

3. Python入門：プログラミングの基礎と創造的学習への展開

Pythonプログラミング講座：基礎から応用まで
(全15回)

URL: <https://www.kkaneko.jp/pro/pf/index.html>

金子邦彦



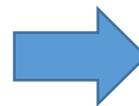
3-1. Python プログラミングの基礎

Python プログラミングの基礎の全体像



- **プログラミング**は、**プログラム**を作成する**創造的な活動**
- **プログラム**は、**コンピュータ**に指示を出し、所定の作業を遂行させる
- **Python** は、**読みやすさ**、**書きやすさ**、**幅広い応用範囲**が特徴

```
a = [200, 400, 300]
for i in a:
    print (i * 1.08)
```



```
216.0
432.0
324.0
```

Python 言語のプログラム

プログラムの
実行結果

変数, 代入

- **変数** : プログラム内で名前を付けて利用する **オブジェクト** である. **値を保存** し, 後から **参照** できる仕組み.
- **代入** : 「**x = 100**」のように書くことで, **x という名前の変数に、値 100 が保存** される操作

例

```
x = 100
```

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

t.goto(0,100)

t オブジェクト

goto(0,100) 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.
- **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

例 **t.goto(0,100)**



引数は 0 と 100

Python プログラムの書き方（コーディングスタイル）



- **インデント（字下げ）によるブロックの区切り**
(通常「タブ」または「4つの半角スペース」)
- **コメント**：行の先頭に「#」．プログラムの説明を記述
- **空白行**：プログラムを読みやすくするために挿入可能

例

```
import turtle
```

空白行

```
for i in range(4):
```

```
    turtle.forward(100)
```

```
    turtle.right(90)
```

} インデント（字下げ）

空白行

```
turtle.done()
```



3-2. 変数, 代入

変数, 代入

- **変数** : プログラム内で名前を付けて利用する **オブジェクト** である. **値を保存** し, 後から **参照** できる仕組み.
- **代入** : 「**x = 100**」のように書くことで, **x という名前の変数に、値 100 が保存** される操作

例

```
x = 100
```

- 保存された値は, プログラムの中で何度でも参照することが可能
- 別の値で上書きすることも可能.

Trinket の概要



- Trinket は**オンライン**の Python、HTML 等の**学習サイト**
- ブラウザで動作
- 有料の機能と無料の機能を提供
- **自作プログラムの公開と共有が可能**
- Python の**標準機能**に加え，外部ライブラリ matplotlib.pyplot, numpy, processing, pygal が利用可能

A screenshot of the Trinket web interface. At the top, there is a breadcrumb navigation path: a home icon, followed by "/ My Trinkets / s11-1". Below this is a toolbar with a hamburger menu icon, the Trinket logo (a key icon), the text "trinket", a "Run" button with a play icon, and a dropdown menu labeled "? Modules". The main area is split into two panes. The left pane is a code editor titled "main.py" with a file icon, containing the following Python code:

```
1 age = 18
2 if age <= 11:
3     print(500)
4 else:
5     print(1800)
6
```

The right pane is titled "Result" and is currently empty.

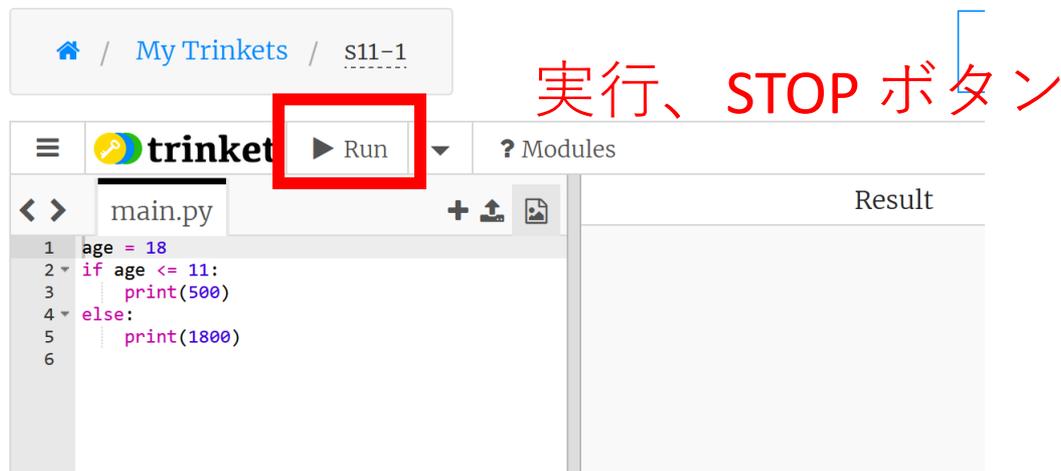


Trinket の基本操作

- 公開プログラムごとに固有の URL が割り当てられる

例 <https://trinket.io/python/0fd59392c8>

- 「Run」ボタンによるプログラムの開始, 「STOP」ボタンによる終了



確認編集用の
メイン画面

実行結果

- **メイン画面でのプログラム編集と再実行が可能**
- 左側で実行結果を確認

演習. 変数と代入

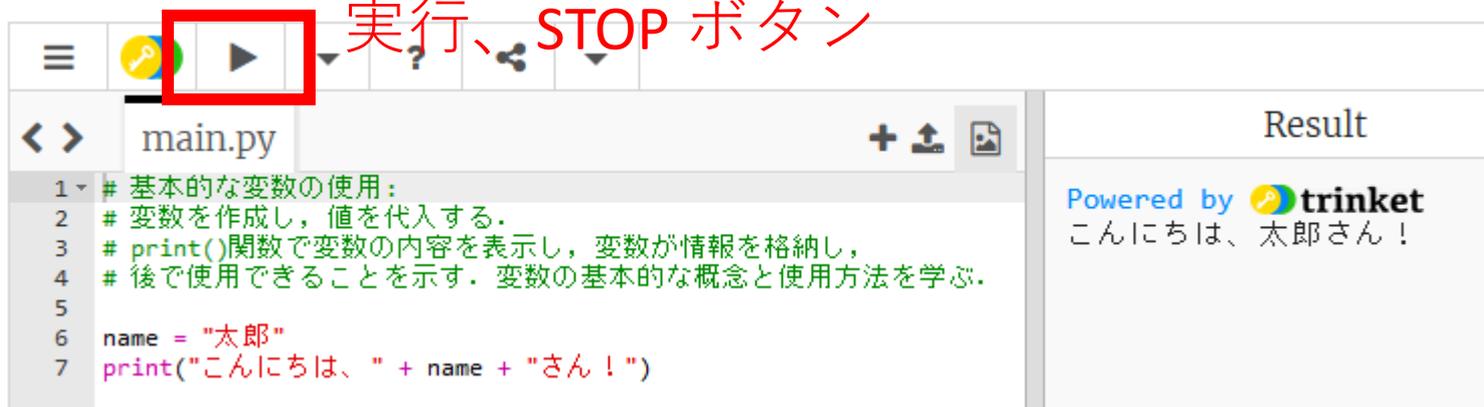
1. 基本的な変数の使用

① trinket の次のページを開く

<https://trinket.io/python/abafd851480a>

- 変数
- 値を代入方法
- print() 関数を使って変数の内容を表示する方法

② 実行結果が、次のように表示されることを確認



実行、STOP ボタン

```
1 # 基本的な変数の使用:  
2 # 変数を作成し、値を代入する。  
3 # print()関数で変数の内容を表示し、変数が情報を格納し、  
4 # 後で使用できることを示す。変数の基本的な概念と使用方法を学ぶ。  
5  
6 name = "太郎"  
7 print("こんにちは、" + name + "さん!")
```

Result

Powered by trinket
こんにちは、太郎さん！

- 実行が開始しないときは、「**実行ボタン**」で**実行**
- プログラムを書き替えて再度実行することも可能

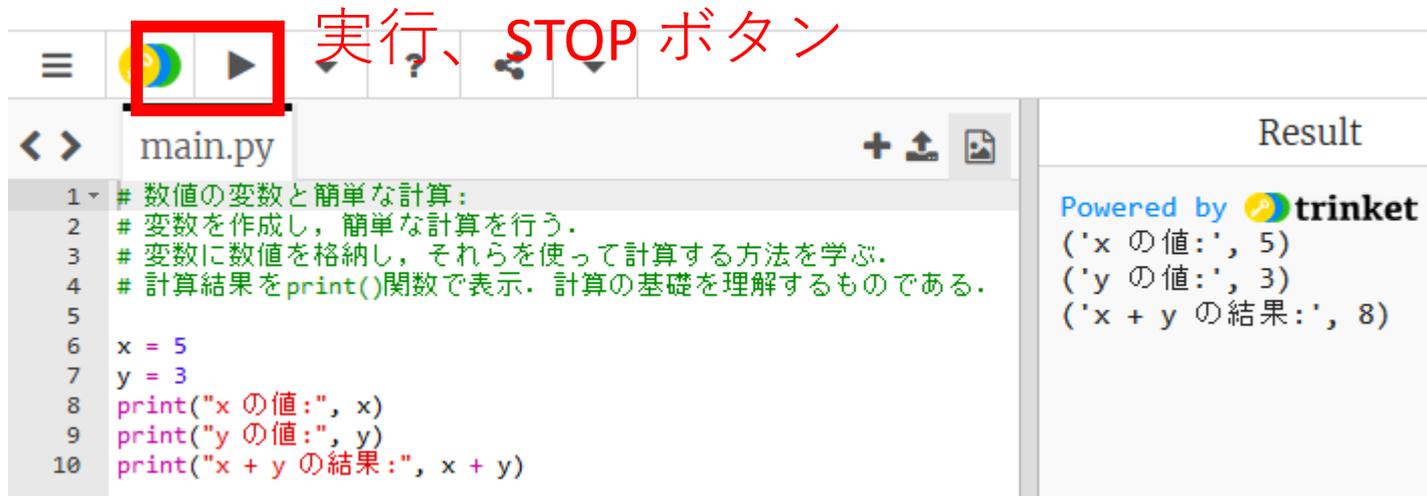
2. 基本的な変数の使用

① trinket の次のページを開く

<https://trinket.io/python/9870e86d63b9>

- 複数の変数を使って簡単な計算を行う
- print() 関数を使って変数の内容を表示する

② 実行結果が、次のように表示されることを確認



実行、STOP ボタン

```
1 # 数値の変数と簡単な計算:  
2 # 変数を作成し、簡単な計算を行う。  
3 # 変数に数値を格納し、それらを使って計算する方法を学ぶ。  
4 # 計算結果をprint()関数で表示。計算の基礎を理解するものである。  
5  
6 x = 5  
7 y = 3  
8 print("x の値:", x)  
9 print("y の値:", y)  
10 print("x + y の結果:", x + y)
```

Result

Powered by  trinket

('x の値:', 5)
('y の値:', 3)
('x + y の結果:', 8)

- 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

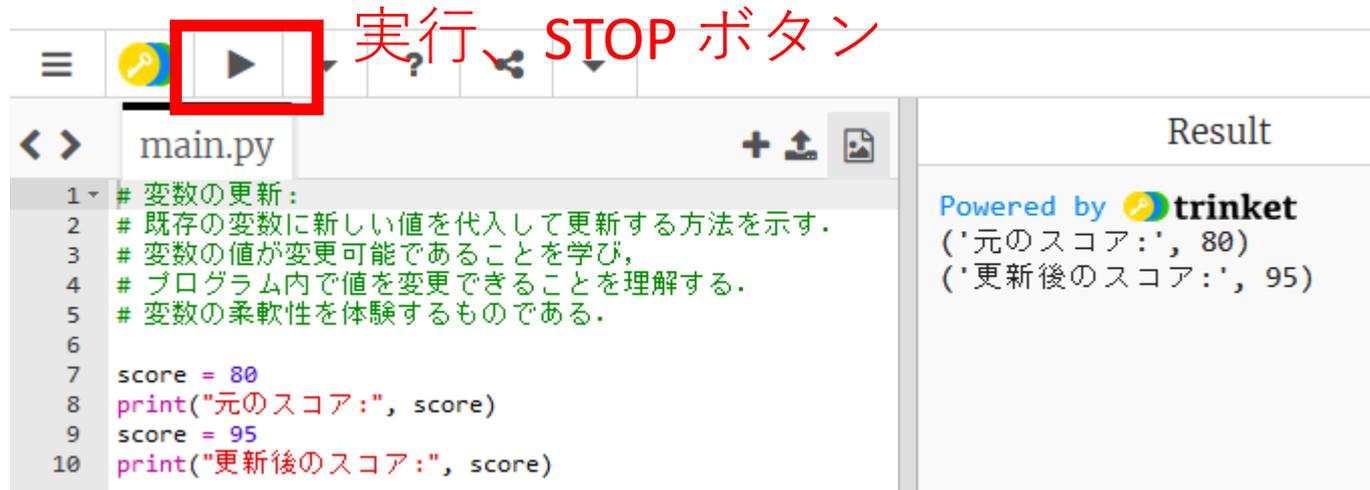
3. 変数の更新

① trinket の次のページを開く

<https://trinket.io/python/b869619b0874>

- 既存の変数に新しい値を代入して更新
- プログラムの中で変数の値が変化

② 実行結果が、次のように表示されることを確認



The screenshot shows the Trinket IDE interface. At the top, there is a toolbar with a play button (execution) and a stop button (STOP ボタン), both highlighted with a red box. Below the toolbar is a code editor with a file named 'main.py'. The code in the editor is as follows:

```
1 # 変数の更新:  
2 # 既存の変数に新しい値を代入して更新する方法を示す。  
3 # 変数の値が変更可能であることを学び、  
4 # プログラム内で値を変更できることを理解する。  
5 # 変数の柔軟性を体験するものである。  
6  
7 score = 80  
8 print("元のスコア:", score)  
9 score = 95  
10 print("更新後のスコア:", score)
```

To the right of the code editor is a 'Result' panel. It displays the output of the program:

```
Powered by trinket  
( '元のスコア:', 80)  
( '更新後のスコア:', 95)
```

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

4. ユーザー入力と変数

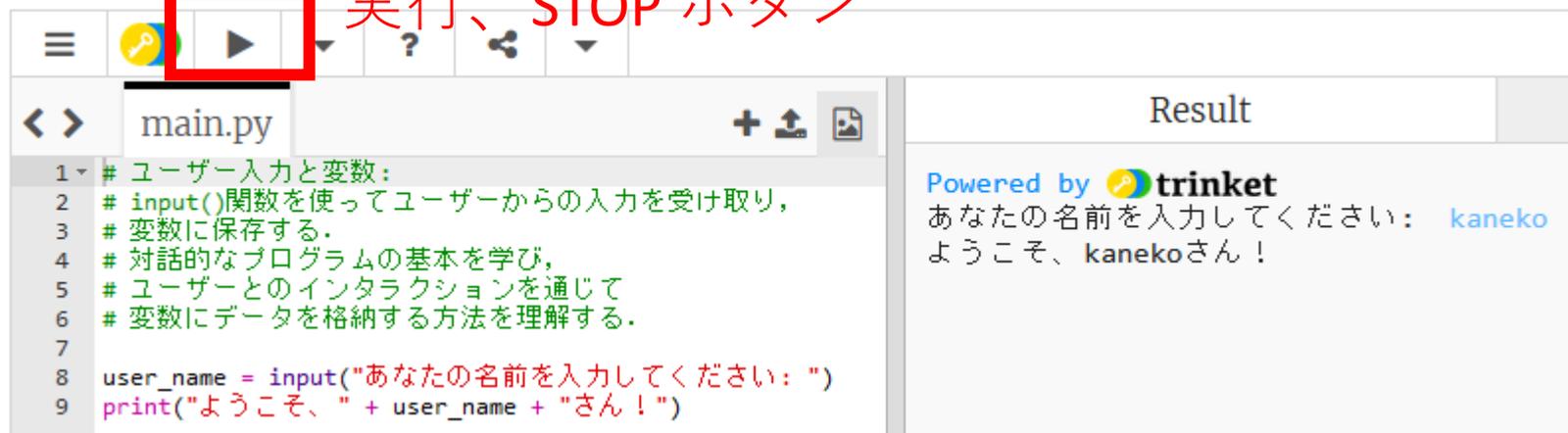
① trinket の次のページを開く

<https://trinket.io/python/45f0bed92360>

- input() 関数を使ってユーザーからの入力を受け取る
- 入力された値を変数に保存し、プログラム内で利用
- 対話的なプログラムの基本

② 実行結果が、次のように表示されることを確認

実行、STOP ボタン



```
1 # ユーザー入力と変数:  
2 # input()関数を使ってユーザーからの入力を受け取り,  
3 # 変数に保存する.  
4 # 対話的なプログラムの基本を学び,  
5 # ユーザーとのインタラクションを通じて  
6 # 変数にデータを格納する方法を理解する.  
7  
8 user_name = input("あなたの名前を入力してください: ")  
9 print("ようこそ、" + user_name + "さん!")
```

Result

Powered by  trinket
あなたの名前を入力してください: kaneko
ようこそ、kanekoさん!

- 実行が開始しないときは、「**実行ボタン**」で**実行**
- プログラムを書き替えて再度実行することも可能

5. 複数の変数

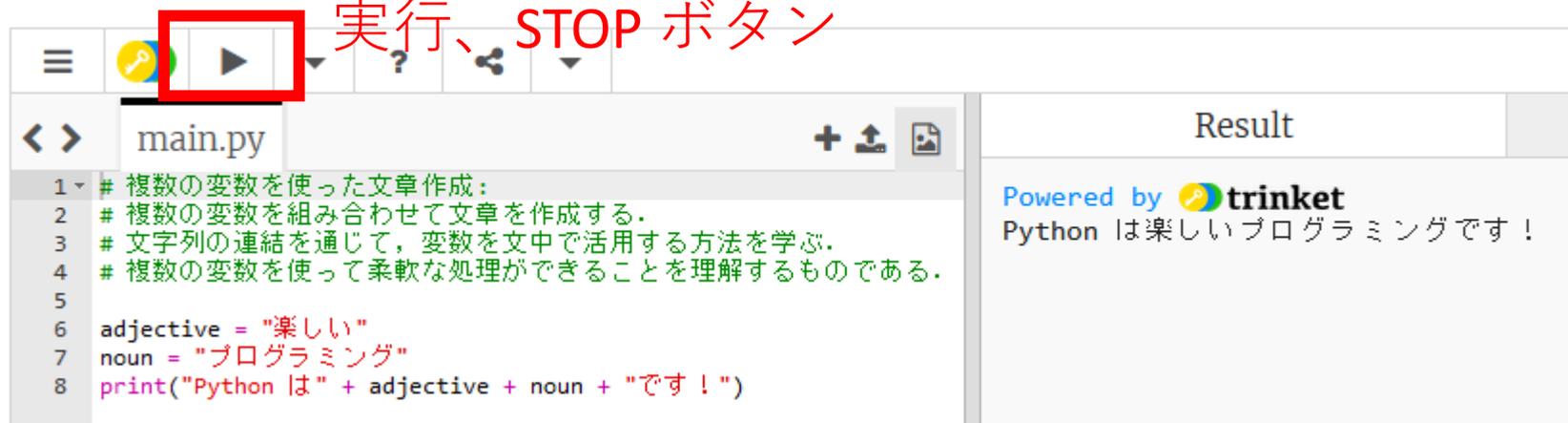
① trinket の次のページを開く

<https://trinket.io/python/abafd851480a>

- 異なる値を別々の変数に保存
- 複数の変数を利用して計算や表示を行う

② 実行結果が，次のように表示されることを確認

実行、STOP ボタン



```
1 # 複数の変数を使った文章作成:  
2 # 複数の変数を組み合わせて文章を作成する.  
3 # 文字列の連結を通じて、変数を文中で活用する方法を学ぶ.  
4 # 複数の変数を使って柔軟な処理ができることを理解するものである.  
5  
6 adjective = "楽しい"  
7 noun = "プログラミング"  
8 print("Python は" + adjective + noun + "です!")
```

Result

Powered by  trinket
Python は楽しいプログラミングです!

- 実行が開始しないときは、「**実行ボタン**」で**実行**
- プログラムを書き替えて再度実行することも可能

まとめ



- 変数と代入は、プログラミングにおける基本的なデータの保存と参照の仕組み
- 変数の使用方法を理解することで、プログラミング学習の基礎を確立できる

3-3. プログラミングの楽しさと達成感

プログラミング学習の意義と目標



- 論理的思考力や課題解決力の向上
- プログラムの作成と実行を通じて、コンピュータの動作原理を学び、問題解決能力を高める
- 自分の作ったプログラムが動作する喜びと達成感



ソースコード

- ソースコードは、プログラミング言語で書かれたプログラム
- 人間が読み書き，編集できる
- プログラムの動作を理解し，必要に応じて**改変**するための基礎になる



```
function() {
  //is the element hidden?
  if (!t.is(':visible')) {
    //it became hidden
    t.appeared = false;
    return;
  }

  //is the element inside the visible window?
  var a = w.scrollLeft();
  var b = w.scrollTop();
  var o = t.offset();
  var x = o.left;
  var y = o.top;

  var ax = settings.accX;
  var ay = settings.accY;
  var th = t.height();
  var wh = w.height();
  var tw = t.width();
  var ww = w.width();

  if (y + th + ay >= b &&
      y <= b + wh + ay &&
      x + tw + ax >= a &&
      x <= a + ww + ax) {

    //trigger the custom event
    if (!t.appeared) t.trigger('appear', settings.data);

  } else {

    //it scrolled out of view
    t.appeared = false;
  }
};

//create a modified fn with some additional logic
var modifiedFn = function() {

  //mark the element as visible
  t.appeared = true;

  //is this supposed to happen only once?
  if (settings.one) {

    //remove the check
    w.unbind('scroll', check);
    var i = $.inArray(check, $.fn.appear.checks);
    if (i >= 0) $.fn.appear.checks.splice(i, 1);

  }

  //trigger the original fn
  fn.apply(this, arguments);
};
```

演習でプログラミングを学ぶ意義, 成果



- ① **プログラミングの基本概念の理解**
- ② **創造的なプログラム作成の態度, 習慣**
- ③ **問題解決能力**
- ④ **自主的な学習態度と探求心**

演習 図形のバリエーション

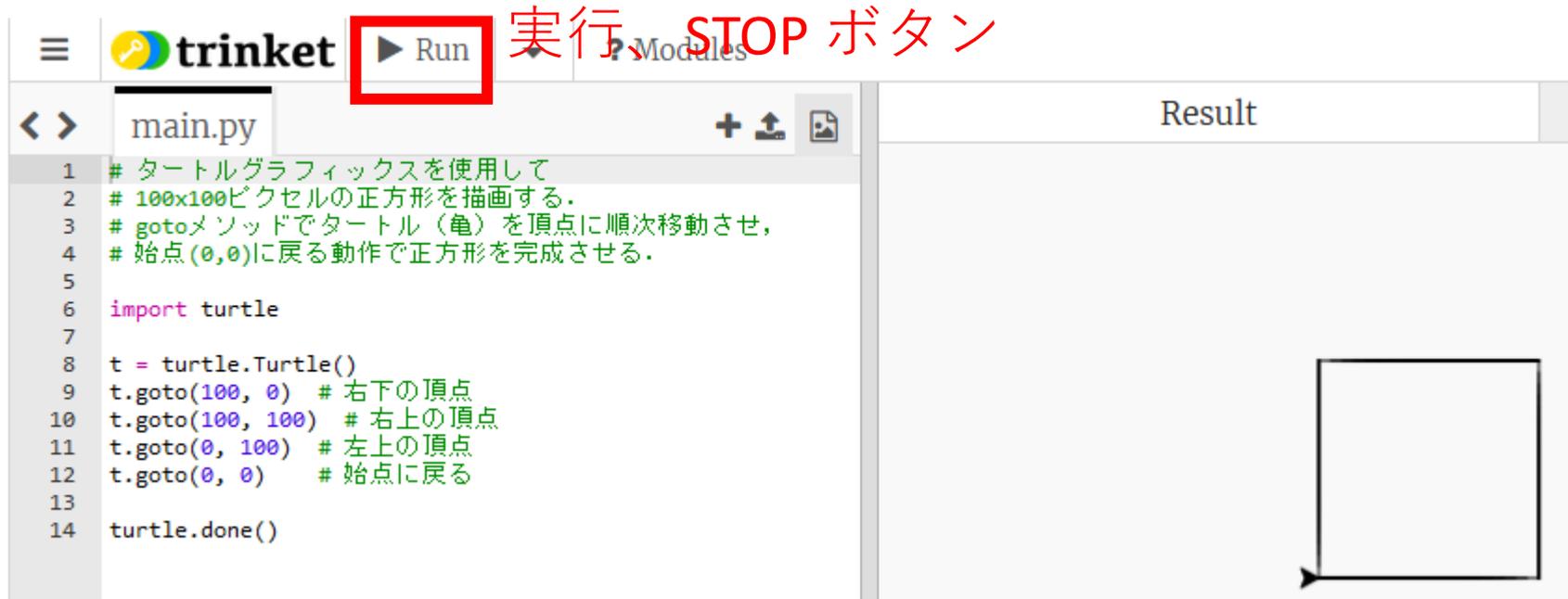


① プログラムを試す. trinket の次のページを開く

<https://trinket.io/python/035810ce8d49>

・ goto の組み合わせによって, 独自の図形を描く

② 実行結果が, 次のように表示されることを確認



```
1 # タートルグラフィックスを使用して
2 # 100x100ピクセルの正方形を描画する.
3 # gotoメソッドでタートル(亀)を頂点に順次移動させ,
4 # 始点(0,0)に戻る動作で正方形を完成させる.
5
6 import turtle
7
8 t = turtle.Turtle()
9 t.goto(100, 0) # 右下の頂点
10 t.goto(100, 100) # 右上の頂点
11 t.goto(0, 100) # 左上の頂点
12 t.goto(0, 0) # 始点に戻る
13
14 turtle.done()
```

・ 実行が開始しないときは、「**実行ボタン**」で**実行**

・ プログラムを**書き替えて再度実行**することも可能

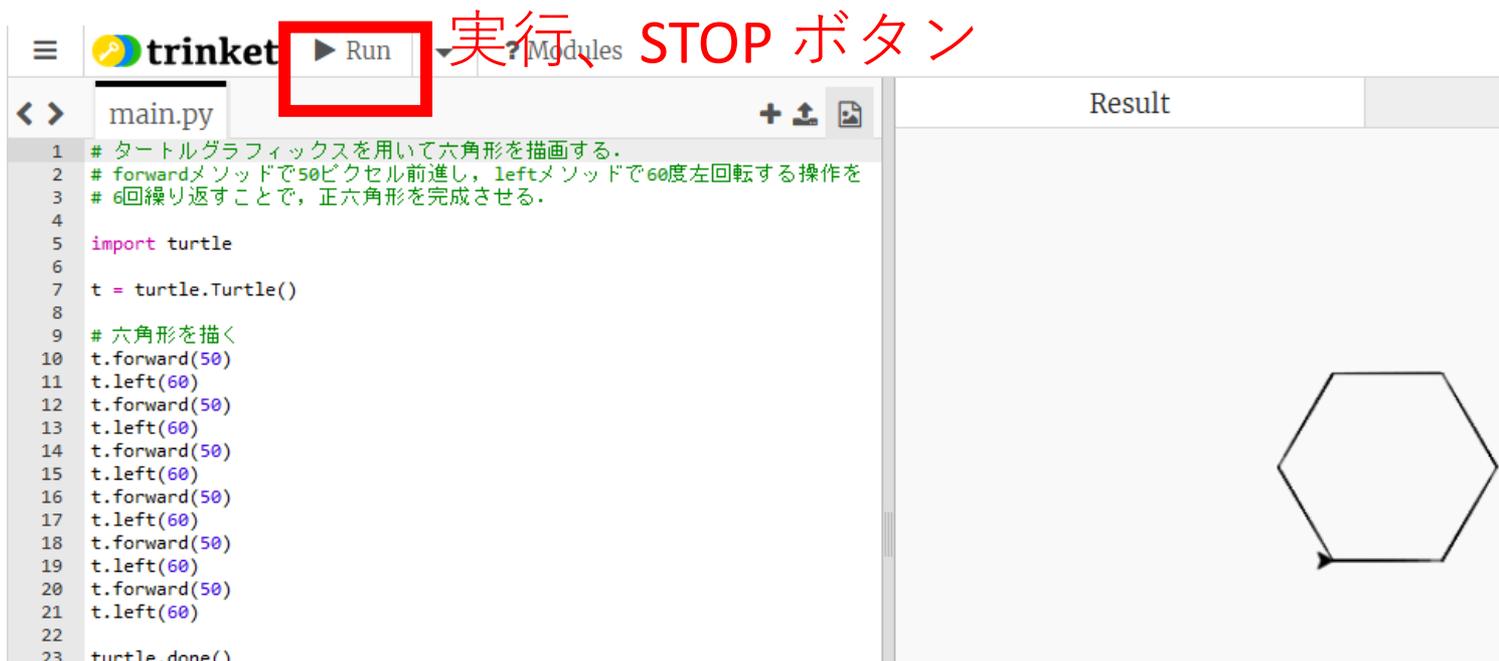
③ プログラムを試す. trinket の次のページを開く



<https://trinket.io/python/ddb861147133>

- forward, left のような新しいメソッドを探求

④ 実行結果が, 次のように表示されることを確認



```
1 # タートルグラフィックスを用いて六角形を描画する.  
2 # forwardメソッドで50ピクセル前進し, leftメソッドで60度左回転する操作を  
3 # 6回繰り返すことで, 正六角形を完成させる.  
4  
5 import turtle  
6  
7 t = turtle.Turtle()  
8  
9 # 六角形を描く  
10 t.forward(50)  
11 t.left(60)  
12 t.forward(50)  
13 t.left(60)  
14 t.forward(50)  
15 t.left(60)  
16 t.forward(50)  
17 t.left(60)  
18 t.forward(50)  
19 t.left(60)  
20 t.forward(50)  
21 t.left(60)  
22  
23 turtle.done()
```

- 実行が開始しないときは、「**実行ボタン**」で**実行**
- プログラムを書き替えて再度実行することも可能

自己主導型学習の4つの観点



① 具体的トピックスでの学習

Pythonでグラフィックスを描く．基本的な図形描画の技術を習得

② Pythonの基本を押さえる

オブジェクト，メソッド，引数の概念を理解

③ 発想力，創造力を育む

独自のデザインし実装する

④ 自主性，自己研鑽力を高める

新しい機能を自ら調べ，理解し，試す

演習 プログラミングの楽しさと 達成感



オブジェクトの生成, 座標指定による移動などの基本操作を学ぶ



① <https://trinket.io/python/5366def2f4>

```
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)
```

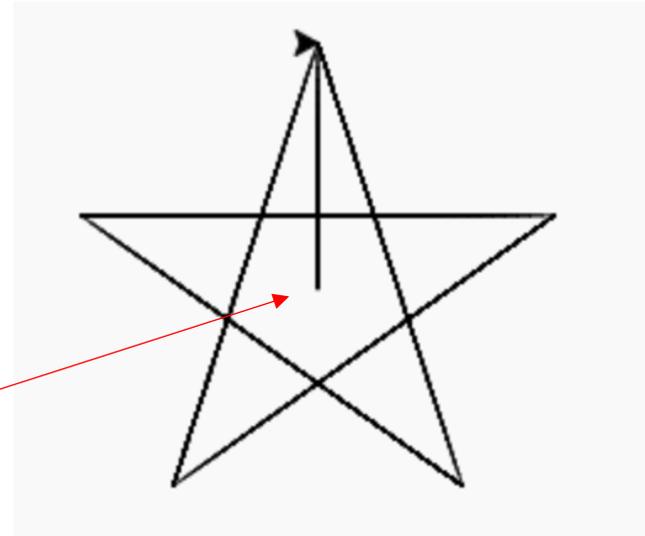
ライブラリのインポート
オブジェクト生成。t へのセット。

移動

実行結果

**最初の位置は
(0, 0)**

(0, 0)



色の指定方法, 繰り返し処理, 円の描画方法など,
より発展的なプログラミング



② <https://trinket.io/python/f8cd554693>

```
import turtle  
t = turtle.Turtle()  
colors = ["red", "green", "blue"]  
for i in range(3):  
    t.color(colors[i])  
    t.circle(30)  
    t.forward(50)
```

ライブラリのインポート

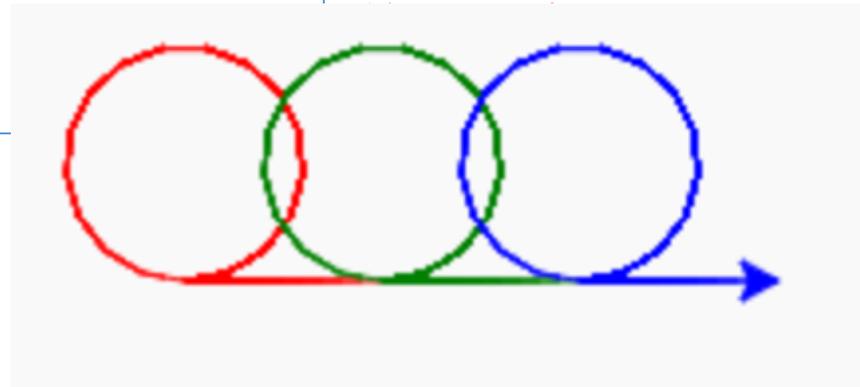
オブジェクト生成。t へのセット。

色は、赤、緑、青

色を変える

半径 30 の円

実行結果



プログラミングを学ぶ意義と達成感

- 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：変数と式，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においても，**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出来るもの。
- ⑤将来は，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。