# cs-4. プログラミング入門

（コンピューターサイエンス）

URL: https://www.kkaneko.jp/cc/cs/index.html

金子邦彦

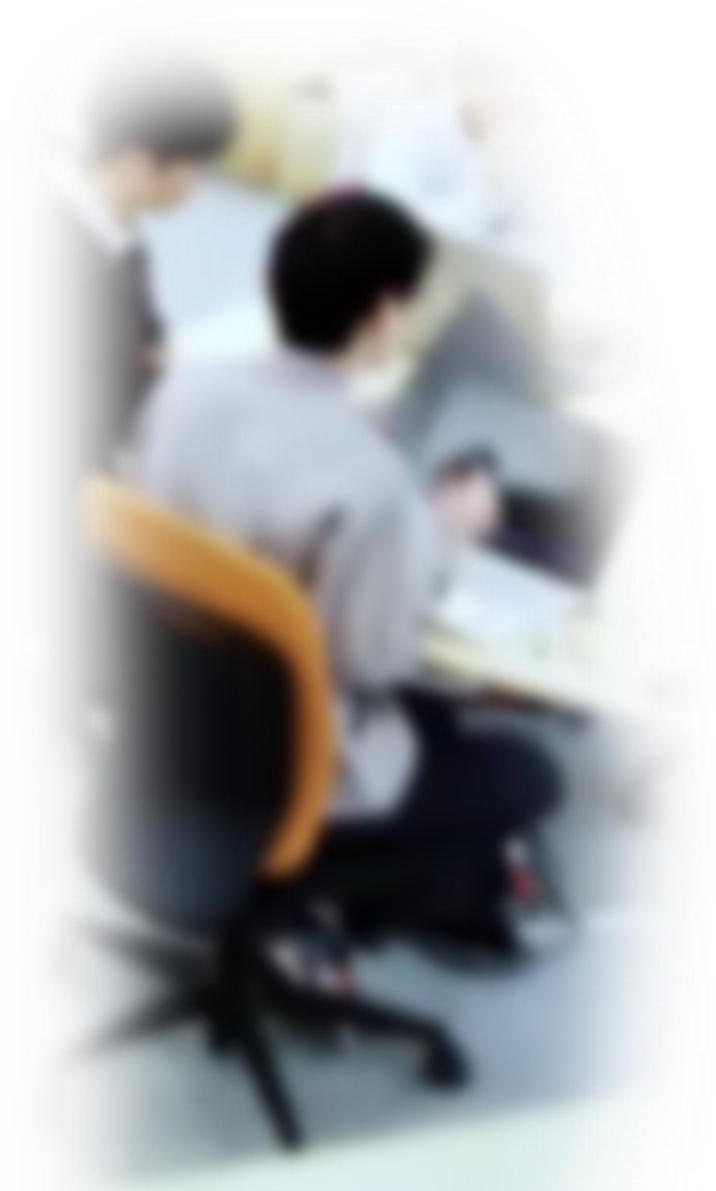
CC BY NC SA

① IT 技術の可能性

② コンピュータを活用した問題解決

③ 批判的思考

④ プログラミング言語の多様性



# アウトライン

1. プログラミングの基本と意義
2. Python言語の特徴とプログラミングの可能性
3. コンピュータの限界と精度の理解
4. さまざまなプログラミング言語

# 4-1. プログラミングの基本と意義

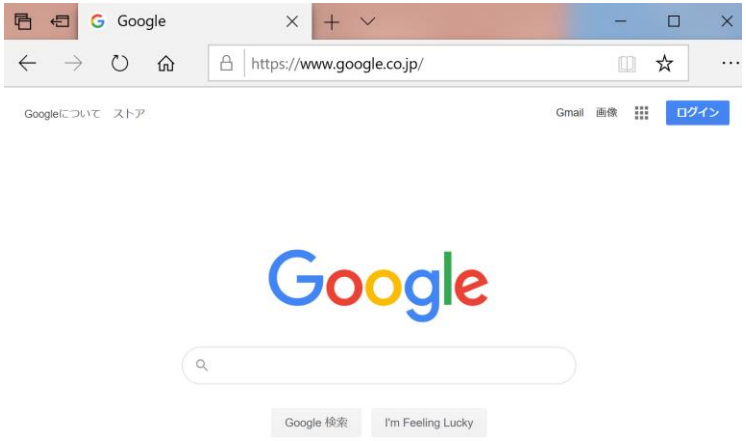
# コンピュータとプログラム

- **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

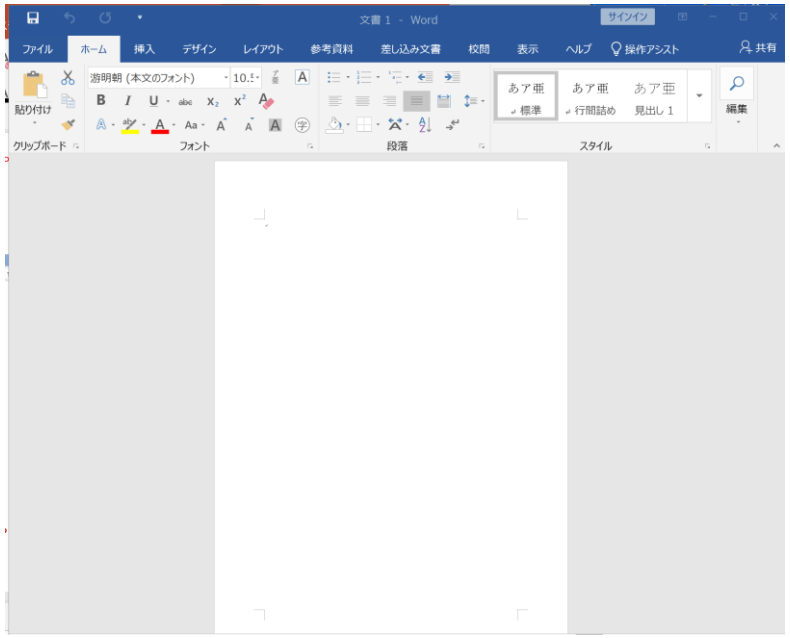
# プログラミング

- **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**
- **アイデアを形にできる**ことが，**プログラミング**の魅力

# ① プログラムとアプリケーション




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

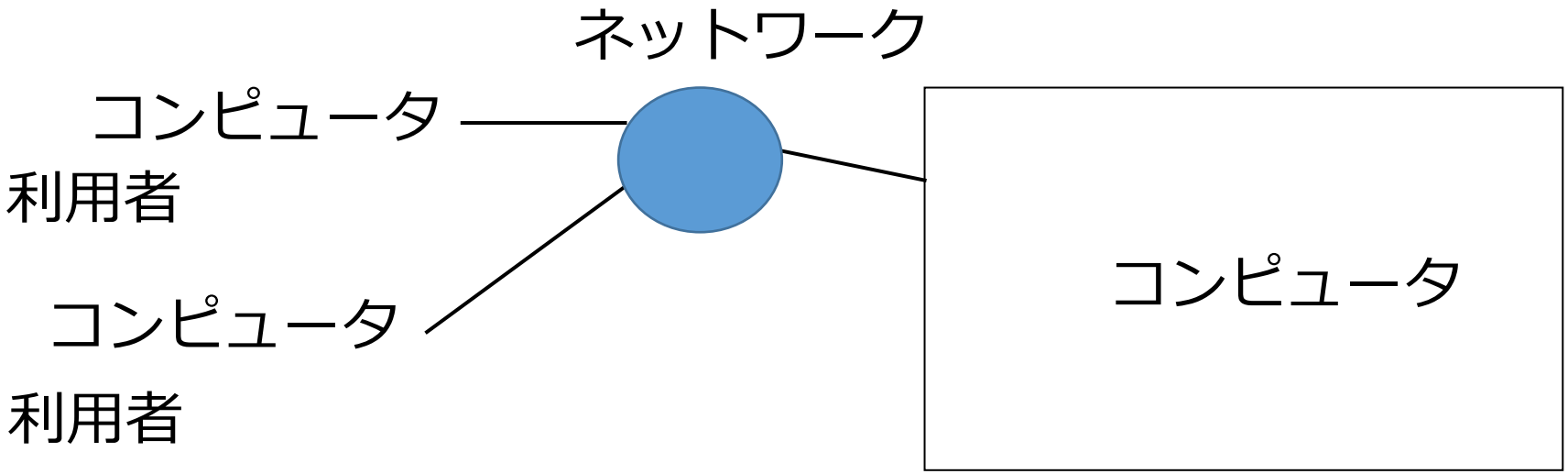
# ② プログラムは，コンピュータの動作をコントロール

```
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ール

# ③ プログラムは，コンピュータ間の連携にも役立つ




サーバは，サービスを提供するITシステム

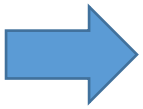
**コンピュータ**同士の接続でも**プログラム**が必要.

# プログラミングの目的

- **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)
```

**Python プログラム**の
**ソースコード**



```
216.0
432.0
324.0
```

**プログラム**の
実行結果

# プログラミングの利点

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**は柔軟性がある.変更により，**プログラムの動作を簡単に調整できる**

# プログラミングの基本と意義まとめ

- コンピュータは，**プログラムの指示に従って動作**するものである.
- プログラミングは，創造的な活動として**プログラムを設計・作成**することである.
- プログラムは，**コンピュータの動作を詳細にコントロール**する.
- プログラムを活用することで，**複雑な作業も自動化**が可能である.
- プログラムは，**作業の再現性**と**柔軟な変更が可能**という利点を持つ.

# 4-2. Python言語の特徴とプログラミングの可能性

# Python

- **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 Sequentia
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()
```

```
poch 1/200
'3 [==============================] - 0s
 3200
 och 2/200
 3 [==============================] - 0s
 200
  ch 3/200
   [==============================] - 0s
  0
```

# Python 言語が広く使用されている理由

**文法のシンプルさ**

- **Python** は，**直感的で読みやすい**文法
- 例えば、**print** で簡単に**出力**できる、**if** や **else** で**条件分岐**、**for** や **while** で**繰り返し（ループ）**

**拡張性**

- **多岐にわたる分野で利用**が可能
- 例えば、**関数やクラスを定義**するための **def** や **class**、**継承**や**オブジェクトの属性名と値**を操作するための **super** や **vars** などがある。

**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
- **オブジェクト指向**の機能を持ち、**__init__** や **self** のようなキーワードを使用して**クラス**を利用できる。

# trinket

- **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）

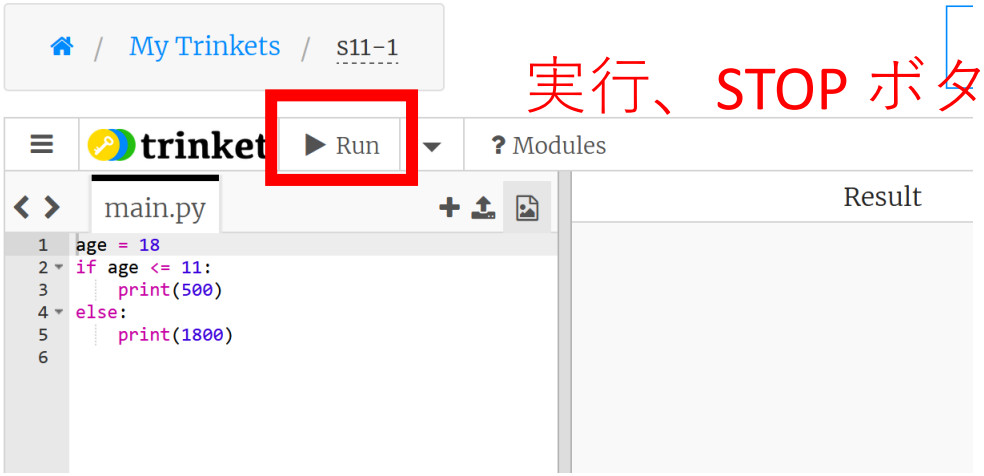
- **Python の標準機能**を登載、その他、次の外部ライブラリがインストール済み

matplotlib.pyplot, numpy, processing, pygal

# 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/0fd59392c8

のように、違うプログラムには違う URL が割り当てられる




- 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

# 演習.
# Python プログラムの実行

① trinket の次のページを開く

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

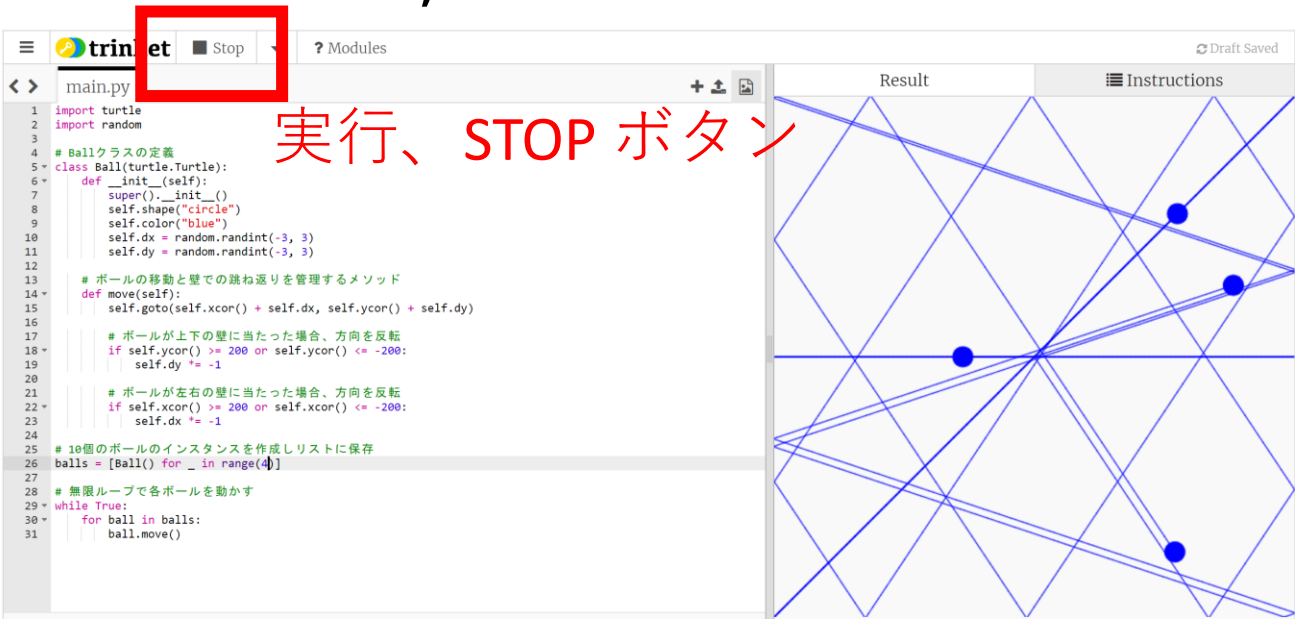
② 実行結果が，次のように表示されることを確認

- 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 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




```
import turtle
import random

# Ballクラスの定義
class Ball(turtle.Turtle):
    def __init__(self):
        super().__init__()
        self.shape("circle")
        self.color("blue")
        self.dx = random.randint(-3, 3)
        self.dy = random.randint(-3, 3)

    # ボールの移動と壁での跳ね返りを管理するメソッド
    def move(self):
        self.goto(self.xcor() + self.dx, self.ycor() + self.dy)

        # ボールが上下の壁に当たった場合、方向を反転
        if self.ycor() >= 200 or self.ycor() <= -200:
            self.dy *= -1

        # ボールが左右の壁に当たった場合、方向を反転
        if self.xcor() >= 200 or self.xcor() <= -200:
            self.dx *= -1

# 10個のボールのインスタンスを作成しリストに保存
balls = [Ball() for _ in range(4)]

# 無限ループで各ボールを動かす
while True:
    for ball in balls:
        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**
- ソースコードを**書き替えて再度実行**することも可能

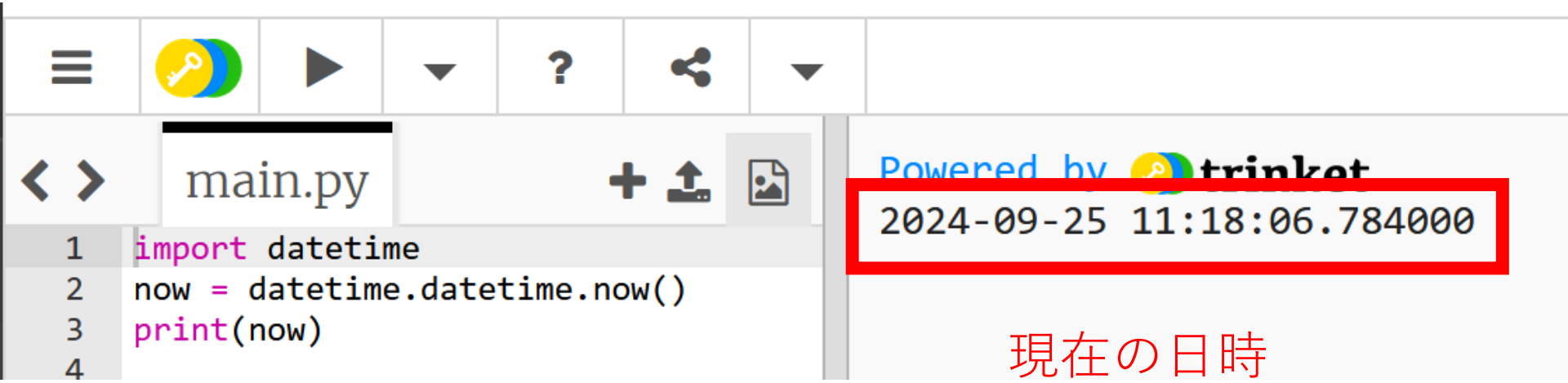
# 演習.
# 簡単なプログラムでも
# さまざまなことが可能

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．**現在の日時が表示される**

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)
```

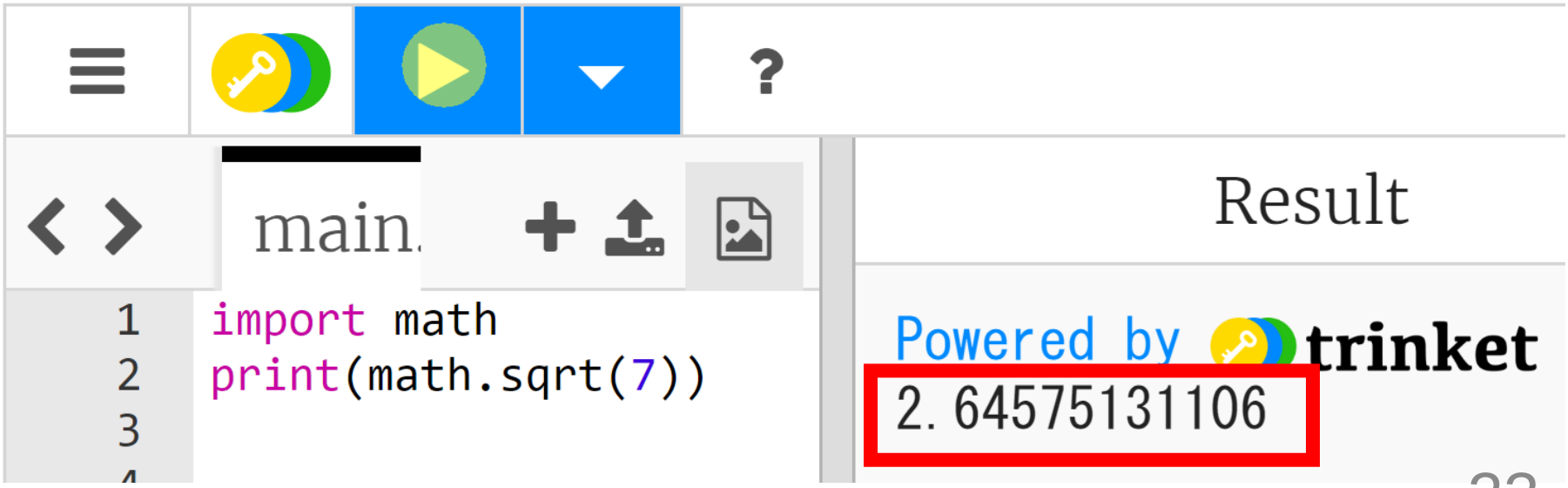


# 面積が 7 の正方形の一辺の長さ

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))
```

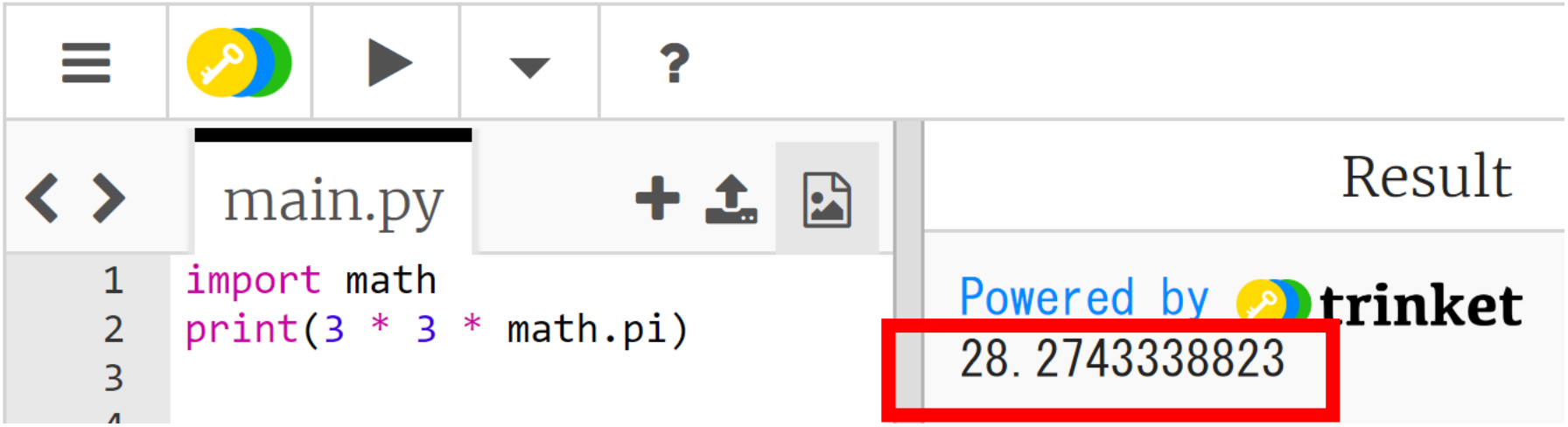


半径 **3** の円の面積は？

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)
```

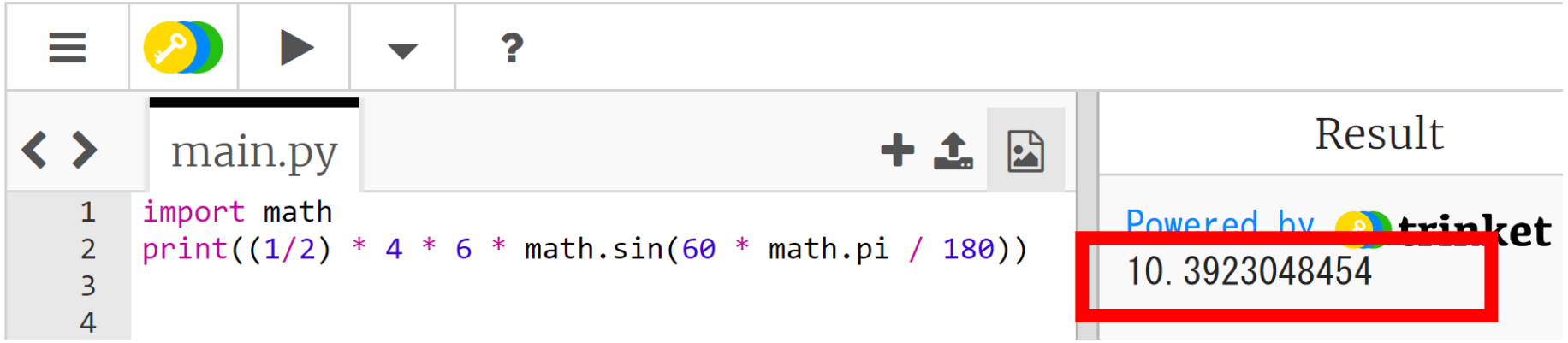


三角形の２辺の長さが，**4**と**6**で，その間の角度が**60**度のとき，面積は (1/2) × **4** × **6** × sin(**60**)

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))
```



# プログラミングの可能性

- **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
- 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

- **人工知能**

自然言語処理，コンピュータビジョン，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

# Python 言語の特徴とプログラミングの可能性まとめ

- **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**が特徴的である.
- Pythonは**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である.
- **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.

# 4-3. コンピュータの限界と精度の理解

# プログラミングにおける注意点 ①

1. **コンピュータ**にも，**できないことがある**．全ての問題を解決できるわけではないことを理解しよう，
2. コンピュータを使用するからといって，**計算が常に完全に正確であるとは限らない．**特に複雑な計算を行う場合には，精度に注意が必要．
3. 人間が**プログラム**を作る際には，**書き間違い，勘違い，思い込みなどによるミスが起こり得る**．

# プログラミングにおける注意点 ②

4. ミスがあり得るため，「**プログラムが期待通りに動いているか**」を確認する**テストは非常に重要**.
5. **ミスの回避**のため，**抽象化、モジュール，標準ライブラリの活用**などの様々な手段を知っておく.
6. 性能や精度を追求し，問題を解決するために，**既存のアルゴリズム**を知っておく

# コンピュータでの数値の扱い

- コンピュータは、整数だけでなく、**小数点以下の値を含む数値（浮動小数点数）を扱う**ことができる。
- 浮動小数点数は、**通常**、コンピュータが、**10進数で約15-16桁の精度まで数値を保持**できるものである。
- **有限の精度**であるため、**この範囲を超える数値**（10進数で17桁以上の数値）を計算しようとする**微小な誤差**が発生する。
- 精密な計算を行う場合などは、精度について理解しておくことが重要となる。

# コンピュータによる「１÷３」の計算

コンピュータを使って**「1 ÷ 3」を計算**してみるとどうなるでしょうか？

- コンピュータは「**0.33333333333333333**」などと表示
- しかし，**「無限に続く」数値を表現することはできない**
- つまり、「1 ÷ 3」の**正確な値を計算できない**

コンピュータが表示する結果には**小さな誤差が含まれている（精度に限界がある）**と理解してください.

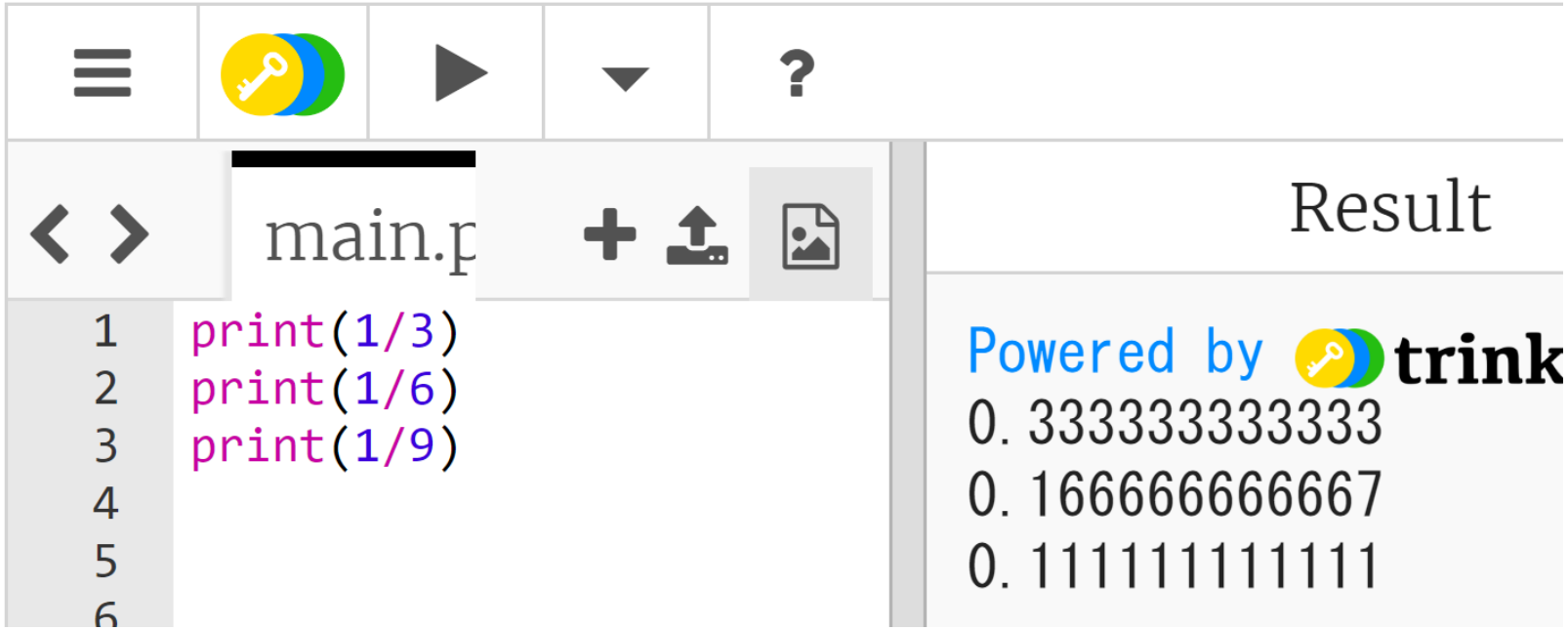
コンピュータを使った計算に注意してください.

# 演習．計算誤差

1/3, 1/6, 1/9 を計算

① trinket の次のページを開く
https://trinket.io/python/8d555705c1

② 実行結果が，次のように表示されることを確認




計算誤差がある

0.333333333333 の 3 倍を計算

③ trinket の次のページを開く
https://trinket.io/python/8a180f7d80

④ 実行結果が，次のように表示されることを確認

計算誤差がある

# コンピュータ計算と誤差

- 「**コンピュータを使えば計算は完璧に正確**」- **この思い込みは避けましょう**。

- 例えば，「1 ÷ 3」の計算結果は，完全な精度では表示できません．**少ないながらも誤差が含まれます**.

- しかしこの微小な誤差は、多くの場合、**私たちの作業には十分な精度**
- 少しの誤差を許容すれば、**多くの計算が効率的に処理できる**という考え方も.

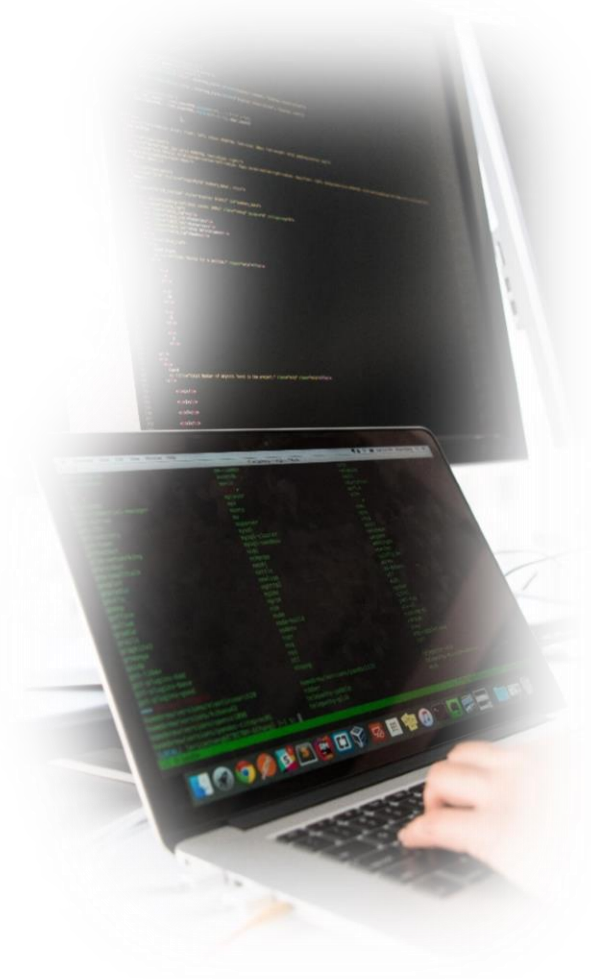
# コンピュータの限界と精度の理解

- 人間のプログラム作成時には，**書き間違い**，**勘違い**，**思い込みによるミス**が起こり得る．
- **プログラムが期待通りに動いているかを確認するテスト**は非常に重要である．
- **ミスの回避**には，様々な知識・スキル（抽象化，モジュール，標準ライブラリの活用など）が重要である．
- **問題解決**のために，既存のアルゴリズムを知っておく必要がある．
- **浮動小数点数**は，**10進数で約15-16桁の精度**まで数値を保持できる．10進数で17桁以上の数値計算では，**微小な誤差が発生**する．

# 4-4. 様々なプログラミング言語

# プログラミングを学ぶときに気を付けること

- **プログラミング言語**は**多種多様**
- それぞれの言語に，特性と利用シーンがある
- **プログラミング**の**基本理念**と**基礎知識**を**理解**することが重要.
- **一つのプログラミング言語**で**基本を身につける**ことで、**他の言語への適応**もスムーズに進むでしょう



# さまざまなプログラミング言語

- Python
- C
- Java
- JavaScript
- R
- Octave
- Scheme

など

ここで行う作業

1．20 より大きければ「big」，
　　さもなければ「small」と表示

2．0 + 1 + 2 + 3 + 4 + 5 を求める

# なぜプログラミング言語はたくさんあるのでしょうか？

| | Python | Java | C / C++ | R | SQL | MATLAB / Octave |
|---|---|---|---|---|---|---|
| **それぞれ特徴がある** | どのコンピュータでも同じプログラムが実行可能 | シンプルで，実行も簡単．初心者にとって学びやすい． | コンピュータの性能を最大限引き出すために適する． | データ処理に特化したコマンド言語 | データベースに特化したコマンド言語 | 数値計算，信号処理などに特化したコマンド言語 |

# Python プログラム見本

```
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)
```

- シンプルで，実行も簡単．初心者にとって学びやすい．
- 多種多様なパッケージを利用することで、初心者でも容易に強力な機能を追加できる．

# Java プログラム見本

```
public class Main {
    public static void main(String[] args) throws Exception {
        int x = 100;
        if (x > 20) {
            System.out.printf("big¥n");
        } else {
            System.out.printf("small¥n");
        }
        int s = 0;
        for(int i = 1; i <= 5; i++) {
            s = s + i;
        }
        System.out.printf("%d¥n", s);
       }
}
```

- Javaはどのコンピュータでも同じプログラムが実行可能
- Windows、Linux、そしてAndroidアプリなど、異なる環境でも同じソースコードで動作
- このように，Java は互換性が高く，広範なアプリケーション開発に適する

# C プログラム見本

```
#include <stdio.h>
int main(void){
    int x, s, i;
    x = 100;
    if (x > 20) {
        printf("big¥n");
    } else {
        printf("small¥n");
    }
    s = 0;
    for(i = 1; i <= 5; i++) {
        s = s + i;
    }
    printf("%d¥n", s);
    return;
}
```

- CとC++はコンピュータの性能を最大限引き出すために適する
- 細かな制御や高速な実行に向いている
- チューニングにより最適化できる．高度なプログラミングやパフォーマンス重視のアプリケーション開発に適する

# R プログラム見本

```
x <- 100
if (x > 20) {
    print("big")
} else {
    print("small")
}
s <- 0
for (i in c(1,2,3,4,5)) {
    s <- s + i
}
print(s)
```

- Rはデータ処理に特化したコマンド言語
- データ専門家にも適する
- Rは豊富な統計やデータ解析の機能を提供
- データの可視化やモデリングなどの作業を効率的に行うことが可能

# Octave プログラム見本

```
x = 100
if (x > 20)
  printf("big¥n")
else
  printf("small¥n")
endif
s = 0
for i = [1 2 3 4 5]
  s = s + i
endfor
printf("%d", s)
```

- 数値計算や信号処理などに特化したコマンド言語
- 行列計算や信号処理などの科学技術計算に向いている
- 高度な数値演算やデータ解析が容易に行える

# JavaScript プログラム見本

```
process.stdin.resume();
process.stdin.setEncoding('utf8');
var util = require('util');
var x = 100;
if (x > 20) {
    process.stdout.write('big¥n');
} else {
    process.stdout.write('small¥n')
}
var s = 0;
for(var i = 1; i <= 5; i++) {
    s = s + i;
}
process.stdout.write(util.format('%d¥n', s));
```

- インタラクティブなウェブページの作成に適する
- そのとき，ユーザーとのリアルタイムな対話，動的なコンテンツの表示が可能
- 幅広い種類の OS でサポートされている

# Scheme プログラム見本

```
(define (decide x)
    (cond
        ((> x 20) "big")
        (else "small")))
(define (sum n)
    (cond
        ((= n 0) 0)
        (else (+ (sum (- n 1)) n))))
(begin
    (print (decide 100))
    (print (sum 5)))
```

- シンプルで明確な構文を持つ
- 関数型プログラミング言語
- 強力な再帰処理や高階関数の活用が簡単にできる

# さまざな種類のプログラミング言語

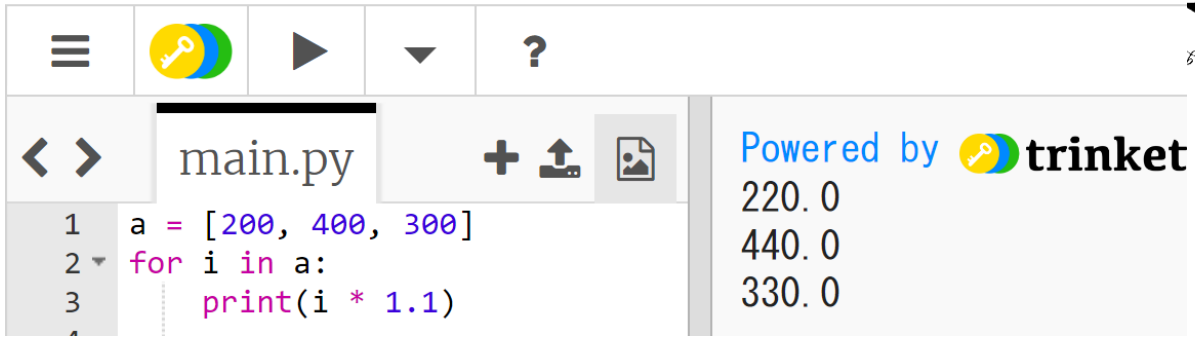
- プログラミングの**基本理念**と**基礎知識**を**理解**していくことが重要
- 一つの言語で基礎を身につけることで，**他の言語への適応もスムーズに**進む

**なぜプログラミング言語はたくさんあるのか？**

- **異なるニーズや目的に対応**
- 広範な用途に適するもの（Python, Java, JavaScript など）もあれば，特定の領域でより強力な機能を提供するものも
- **自分の目標や学びたいことに応じて言語を選ぶことが重要.** 複数の言語を使い分けることもある.

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

Python プログラムの
ソースコード




オンラインでの Python プログラム
実行（trinketを使用）

```python
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)
```

**Python**

```java
public class Main {
    public static void main(String[] args) throws Exception
{
        int x = 100;
        if (x > 20) {
            System.out.printf("big¥n");
        } else {
            System.out.printf("small¥n");
        }
        int s = 0;
        for(int i = 1; i <= 5; i++) {
            s = s + i;
        }
        System.out.printf("%d¥n", s);
      }
}
```

**Java**

```c
#include <stdio.h>
int main(void){
    int x, s, i;
    x = 100;
    if (x > 20) {
        printf("big¥n");
    } else {
        printf("small¥n");
    }
    s = 0;
    for(i = 1; i <= 5; i++) {
        s = s + i;
    }
    printf("%d¥n", s);
    return;
}
```

**C**

さまざまな
プログラミング言語

## ① IT 技術の可能性

プログラミングを通じて，新しい創造，新しい発見，作業の自動化などが実現できます。

## ② コンピュータを活用した問題解決

プログラミングを通じた問題解決の手順の理解は、**問題解決能力**や**論理的思考力**の向上に役立ちます。

## ③ 批判的思考

「**コンピュータを使えば計算は完璧に正確**」という一般的な**思い込みを覆す**ことで、常に批判的に見ることや、**根拠を確認すること**が重要である。

## ④ プログラミング言語の多様性

一つの言語で基礎を身につけることで，他の言語への適応もスムーズに進みます．将来，さまざまなプログラミング言語を学ぶことで，それぞれの特性や利点を理解し，ニーズに合わせた言語を選択する能力が取得できます．

# まとめ

- **プログラミングは創造的な活動**であり，多様な問題を解決し，**人間の能力を増幅**する.
- **プログラミング**は，計算，データ処理，データ送受信，人工知能，グラフィックス，シミュレーションなどの**多様な活動**を可能にする
- **コンピュータによる計算**には，精度の限界が存在する.「1 ÷ 3」などの単純な計算でも**微小な誤差が含まれている場合がある**.
- **プログラミング言語は多様**であり，それぞれの言語に特性がある．**異なるニーズ**や**目的にあわせて**，**プログラミング言語が選択**される.
- **基本的な概念と知識を一つの言語で習得**することにより，**他の言語**への**適応**も容易になる.