

# 1. プログラミングの基礎と Python 言語入門：創造的なデジ タルスキル

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



# 1-1. プログラミングの基本 と意義

# コンピュータとプログラムの関係



- **コンピュータ**は、**プログラム**に従って動作
- **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

この関係が、コンピュータシステムの基礎

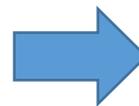
```
x1="100%" y1="0%" x2="0%" y2="100%"
color="#06101F" offset="0%" />
color="#1D304B" offset="100%" />
height="96" viewBox="0 0 96 96"
gradient x1="87.565%" y1="15.878%"
stop stop-color="#FFF" stop-opacity="1"
stop stop-color="#FFF" offset="100%"
rGradient>
x="-500%" y="-500%" width="1000%"
offset dy="16" in="SourceAlpha"
ussianBlur stdDeviation="24"
orMatrix values="0 0 0 0 0 0 0 0 0 0"
```

# プログラムの本質



- **プログラム**を設計し作成するプロセス（プログラミング）は、**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
- 複雑な作業も**自動化**し、効率化することが可能

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

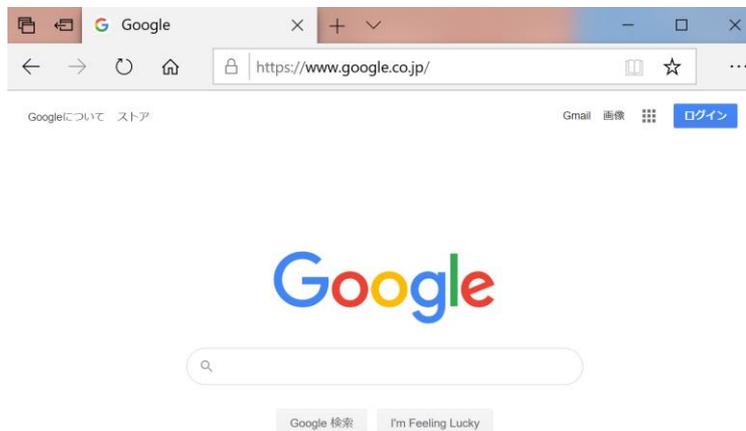


```
216.0
432.0
324.0
```

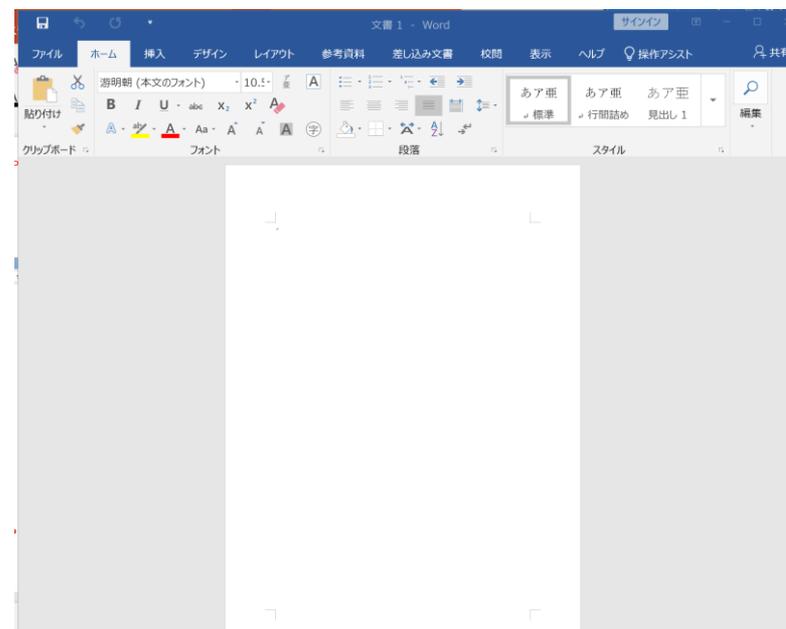
**Python 言語のプログラム**

**プログラムの**  
実行結果

# ① プログラムによるアプリケーションの実現



Web ブラウザ



ワープロ  
(マイクロソフト・ワード)

**プログラム**が動作し、**アプリケーション**の機能を実現

## ② プログラムによるコンピュータの制御

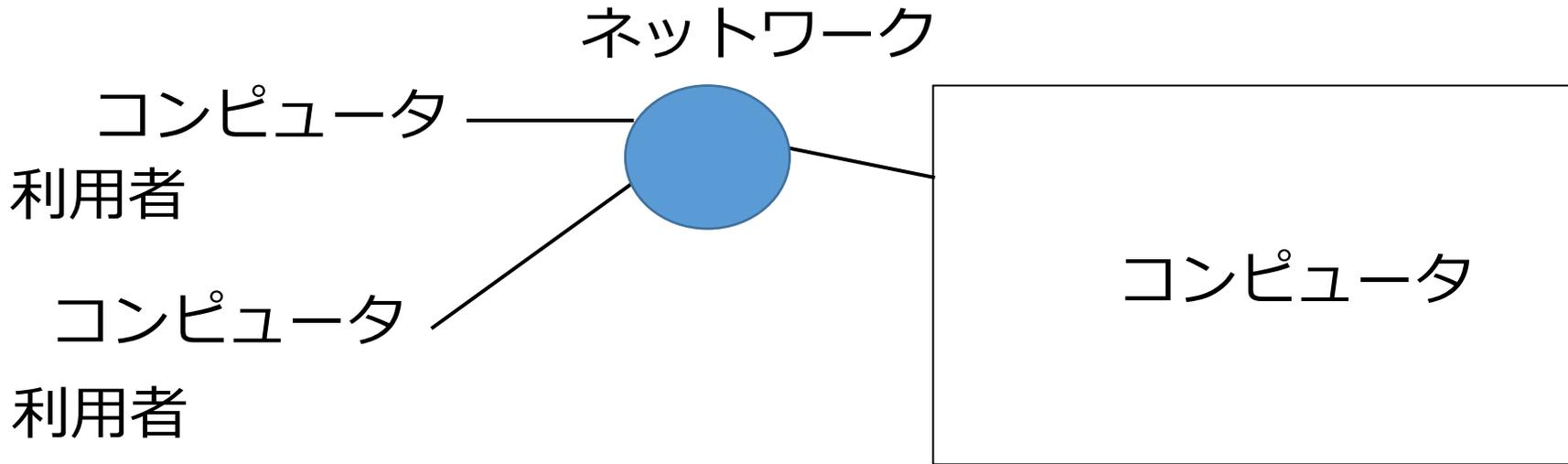
```
In [7]: from keras.models import Sequential
...: model = Sequential()
...: from keras.layers import Dense, Activation
...:
...: model.add(Dense(units=64, input_dim=len(x_train[0])))
...: model.add(Activation('relu'))
...: model.add(Dense(units=max(set(y_train)) - min(set(y_train)) + 1))
...: model.add(Activation('softmax'))
...: model.compile(loss='sparse_categorical_crossentropy',
...:               optimizer='sgd',
...:               metrics=['accuracy'])
...: model.fit(x_train, y_train, epochs=200)
...: score=model.evaluate(x_test, y_test, batch_size=1)
...: print(score)
...: model.predict(x_test)
...: model.summary()

Epoch 1/200
3/3 [=====] - 0s 5ms/step - loss: 1.0583 - accuracy:
0.3200
Epoch 2/200
3/3 [=====] - 0s 0s/step - loss: 1.0530 - accuracy:
0.3200
Epoch 3/200
3/3 [=====] - 0s 0s/step - loss: 1.0485 - accuracy:
0.3200
```

Python 言語を使って  
ニューラルネットワーク  
を作成. AIシステムを構築

**プログラム**は, **コンピュータ**の動作を細かくコントロール

### ③ プログラムによるコンピュータ間連携の実現



サーバは、サービスを提供する  
ITシステム

**コンピュータ**同士の接続でも**プログラム**が必要。

# プログラミングの4つの基本的特徴

- ① **プログラム**の内容によって、**コンピュータ**はさまざまな作業を実行できる
- ② **プログラム**を利用することで、多くの作業を自動化できる
- ③ **プログラム**で行った作業をいつでも再現できる。
- ④ **プログラム**は柔軟性があり、変更により、プログラムの動作を簡単に調整できる

# プログラミングの魅力



- 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



# プログラミングの達成感



- **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**

# プログラミングの活用領域



- **プログラム**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
- **プログラム**を活用することで、複雑な作業の自動化が可能である

# 1-2. Python言語の特徴とプログラミングの可能性

# Python 言語の概要

- **Python** は多くの人々に利用されている**プログラミング言語**の1つ
- **読みやすさ, 書きやすさ, 幅広い応用範囲**が特徴

```
from keras.models import Sequential
: model = Sequential()
.: from keras.layers import Dense, Ac
.:
.: model.add(Dense(units=64, input_di
.: model.add(Activation('relu'))
.: model.add(Dense(units=max(set(y_tr
.: model.add(Activation('softmax'))
.: model.compile(loss='sparse_categor
.: optimizer='sgd',
.: metrics=['accuracy'])
.: model.fit(x_train, y_train, epochs
.: score=model.evaluate(x_test, y_tes
.: print(score)
.: model.predict(x_test)
.: model.summary()
epoch 1/200
3 [=====] - 0s
3200
epoch 2/200
3 [=====] - 0s
3200
epoch 3/200
[=====] - 0s
0
```

# Python 言語の特徴



- Python は、**直感的で読みやすい**文法構造
- **シンプル**なスクリプトから、**高度なプログラム**まで、さまざまな規模の開発に対応できる**柔軟性**を備える

# Python 言語の主な利点



## 文法のシンプルさ

直観的で読みやすい

例 `print` で簡単に出力

`if` や `else` で条件分岐

`for` や `while` で繰り返し（ループ）

字下げでブロック構造を示す

## 拡張性

多岐にわたる分野で利用が可能

例 関数やクラスを定義する `def` や `class`

継承やオブジェクトの属性名と値を操作する `super` や `vars`

## 柔軟性

シンプルなスクリプトも、高度なプログラムも作成可能

# 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 showing a file named "main.py" with the following Python code:

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

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

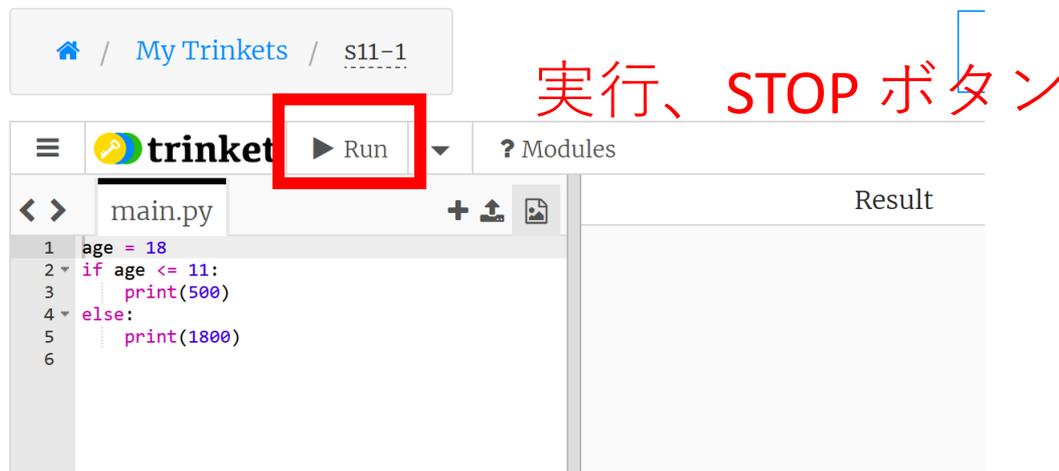


# Trinket の基本操作

- 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. Trinket による Python プ ログラム実行

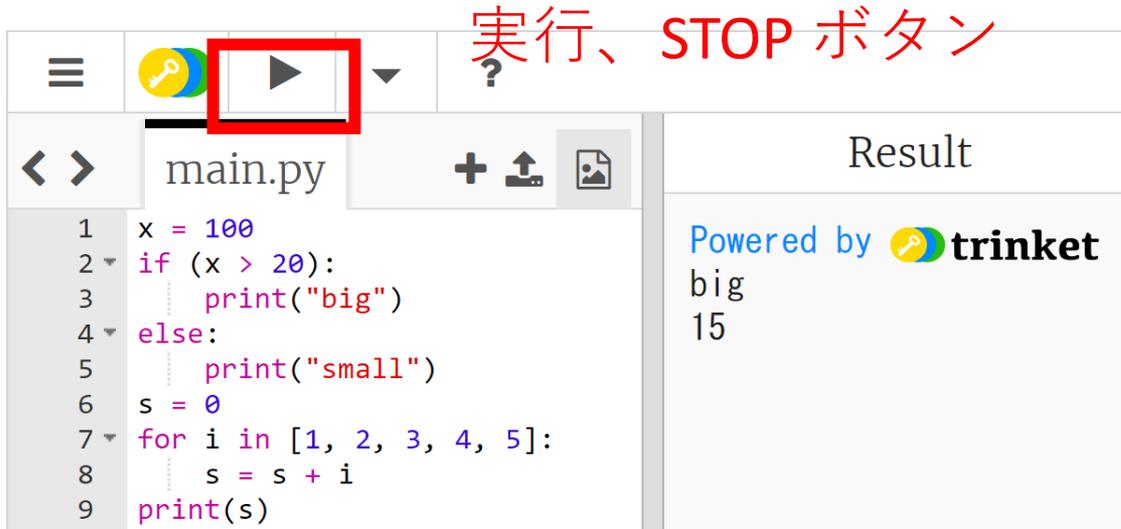


① trinket の次のページを開く

<https://trinket.io/python/6c652f1c2f>

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 x = 100
2 if (x > 20):
3     print("big")
4 else:
5     print("small")
6 s = 0
7 for i in [1, 2, 3, 4, 5]:
8     s = s + i
9 print(s)
```

Result

Powered by  trinket

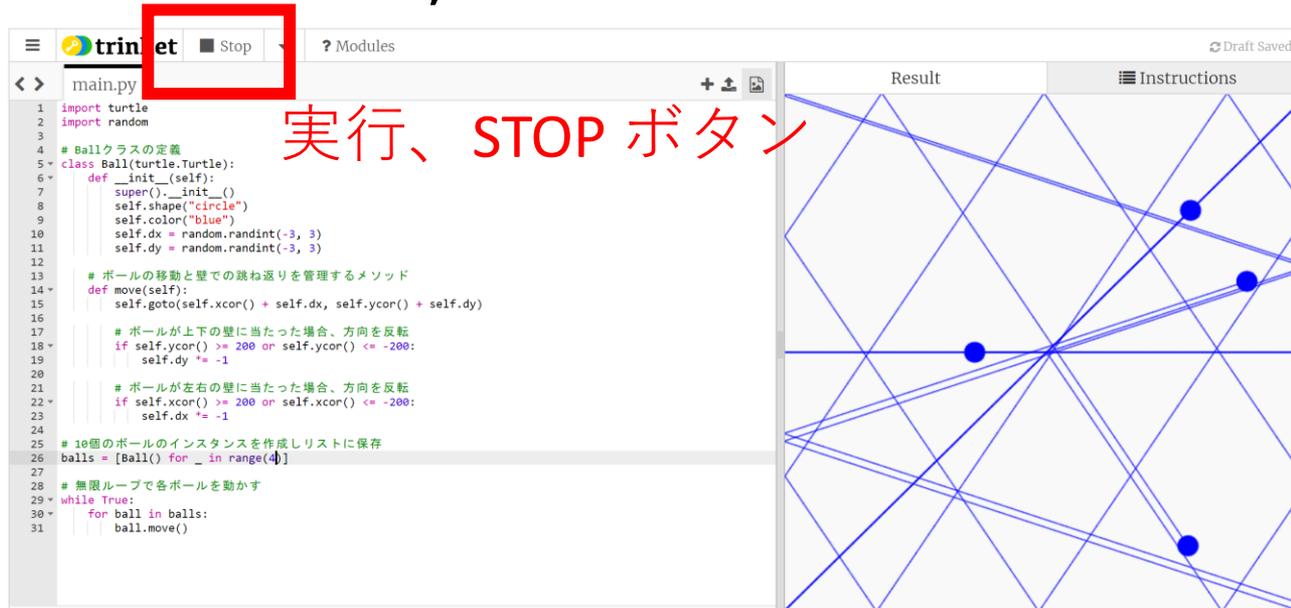
big  
15

- 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 trinket の次のページを開く

<https://trinket.io/python/94d1563844>

### ④ 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. On the left, a code editor displays Python code for a ball simulation. A red box highlights the 'Stop' button in the top toolbar. The right side of the interface is split into 'Result' and 'Instructions' tabs. The 'Result' tab shows a 2D coordinate system with a grid of blue lines and four blue circles representing balls. The text '実行、STOP ボタン' is overlaid on the code editor.

```
1 import turtle
2 import random
3
4 # Ballクラスの定義
5 class Ball(turtle.Turtle):
6     def __init__(self):
7         super().__init__()
8         self.shape("circle")
9         self.color("blue")
10        self.dx = random.randint(-3, 3)
11        self.dy = random.randint(-3, 3)
12
13 # ボールの移動と壁での跳ね返りを管理するメソッド
14 def move(self):
15     self.goto(self.xcor() + self.dx, self.ycor() + self.dy)
16
17 # ボールが上下の壁に当たった場合、方向を反転
18 if self.ycor() >= 200 or self.ycor() <= -200:
19     self.dy *= -1
20
21 # ボールが左右の壁に当たった場合、方向を反転
22 if self.xcor() >= 200 or self.xcor() <= -200:
23     self.dx *= -1
24
25 # 10個のボールのインスタンスを作成しリストに保存
26 balls = [Ball() for _ in range(4)]
27
28 # 無限ループで各ボールを動かす
29 while True:
30     for ball in balls:
31         ball.move()
```

ボールが  
壁に当たったら  
反射する。

- 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

演習.  
基本的な計算から高度な  
数学処理まで



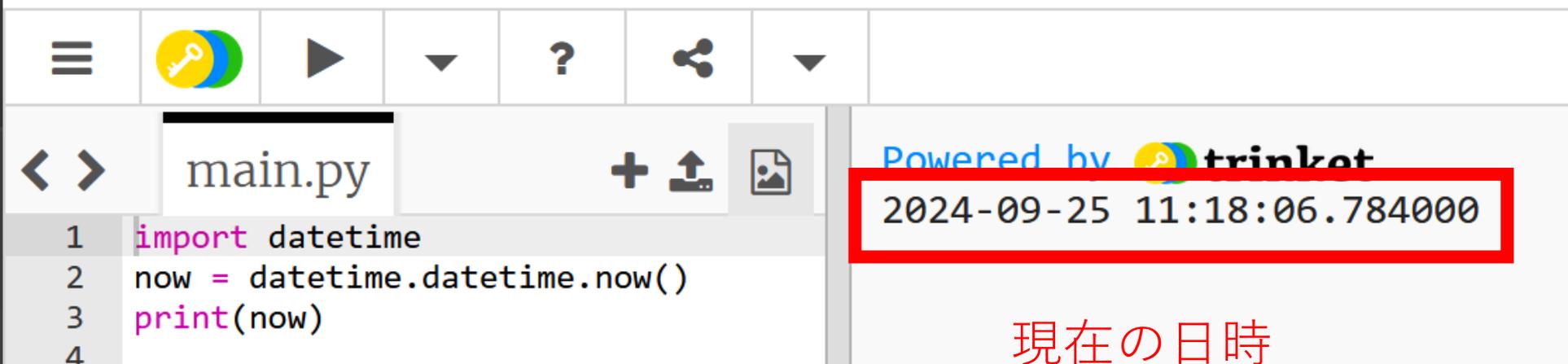
# オペレーティングシステム（コンピュータ）のタイマー を利用した**現在日時**の表示

① trinket の次のページを開く

<https://trinket.io/python/2b804ab19a>

② 実行結果が、次のように表示されることを確認

```
import datetime
now = datetime.datetime.now()
print(now)
```



main.py

```
1 import datetime
2 now = datetime.datetime.now()
3 print(now)
4
```

Powered by  trinket

2024-09-25 11:18:06.784000

現在の日時

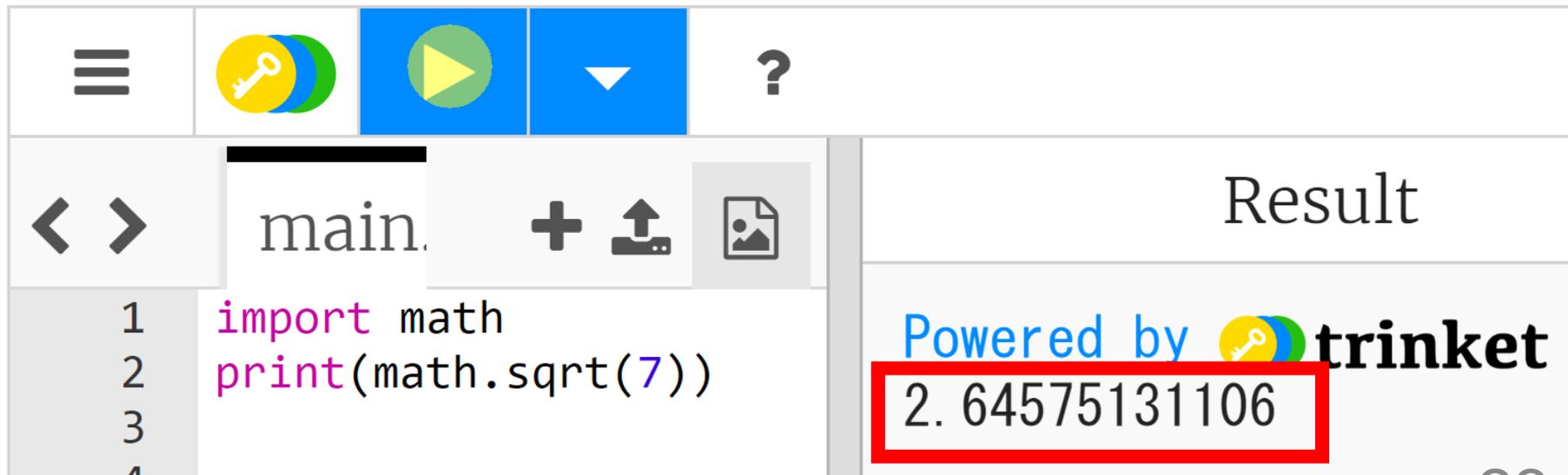
平方根：面積が **7** の正方形の一辺の長さ

③ trinket の次のページを開く

<https://trinket.io/python/597e5771ff>

④ 実行結果が、次のように表示されることを確認

```
import math
print(math.sqrt(7))
```



The screenshot shows the Trinket Python IDE interface. The top bar contains a menu icon, a key icon, a play button, a dropdown arrow, and a question mark. Below the top bar, there are navigation arrows, a file name 'main.py', and icons for adding files, uploading, and images. The code editor shows the following code:

```
1 import math
2 print(math.sqrt(7))
3
4
```

The output area on the right is titled 'Result' and displays the text 'Powered by  trinket' and the numerical result '2.64575131106', which is highlighted with a red box.

円周率：半径 **3** の円の面積は？



⑤ trinket の次のページを開く

<https://trinket.io/python/4e3559f879>

⑥ 実行結果が、次のように表示されることを確認

```
import math
print(3 * 3 * math.pi)
```

A screenshot of the Trinket Python IDE interface. The top toolbar contains icons for a menu, a key (representing a keychain), a play button, a dropdown arrow, and a question mark. Below the toolbar, the file name "main.py" is displayed. The code editor shows the following Python code:

```
1 import math
2 print(3 * 3 * math.pi)
3
4
```

To the right of the code editor, the "Result" section displays the output of the code execution: "Powered by  trinket" followed by the numerical result "28.2743338823". The numerical result is enclosed in a red rectangular box.

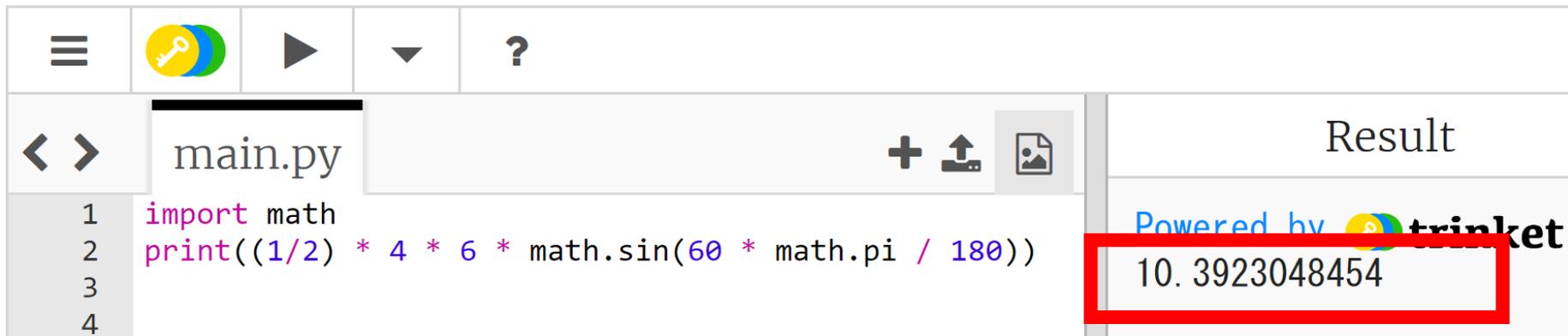
三角関数：三角形の2辺の長さが、**4**と**6**で、その間の角度が**60**度のとき、面積は  $(1/2) \times 4 \times 6 \times \sin(60)$

⑦ trinket の次のページを開く

<https://trinket.io/python/bdcce27488>

⑧ 実行結果が、次のように表示されることを確認

```
import math
print((1/2) * 4 * 6 * math.sin(60 * math.pi / 180))
```



The screenshot shows the Trinket.io Python editor interface. The code editor on the left contains the following Python code:

```
1 import math
2 print((1/2) * 4 * 6 * math.sin(60 * math.pi / 180))
3
4
```

The output panel on the right, titled "Result", displays the result of the execution: `10.3923048454`. The result is highlighted with a red box. The interface also shows a "Powered by trinket" logo at the bottom of the output panel.

# プログラミングの意義と可能性



- **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技術である
  - **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
- 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
- 論理的思考力の向上
- 問題解決能力の育成
- デジタル社会での**必須スキル**

# プログラミングの応用分野



- **Web開発**

フロントエンド (HTML, CSS, JavaScript) , バックエンド (Python, Django, Flask)

- **データ分析**

ビッグデータ処理, 統計分析, データビジュアライゼーション (データの可視化)

- **人工知能**

自然言語処理, コンピュータビジョン (画像認識技術) , 予測モデリング

- **ゲーム開発**

2Dゲーム, 3Dゲーム, モバイルゲーム

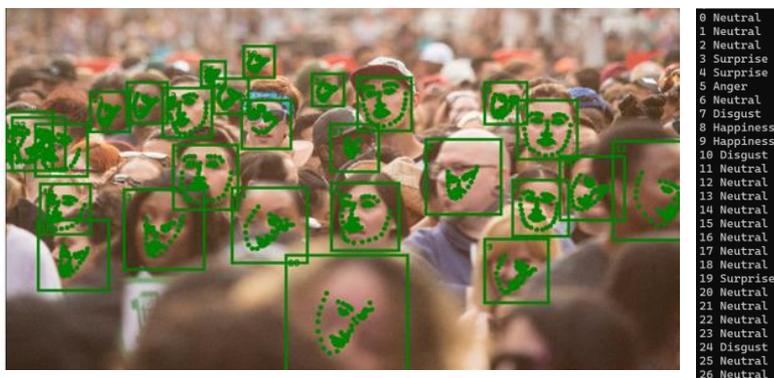
- **IoT (Internet of Things)**

センサーデータの収集と分析, スマートホームシステム

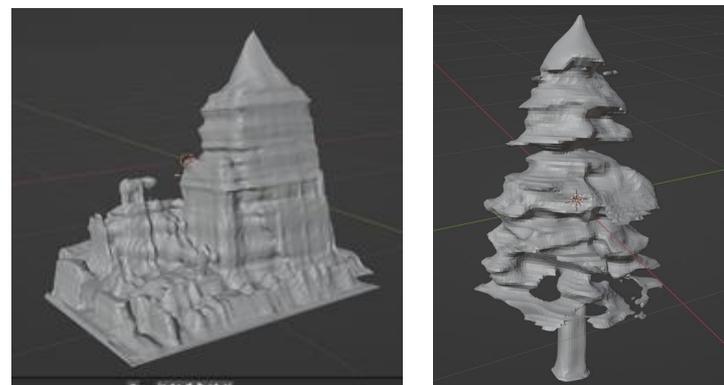
- **サイバーセキュリティ**

ネットワークセキュリティ, 暗号化技術

**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。



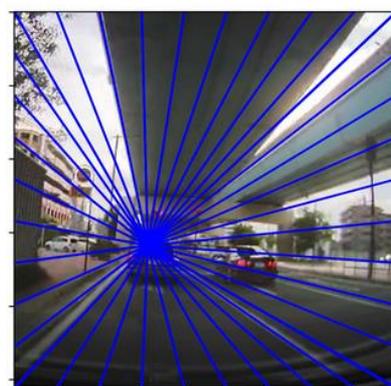
顔情報処理



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 28

# Python プログラムのパソコン上での実行



Python プログラムはオンライン実行（例：Trinket）のほか、パソコンでも実行可能。（パソコンでの実行の場合には、Python 処理系のインストールが必要）

## ① Python プログラムのファイル保存

```
x = 100
if (x > 20):
    print("big")
else:
    print("small")
s = 0
for i in [1, 2, 3, 4, 5]:
    s = s + i
print(s)
```

作成した **Python プログラム** の **ソースコード** を、例えば「foo.py」という名前の **ファイル** に保存

## ② Python プログラムの実行

```
kaneko@www:/tmp$ python foo.py
big
15
```

プログラムを実行するには、シェル（例えば、Windows の場合はコマンドプロンプト）を開き、「python foo.py」のようなコマンドで実行

# まとめ



- Pythonは直感的で読みやすい文法を持つプログラミング言語.
- 文法のシンプルさ, 拡張性, 柔軟性が特徴的.
- 基本的な計算から高度なプログラミングまで幅広く対応可能.
- Web開発, データ分析, AI, ゲーム開発など応用分野が多岐にわたる.
- プログラミングは人間の能力を増幅し, 創造的な活動を支援する重要な活動である.