turtle
2 t = turtle.Turtle()
3 t.goto(0, 100)
4 t.goto(58, -80)
5 t.goto(-95, 30)
6 t.goto(95, 30)
7 t.goto(-58, -80)
8 t.goto(0, 100)
```

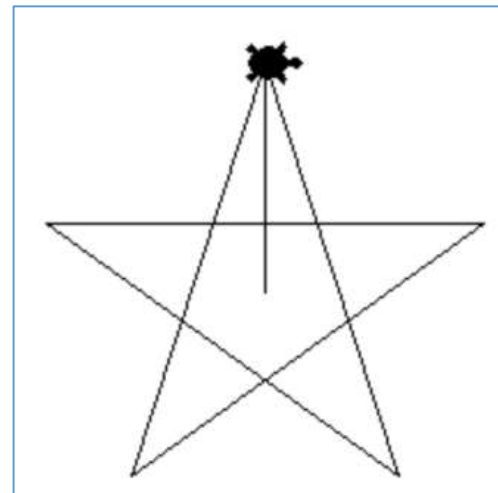
- 「**turtle.Turtle()**」では、turtleモジュールの**Turtle**クラスの**オブジェクト生成**
- **Turtle** クラスは、図形の描画や移動などの操作の**機能**を実現するためのクラス

# Python の基礎③ メソッド



```
1 import turtle
2 t = turtle.Turtle()
3 t.goto(0, 100)
4 t.goto(58, -80)
5 t.goto(-95, 30)
6 t.goto(95, 30)
7 t.goto(-58, -80)
8 t.goto(0, 100)
```

- **メソッド**は、オブジェクトが持つ機能呼び出すためのもの
- 「**goto**」は**指定した座標への移動**



実行結果



# 演習 プログラミングはクリエイティブ

## 【構成】

- ①オブジェクト生成
- ②移動
- ③色、円

## 【トピックス】

- ・ trinket でのプログラム実行
- ・ モジュールのインポート
- ・ オブジェクトの生成
- ・ メソッド（移動）



各自の自発的な演習、自己研鑽の時間

## ① Pythonでグラフィックスを描く

資料のプログラムを動かし理解を深める

## ② Pythonの基本を押さえる

オブジェクト、メソッド、引数

## ③ 発想力、創造力

turtleモジュールを使用して、あなた自身がデザインした図形を描く。

## ④ 自主性、自己研鑽力、自分なりに工夫したことを振り返る

説明されなかった機能（他の図形の書き方）などを自主的に調べ、理解し、自分で試してみる。そして、自分なりに工夫したことを振り返り、省察することで、さらに実力アップ。

- Trinket は**オンライン**の Python、HTML 等の**学習サイト**
- 有料の機能と無料の機能がある
- **自分が作成した Python プログラムを公開し、他の人に実行してもらうことが可能**（そのとき、書き替えて実行も可能）
- **Python の標準機能**を登載、その他、次のパッケージがインストール済み

math, matplotlib.pyplot, numpy, operator, processing, pygal, random, re, string, time, turtle, urllib.request

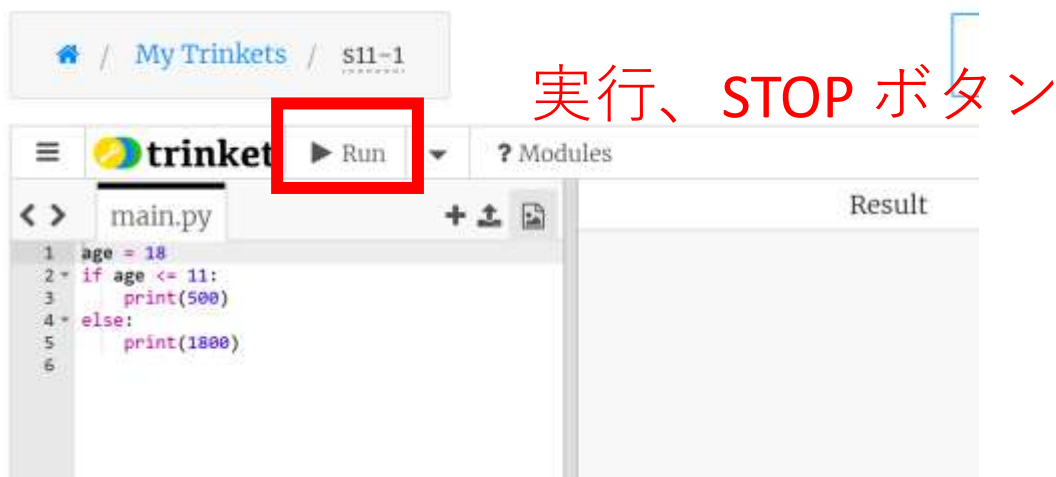
# trinket でのプログラム実行



- trinket は Python, HTML などのプログラムを書き実行できるサイト

- <https://trinket.io/python/cdc4896571>

のように、違うプログラムには違う URL が割り当てられる



ソースコードの  
メイン画面

実行結果

- 実行が開始しないときは、「**実行ボタン**」で**実行**
- ソースコードを書き替えて再度実行することも可能



各自の自発的な演習、自己研鑽の時間

## ① Pythonでグラフィックスを描く

資料のプログラムを動かして理解を深める

## ② Pythonの基本を押さえる

オブジェクト、メソッド、引数

## ③ 発想力、創造力

turtleモジュールを使用して、あなた自身がデザインした図形を描く。

## ④ 自主性、自己研鑽力、自分なりに工夫したことを振り返る

説明されなかった機能（他の図形の書き方）などを自主的に調べ、理解し、自分で試してみる。そして、自分なりに工夫したことを振り返り、省察することで、さらに実力アップ。

- Trinket は**オンライン**の Python、HTML 等の**学習サイト**
- 有料の機能と無料の機能がある
- **自分が作成した Python プログラムを公開し、他の人に実行してもらうことが可能**（そのとき、書き替えて実行も可能）
- **Python の標準機能**を登載、その他、次のモジュールやパッケージがインストール済み

math, matplotlib.pyplot, numpy, operator, processing, pygal, random, re, string, time, turtle, urllib.request

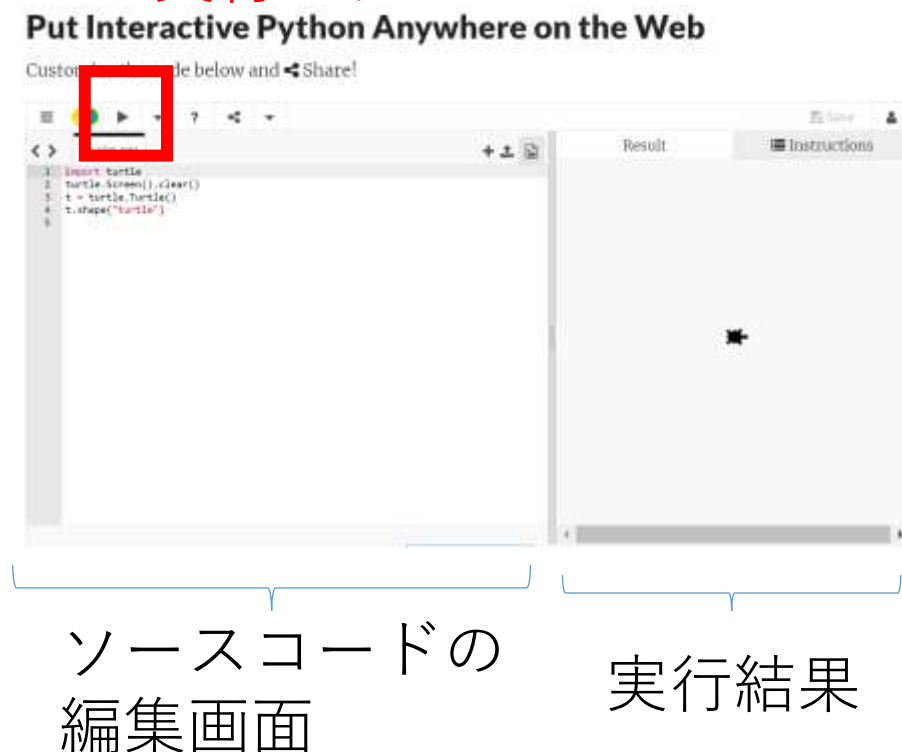


# trinket でのプログラム実行

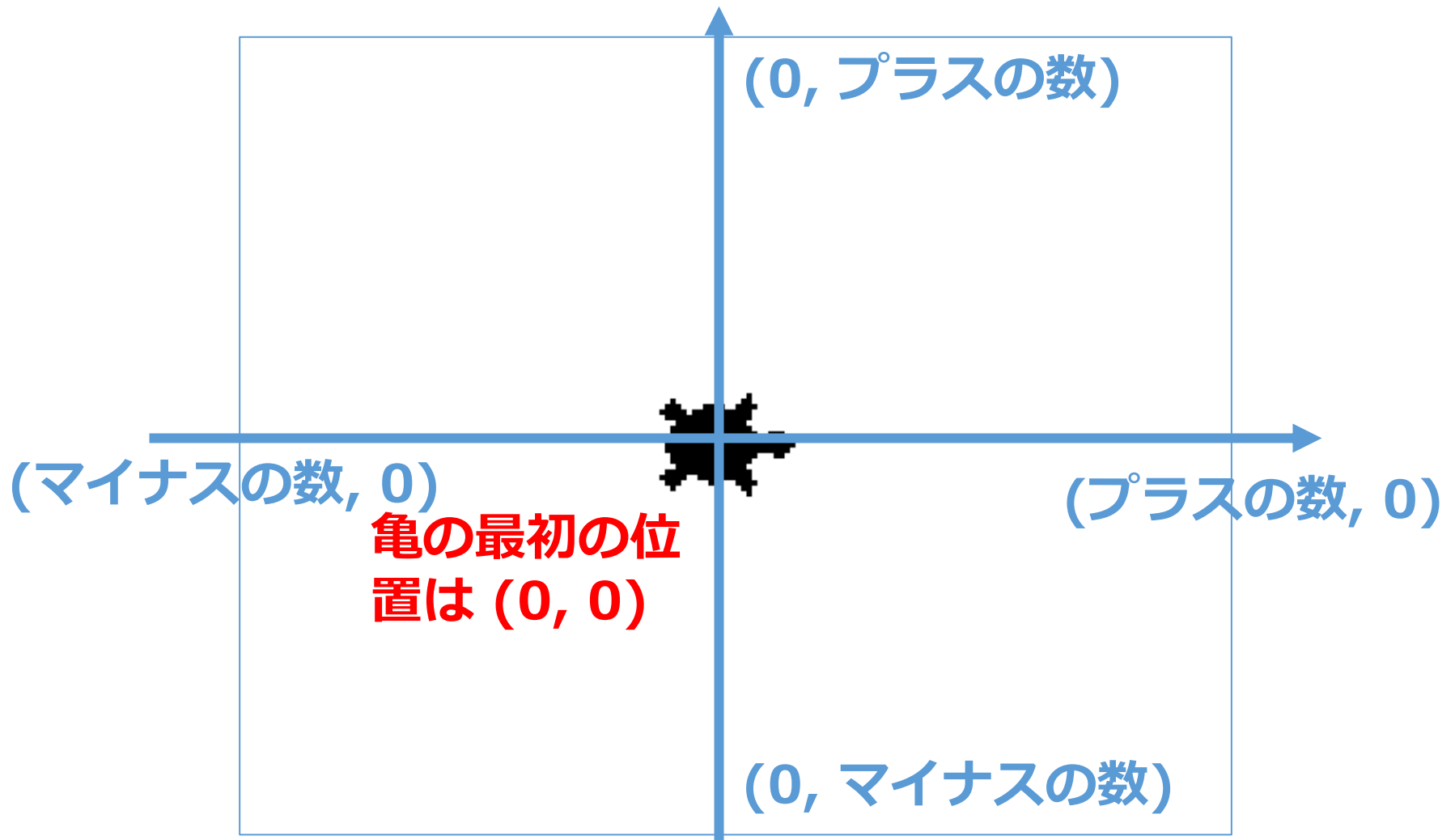


- URL: <https://trinket.io/python/cdc4896571> のようになる  
違うプログラムには違う URL が割り当てられる

## 実行ボタン



- 「**実行ボタン**」で**実行**
- ソースコードを書き替えて再度実行することも可能





### ① 1 つめ

<https://trinket.io/python/f29bfe71cd>

### ② 2 つめ

<https://trinket.io/python/5366def2f4>

### ③ 3 つめ

<https://trinket.io/python/f8cd554693>

①



```
import turtle
```

```
t = turtle.Turtle()
```

```
t.goto(0,100)
```

```
t.goto(100,0)
```

モジュールのインポート

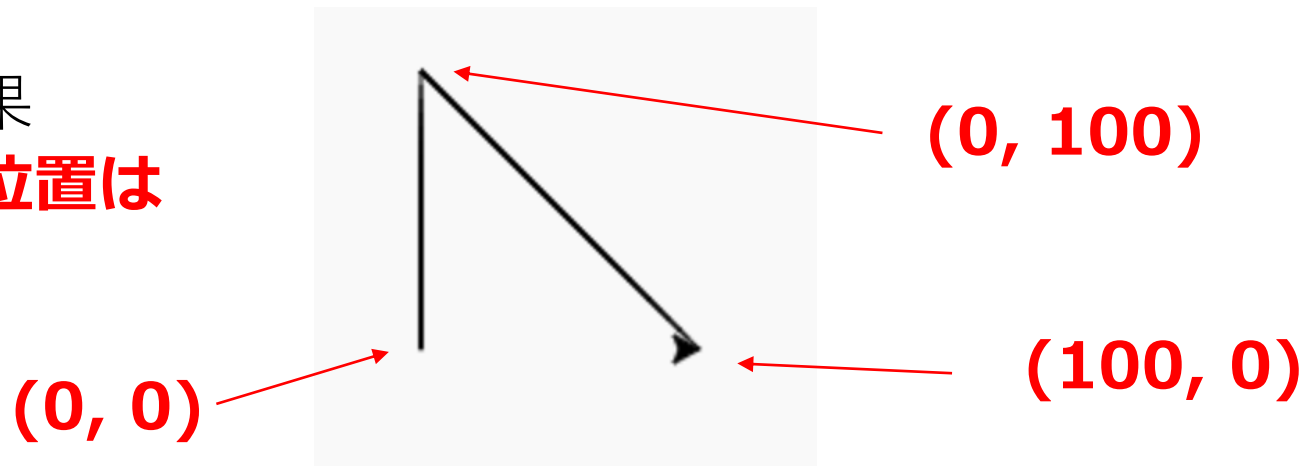
オブジェクト生成。t へのセット。

**(0, 100) への移動**

**(100, 0) への移動**

実行結果

**最初の位置は  
(0, 0)**



②



```
import turtle
```

```
t = turtle.Turtle()
```

```
t.goto(0, 100)
```

```
t.goto(58, -80)
```

```
t.goto(-95, 30)
```

```
t.goto(95, 30)
```

```
t.goto(-58, -80)
```

```
t.goto(0, 100)
```

モジュールのインポート

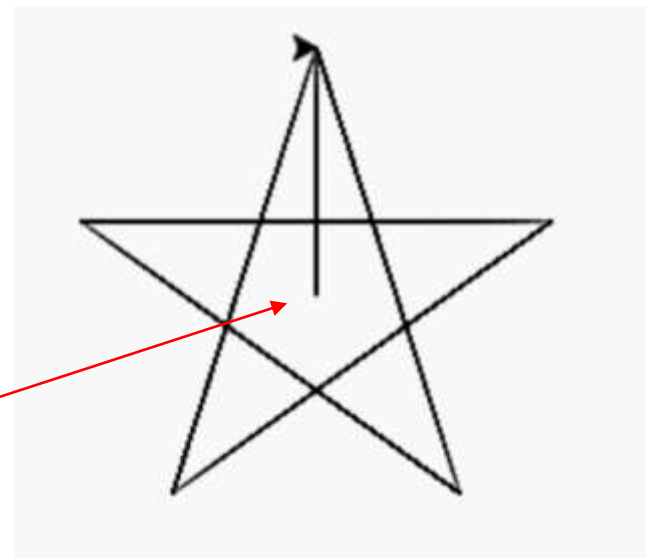
オブジェクト生成。t へのセット。

**移動**

実行結果

**最初の位置は  
(0, 0)**

**(0, 0)**



### ③色、円



```
import turtle
```

```
t = turtle.Turtle()
```

```
colors = ["red", "green", "blue"]
```

```
for i in range(3):
```

```
    t.color(colors[i])
```

```
    t.circle(30)
```

```
    t.forward(50)
```

モジュールのインポート

オブジェクト生成。t へのセット。

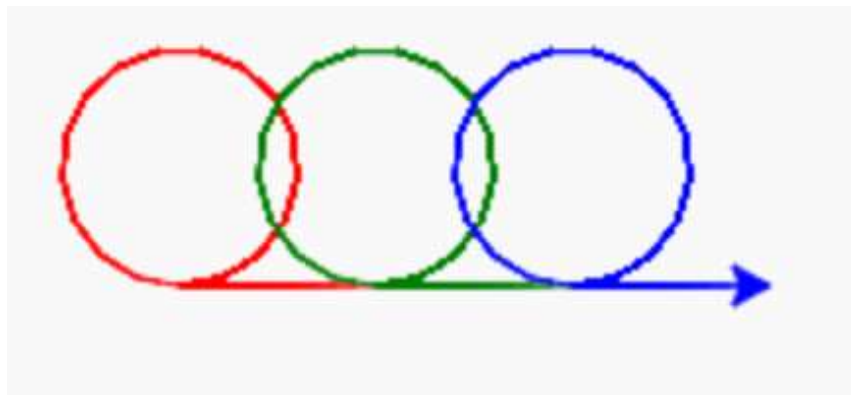
色は、赤、緑、青

色を変える

半径 30 の円

前に50進む

実行結果







# プログラミングまとめ

- ①卒業後は、ITエンジニアとして活躍しよう
- ②**基本的なプログラミングの知識**：モジュールの使用、オブジェクトの生成、メソッドの使用をおさえておくこと
- ③プログラミングの世界では、**失敗やエラーは成長のチャンス**になる。
- ④**楽しみながらプログラミングに取り組もう。プログラミングは創造的な作業であり、自分のアイデアを形にすることができるもの。**（「数学の問題を解く」ような勉強とは違う）
- ⑤**自分が興味を持つテーマやプロジェクト**を見つけ、プログラミングによる作品作りに**挑戦**してみよう。行動や体験が自信と成長につながる。
- ⑥**仲間やインターネット、先生、対話型AI（ChatGPTなど）から学び、教えることも重要。共同作業やアイデアの共有を通じて、学び、新たな視点を得ることができる**



### 3. 対話型AI (ChatGPT)

相談相手、質問相手として活用

- AI による対話システム
- 文章処理（要約、推敲、翻訳）のほか、問答により学び、新たな視点を得ることができる
- URL: <https://chat.openai.com/>
- 登録必要
- **AIの回答が間違っている場合がある（必ず根拠を確かめよう）**
- **秘密の情報やプライバシーについて投稿してはいけない**
- 大学のレポートなどは、学生自身の成果物を求めているので、AIの回答をそのまま提出してはいけない

# ChatGPT の登録手順（登録だけであればクレジットカード等は不要．スマホは必要）



- ① ChatGPT の公式ページにアクセス  
<https://chat.openai.com/>
- ② 「**Sign up**」をクリック
- ③ メールアドレスとパスワードを各自で設定
- ④ 「**Continue**」をクリック
- ⑤ ③のメールアドレスに確認メールが届くので、確認メール内の「**Verify email address**」をクリック
- ⑥ 氏名、誕生日、スマートフォンの電話番号を登録（画面は閉じないこと）
- ⑦ ⑥の**電話番号の SMS** に 6 桁の認証コードが届くので、画面に認証コードを入れる
- ⑧ 「**Verify**」をクリック
- ⑨ 使用目的を選択

# ChatGPT との問答の例① 質問



KU

タートルグラフィックスで `t.shape("turtle")` を教わった。turtle 以外にあるか



はい、Pythonの`turtle`モジュールでは、`shape()`メソッドに指定できる形状がいくつかプリセットされています。これらは下記の通りです：

- "arrow"
- "blank"
- "circle"
- "classic"
- "square"
- "triangle"
- "turtle"

たとえば、"circle"や"triangle"を使って図形を描くことができます：

python

 Copy code

```
import turtle

t = turtle.Turtle()

# Circle shape
t.shape("circle")
```

# ChatGPT との問答の例② プログラム作成を頼む



KU

次のプログラムを工夫して、花の形を描きたい

```
import turtle
turtle.Screen().clear()
t = turtle.Turtle()
t.shape("turtle")
t.goto(0,100)
t.goto(100,0)
```

以下のように、与えられたプログラムを工夫して花の形を描くことができます。

```
python

import turtle

def draw_flower():
    window = turtle.Screen()
    window.bgcolor("white")

    # 花の色を設定
    colors = ["red", "blue", "purple", "yellow", "orange", "pink"]

    # Turtleを作成
    flower = turtle.Turtle()
    flower.shape("turtle")
    flower.speed(10)

    for _ in range(36):
        flower.color(colors[_ % len(colors)])

    # 花びらの描画
    flower.goto(0, 100)
    flower.goto(100, 0)
    flower.goto(0, -100)
    flower.goto(-100, 0)
    flower.goto(0, 100)
    flower.right(10)

    flower.hideturtle()
    window.exitonclick()

# プログラムの実行
draw_flower()
```

# ChatGPT との問答の例③ プログラム内のミスを探すことを頼む



次は Python のタートルグラフィックスのプログラムです。間違いがあれば修正してください。import turtle  
turtle.Screen().clear()  
t = turtle.turtle()  
t.shape("turtle")  
t.goto(0,100)  
t.goto(100,0)

プログラムの一部にいくつかの修正が必要です。以下に修正版を示します。

```
python Copy code  
  
import turtle  
  
screen = turtle.Screen()  
screen.clear()  
  
t = turtle.Turtle()  
t.shape("turtle")  
t.goto(0, 100)  
t.goto(100, 0)
```

修正内容:

1. `'turtle.turtle()'` ではなく `'turtle.Turtle()'` を使用します。
2. `'turtle.Screen().clear()'` ではなく、`'screen = turtle.Screen()'` としてスクリーンオブジェクトを作成し、`'screen.clear()'` を呼び出します。

修正後のプログラムは、タートルグラフィックスのウィンドウを作成し、タートルを亀の形に設定して、座標 (0, 100) から座標 (100, 0) まで移動します。



# ChatGPT との問答の例④ 疑問に思ったことを相談する



Pythonの勉強をするときturtleグラフィックスで何を練習しますか



Pythonの勉強において、turtleグラフィックスは以下のような項目で練習することができます。

1. 基本的な動きの制御: turtleを前進させたり後退させたり、左右に回転させたりする方法を学びます。これにより、turtleがどのように動くのかを理解し、基本的なコマンドを使いこなすことができます。
2. 図形の描画: turtleを使用して、線や多角形、円などの図形を描画します。turtleがどのように動いて図形を描くのかを理解し、線の長さや角度を変えることで様々な形を作り出すことができます。
3. ループの活用: turtleグラフィックスを使って、繰り返し処理を行います。ループを活用することで、複雑なパターンや図形を効率的に描くことができます。
4. 条件分岐の応用: turtleが特定の条件に応じて動くように制御します。例えば、特定の位置に到達したら色を変える、特定の角度になったら方向を変えるなどの条件分岐を組み込むことができます。
5. アニメーションの作成: turtleの動きを連続的に更新することでアニメーションを作成します。ループや時間の概念を組み合わせることで、turtleが動く様子をリアルタイムで表示することができます。

これらの要素を組み合わせ、turtleグラフィックスを活用した様々なプログラムを作成することで、Pythonの基本的な構文や制御フローの理解を深めることができます。また、創造力を発揮して自分自身のアイデアを実現することもできます。



# ChatGPT との問答の例⑤ 勉強用に要点の作成を頼む



KU

Pythonの基礎を7行以内でまとめ、表形式で表示してください



以下にPythonの基礎を7行以内でまとめた表を示します。

項目	例
データ型	数値: <code>`1`</code> , <code>`3.14`</code>  文字列: <code>`'Hello'`</code> , <code>`"Python"`</code>  ブール値: <code>`True`</code> , <code>`False`</code>
変数の代入	<code>`x = 10`</code> <code>`name = 'Alice'`</code>
演算子	算術演算子: <code>`+`</code> , <code>`-`</code> , <code>`*`</code> , <code>`/`</code>  比較演算子: <code>`==`</code> , <code>`!=`</code> , <code>`&lt;`</code> , <code>`&gt;`</code>  論理演算子: <code>`and`</code> , <code>`or`</code> , <code>`not`</code>
制御構造	条件分岐: <code>`if`</code> 条件: <code>`&lt;br&gt;`</code> 繰り返し: <code>`for`</code> 要素 <code>`in`</code> リスト: <code>`&lt;br&gt;`</code> ループの中断: <code>`break`</code>  ループのスキップ: <code>`continue`</code>
リスト	<code>`numbers = [1, 2, 3]`</code> <code>`fruits = ['apple', 'banana', 'orange']`</code>
関数	関数の定義: <code>`def`</code> 関数名(引数): <code>`&lt;br&gt;`</code> 関数の呼び出し: <code>`関数名(引数)`</code>
モジュール	モジュールのインポート: <code>`import`</code> モジュール名 モジュールの機能の利用: <code>`モジュール名.機能名`</code>

この表にはPythonの基礎として代表的な要素がまとめられています。それぞれの項目には具体的な例が示されており、コードの一部を想像する助けとなるでしょう。ただし、Pythonの基礎を完全に網羅しているわけではありませので、さらなる学習を進める際には詳細なリソースやドキュメントを参照することをおすすめします。

# ChatGPT との問答の例⑥ 勉強用に確認問題の作成を頼む



KU

Pythonの基礎を確認できる、下線補充の問題を7問作成してください。

1. データ型: 整数型の変数 `age` に値 `25` を代入するコードを書いてください。

```
python Copy code  
  
age = 25
```

1. 変数の代入: 文字列型の変数 `name` に値 `"John"` を代入するコードを書いてください。

```
python Copy code  
  
name = "John"
```

1. 演算子: 変数 `x` と `y` の和を計算し、結果を変数 `total` に代入するコードを書いてください。

```
python Copy code  
  
total = x + y
```

1. 制御構造: 条件式 `x > 10` が真の場合に `"x is greater than 10"` と表示する条件分岐のコードを書いてください。

```
python Copy code  
  
if x > 10:  
    print("x is greater than 10")
```

1. リスト: 整数値 `1`、`2`、`3` を要素とするリスト `numbers` を作成するコードを書いてください。

```
python Copy code  
  
numbers = [1, 2, 3]
```

1. 関数: 引数 `name` を受け取り、`"Hello, name!"` というメッセージを表示する関数 `greet` を定義するコードを書いてください。

```
python Copy code  
  
def greet(name):  
    print("Hello, " + name + "!")
```

1. モジュールの利用: `math` モジュールをインポートし、`math.sqrt()` 関数を使って変数 `x` の平方根を計算するコードを書いてください。

```
python Copy code  
  
import math  
  
result = math.sqrt(x)
```

- AIの回答が間違っている場合がある:

ChatGPTは、不正確な回答を提供する可能性がある。提供される回答は、参考にとどめ、**必ず他の信頼性の高い情報源などを確認**する。

- 秘密の情報やプライバシーについての投稿:

**個人情報や機密情報を投稿しないこと**

- 自習などでの利用:

ChatGPTは、相談相手として、自習などに役立つ。しかし、**ChatGPT の回答をそのまま学校のレポートや宿題として提出してはいけません**。レポートや宿題は、学生自身の知識と理解を高めるためのもの。AIは参考として活用し、自分自身の考えや研究成果など、自分の成果物を提出することが大切。



## 4. AI のプログラム例





# 画像分類 「犬」か「猫」か？

人工知能が、画像分類を行う。与えられた画像が特定のカテゴリに属するかどうかを判定するもの。



# 人工知能エンジニアのスキル



人工知能エンジニアは、**多様な幅広いスキル**を活用

- 機械学習、ディープラーニング（ニューラルネットワーク、CNN、Transformer など）
- プログラミング（Python、関連スキル）
- データの前処理や選別、データサイエンス
- 倫理的な理解（AIの社会への影響、プライバシーなど）
- ソフトスキル（チームで働く能力、スケジュール管理能力、コミュニケーション力、説明力）

# 人工知能を制作し実行するためのプログラミングスキル



- **プログラミングは、基本を押さえる**（パッケージ、オブジェクト生成、メソッド、変数、関数、if、for）
- **関連スキル**を必要に応じて自主的に学ぶ
  - **システムデザイン**: ソフトウェアのアーキテクチャと設計
  - **試作とテスト**: コードが意図した通りに動作することを確認。テスト駆動開発（TDD）など。
  - **バージョン管理**: gitのようなバージョン管理ツールで、プログラムの変更履歴を管理。
  - **クラウドの活用**: AWSなどのクラウドサービスを利用。アプリケーションのデプロイ。
  - **データベース管理**: SQLなどのデータベース言語。
  - **高速数値計算**: NumPyやPandasのようなライブラリを使って、大量のデータを効率的に処理。
  - **ビッグデータ処理**: 大規模なデータセットを扱うHadoopやSparkなど。

# 人工知能プログラムの構成



1. 学習データの読み込みと前処理
2. モデルの構築
3. モデルのコンパイル
4. 学習
5. モデルの検証
6. タスク（人工知能に本来させたい処理）の実行



# 「犬」か「猫」かの判別のために 学習データを用いる



1 7 6 の犬画像と 1 8 3 の猫画像で  
人工知能が学習

人工知能エンジニアは「データから新しい価値を  
創造できる能力」を持つ

# 演習

## カラー写真の犬と猫の分類

### 【構成】

- ①プログラムを実行
- ②ソースコードの確認

### 【トピックス】

- 人工知能プログラムの構成：1. 学習データの読み込みと前処理、2. モデルの構築、3. モデルのコンパイル、4. 学習、5. モデルの検証、6. タスクの実行
- プログラミングの基本：パッケージ、オブジェクト生成、メソッド、変数、関数、if、for

# この演習を実施するための前準備



1. Python のインストール

2. コマンドプロンプトを管理者として実行し、次のコマンドを実行

```
python -m pip install -U pip setuptools
```

```
python -m pip install -U jupyterlab jupyter jupyter-console jupyter-text PyQt5
```

```
interact-on-jupyter spyder
```

```
python -m pip install -U PythonTurtle
```

```
python -m pip install tensorflow==2.10.1 pillow matplotlib
```

```
mkdir c:\coco2017
```

```
cd c:\coco2017
```

```
curl -O https://www.kkaneko.jp/a/dog.zip
```

```
curl -O https://www.kkaneko.jp/a/cat.zip
```

```
curl -O https://www.kkaneko.jp/a/dogcat.zip
```

```
powershell Expand-Archive -DestinationPath . dog.zip
```

```
powershell Expand-Archive -DestinationPath . cat.zip
```

```
powershell Expand-Archive -DestinationPath . dogcat.zip
```

```
del ensu1.py
```

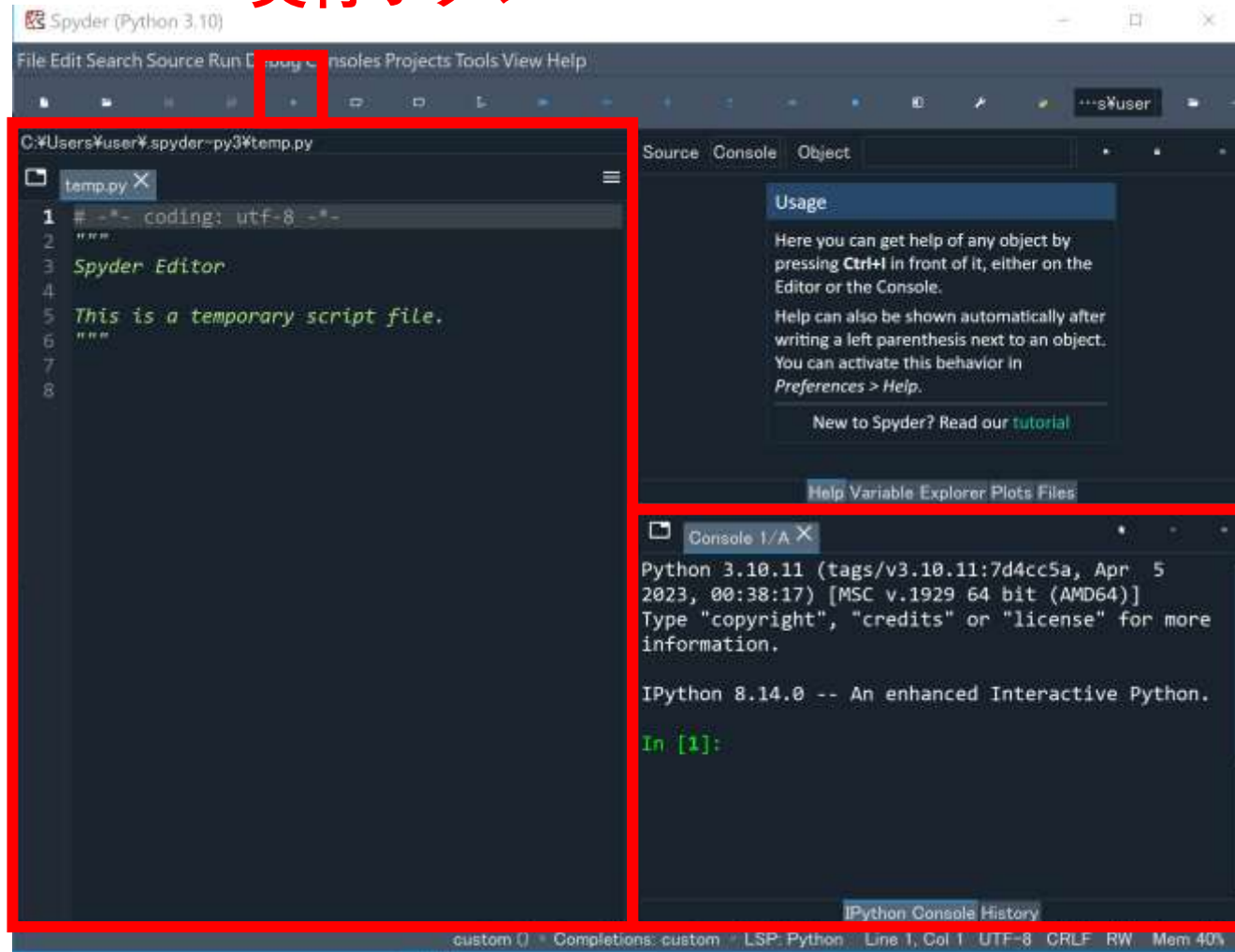
```
curl -O https://www.kkaneko.jp/a/ensu1.py
```

# Spyder の画面



## 実行ボタン

編集



コンソール

# Spyderの起動



## Spyder は Python の開発環境

### 起動

- ① コマンドプロンプトを開く（検索窓で **cmd** が便利）



- ② コマンドプロンプトを開く → Enter キー

```
Microsoft Windows [Version 10.0.19041.264]
(c) 2020 Microsoft Corporation. All rights reserved.

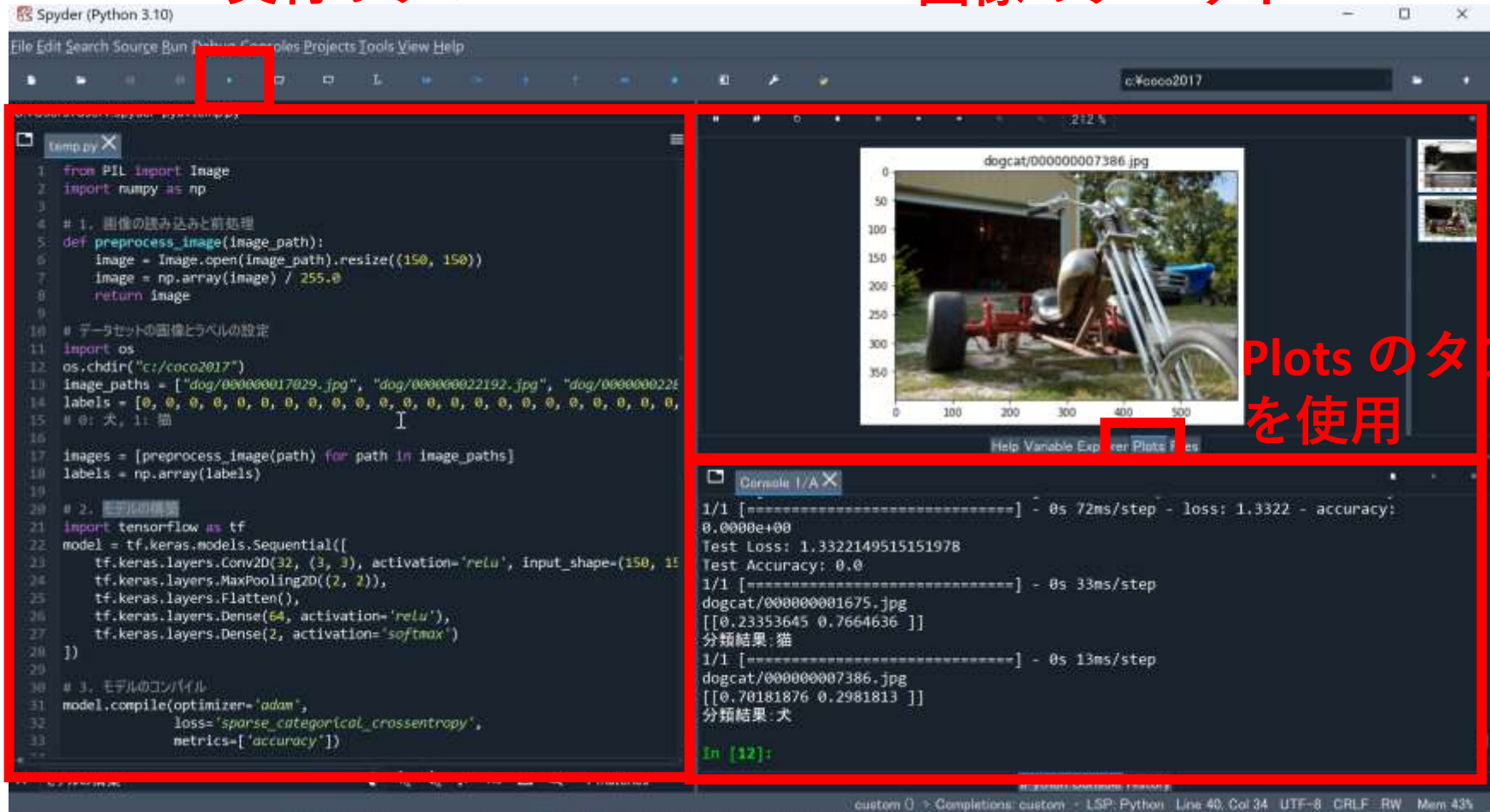
C:\Users\user>spyder_
```



## ① プログラムを実行

## 実行ボタン

## 画像のプロット



# 編集

# コンソール

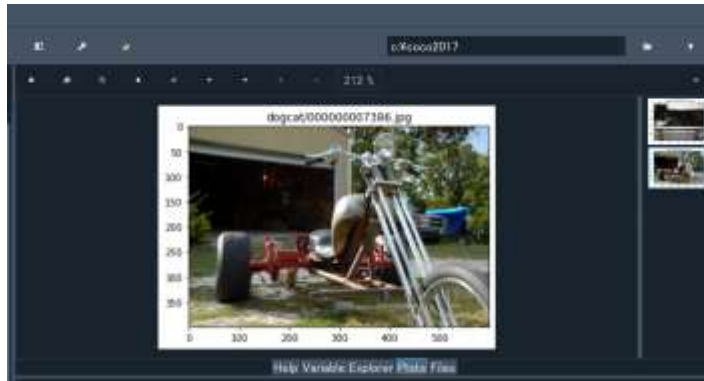


```
Test Accuracy: 0.75
1/1 [=====] - 0s 33ms/step
dogcat/000000001675.jpg
[[0.23353645 0.7664636 ]]
分類結果;猫
1/1 [=====] - 0s 13ms/step
dogcat/000000007386.jpg
[[0.70181876 0.2981813 ]]
分類結果:犬
```

## コンソール

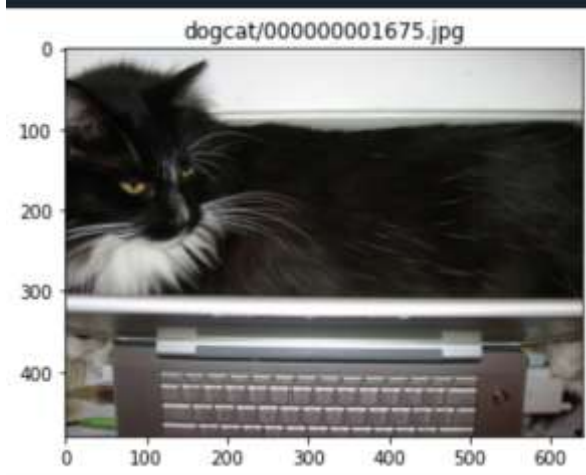
1 枚目の画像	犬である確率	23%
	猫である確率	77%
2 枚目の画像	犬である確率	70%
	猫である確率	30%

(乱数を使っているので実行のたびに結果が変わる)



←選択可能

右上の画面に画像表示



1 枚目の画像



2 枚目の画像





57



```

30 # 3. モデルのコンパイル
31 model compile optimizer='adam',
32         loss='sparse_categorical_crossentropy',
33         metrics=['accuracy'])
34
35 # 4. 学習
36 model fit(np.array(images), labels, epochs=5)
37
38 # 5. モデルの評価
39 test_images = [preprocess_image(path) for path in ["dogcat/000000001675.jpg", "a
40 test_labels = np array [0, 1]) # テスト用のラベル
41 test_loss, test_accuracy = model.evaluate np array test_images), test_labels)
42 print("Test Loss:", test_loss)
43 print("Test Accuracy:", test_accuracy)
44
45 # 6. タスクの実行(学習済みモデルを使用して新たな画像に対して予測を行う)
46 import matplotlib.pyplot as plt
47 for test_image_path in ["dogcat/000000001675.jpg", "dogcat/000000007386.jpg"]: #
48     test_image = preprocess_image test_image_path)
49     test_image = np expand dims test_image, axis=0) # 予測のために次元を追加
50     predictions = model predict test_image)
51     image = plt.imread test_image_path)
52     plt.imshow image)
53     plt.title test_image_path)
54     plt.show()
55     print(test_image_path)
56     print(predictions)
57     predicted_label = np argmax predictions[0])
58     if predicted_label == 0:
59         print("分類結果:犬")
60     else:
61         print("分類結果:猫")

```


 モジュールや  
パッケージ


 関数やメソッドや  
クラス名

# 人工知能による画像認識

画像認識でできることはたくさんある

- 画像分類
  - 物体検出
  - 顔検出
  - 表情推定
  - 文字検出
  - セグメンテーション
  - 追跡
  - 姿勢推定
- など

- **画像分類:** 与えられた画像が「犬」か「猫」かを判定するなど
- **人工知能エンジニアのスキル:** 機械学習、ディープラーニング、プログラミング、データの前処理、倫理的な理解、ソフトスキル。
- **人工知能制作と活用のためのプログラミングスキル:** 基本の押さえ、関連スキルの自主学習（システムデザイン、試作とテスト、バージョン管理、クラウドの活用、データベース管理、高速数値計算、ビッグデータ処理）
- **人工知能プログラムの構成:** 学習データの読み込みと前処理、モデルの構築、モデルのコンパイル、学習、モデルの検証、タスクの実行。

- Pythonの基礎を実践的に学び、プログラミングでアイデアを形にする喜び。
- 対話型AIの活用を通じて、AIの仕組みと開発手順を体験的に理解。
- 人工知能エンジニアに必要な技術と倫理への理解を深め、将来の可能性を広げる。
- 自主的な演習と課題で、問題解決力と自己研鑽力を伸ばせる。
- 最先端技術に触れる喜びから、情報工学への興味と学習意欲が高まり、未来のIT社会での活躍への自信がつく。