


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



- 
- 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
2. Python
3. Python プログラム実行
4. プログラムによる問題解決
5. 計算誤差
6. さまざまなプログラミング言語

4-1. プログラミング

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

```
x1="100%" y1="0%" x2="0%" y2="100%"  
color="#06101F" offset="0%">  
color="#1D304B" offset="100%">
```

```
height="450" rx="8" fill="url(#...)
```

```
<rect x="0" y="0" width="100%" height="96" viewBox="0 0 96 96" fill="url(#...)">
```

```
<linearGradient x1="87.565%" y1="15.875%" x2="0%" y2="100%">
```

```
<stop stop-color="#FFF" stop-opacity="1" offset="0%">
```

```
<stop stop-color="#FFF" offset="100%">
```

```
</linearGradient>
```

```
<rect x="-500%" y="-500%" width="1000%" height="1000%" fill="url(#...)">
```

```
<filter id="SourceAlpha" in="SourceAlpha" result="alpha">
```

```
<feGaussianBlur stdDeviation="24" in="alpha" result="blur">
```

```
</feGaussianBlur>
```

```
</filter>
```

```
<rect x="0" y="0" width="100%" height="96" viewBox="0 0 96 96" fill="url(#...)">
```

```
</rect>
```

```
</g>
```

```
</svg>
```

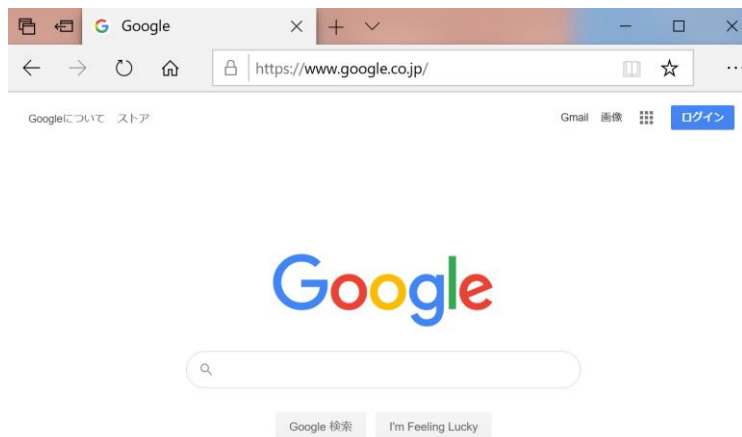
プログラミング



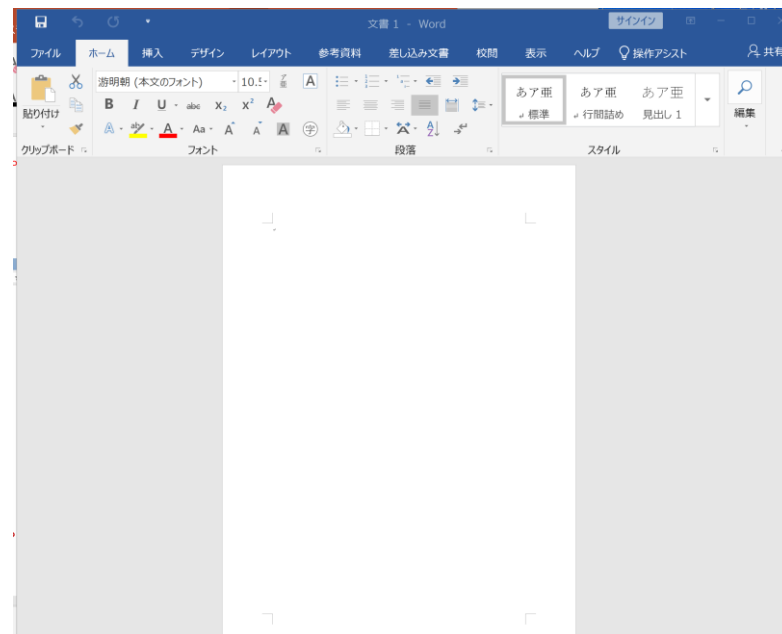
- **プログラム**を設計し作成する
プロセス（プログラミング）
は、**創造的な活動**
- アイデアを形にできることが、
プログラミングの魅力



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



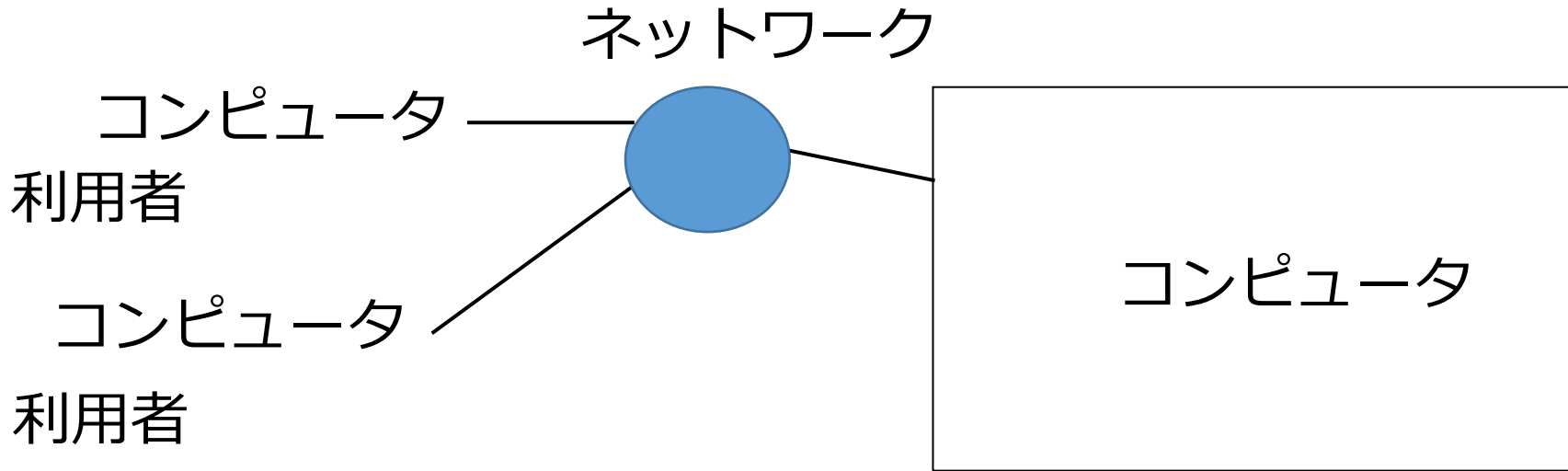
```
In [7]: from keras.models import Sequential
...: model = Sequential()
...: from keras.layers import Dense, Activation
...:
...: model.add(Dense(units=64, input_dim=len(x_train[0])))
...: model.add(Activation('relu'))
...: model.add(Dense(units=max(set(y_train)) - min(set(y_train)) + 1))
...: model.add(Activation('softmax'))
...: 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ロール

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.

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のもの
- 人間も読み書き，編集できる
- ソースコードにより，プログラムの動作を理解し，必要に応じて改変できる



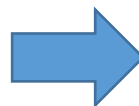
```
function isVisible() {  
    //is the element hidden?  
    if (!t.is(':visible')) {  
        //it became hidden  
        t.appeared = false;  
        return;  
    }  
  
    //is the element inside the visible window?  
    var a = w.scrollLeft();  
    var b = w.scrollTop();  
    var o = t.offset();  
    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);  
};
```

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)
```



```
216.0
432.0
324.0
```

Python プログラムの
ソースコード

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加担に調整できる



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

4-2. Python

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 Sequential
: model = Sequential()
.: from keras.layers import Dense, Activation
.:
... model.add(Dense(units=64, input_dim=x_train.shape[1]))
... model.add(Activation('relu'))
... model.add(Dense(units=max(set(y_train).difference({0})),
... 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))
... print(score)
... model.predict(x_test)
... model.summary()

Epoch 1/200
3 [=====] - 0s
3200

Epoch 2/200
3 [=====] - 0s
3200

Epoch 3/200
3 [=====] - 0s
3200
```


Python 言語の特徴



- **簡単**

Python は、**単純で読みやすい文法**. 「初心者に学びやすい」言語

- **便利**

多数の拡張機能を持つ

- **高度で複雑なプログラムも作成可能**

オブジェクト, クラス, メソッド, 属性, クラス階層, 継承などの機能

4-3. Python プログラム実行

コマンドによる Python プログラム実行



① Python プログラムの保存

```
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

- 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

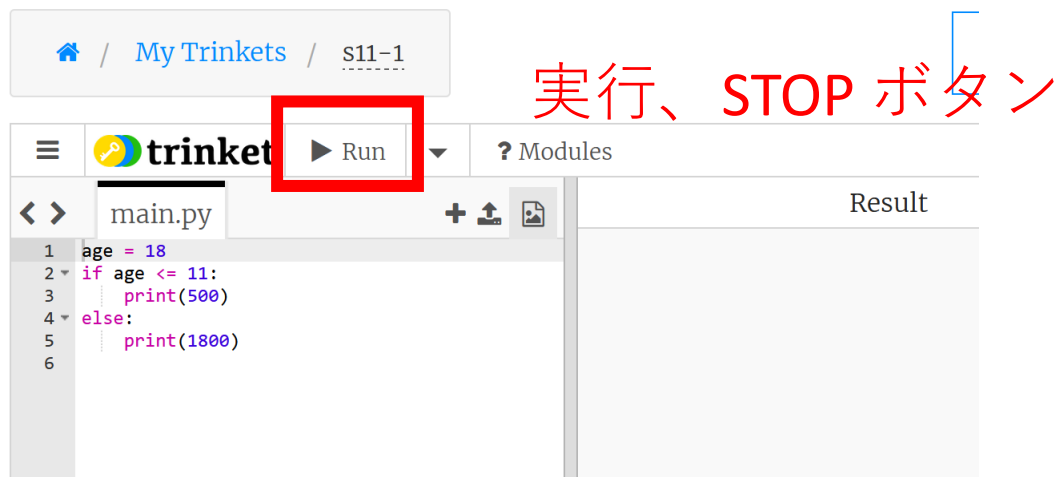
trinket でのプログラム実行



- 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



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

演習

Python プログラムの実行

資料 : 23, 24

【トピックス】

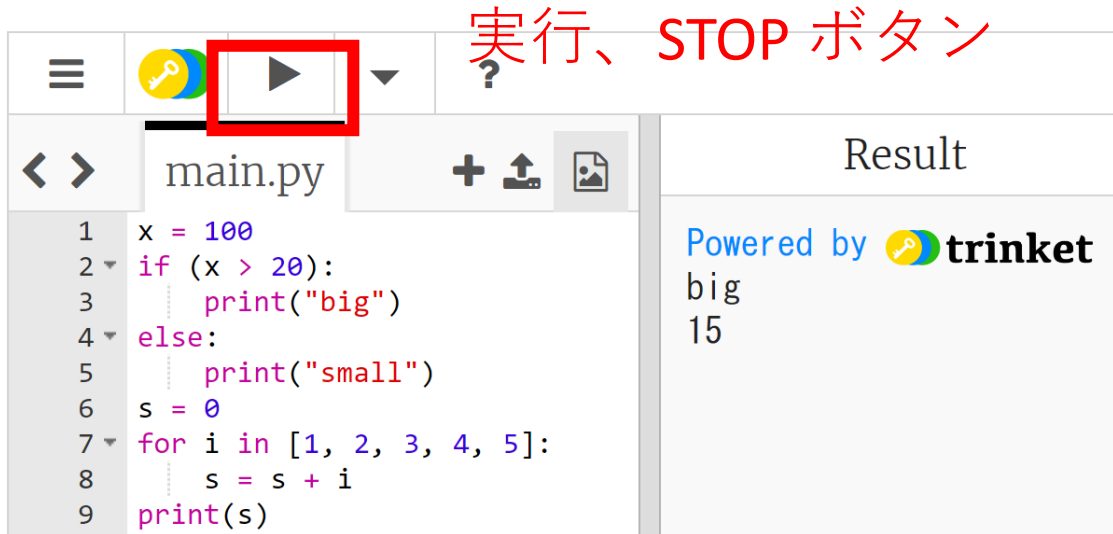
- trinket
- Python プログラムのソースコード
- Python プログラムの実行

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、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

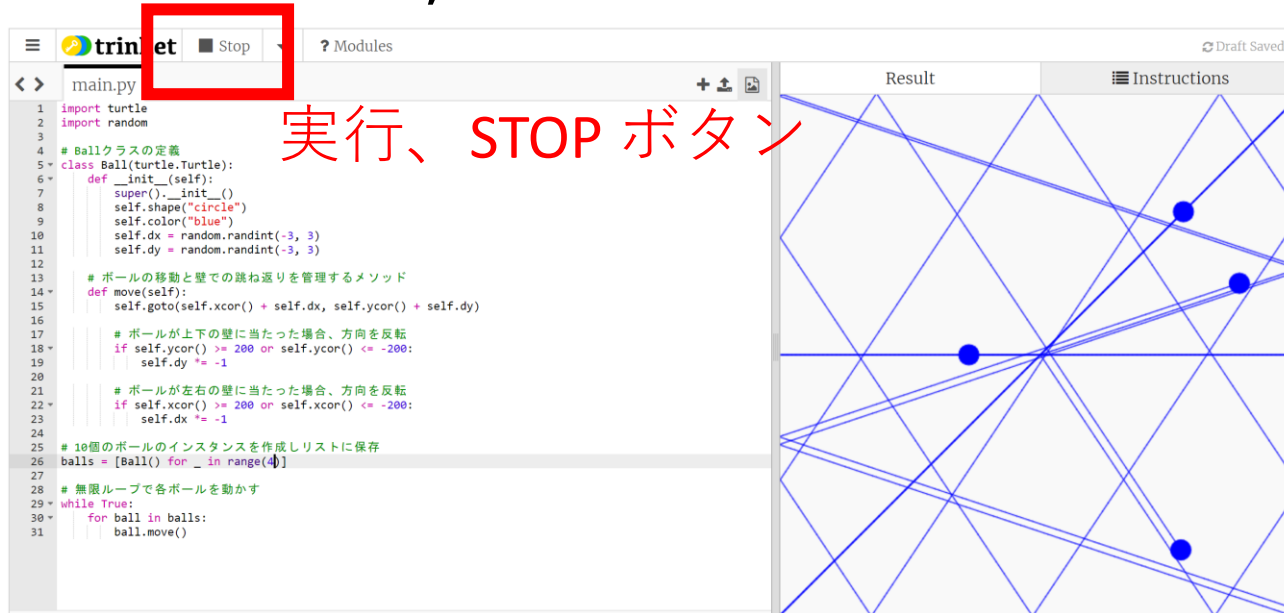
big
15

- 実行が開始しないときは、「**実行ボタン**」で**実行**
- 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



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

4-4. プログラムによる問題解決

プログラムの役割と魅力



- **プログラム**は, **コンピュータ**を使って多様な問題
を解決する手段になる
- **プログラミング**を学ぶことで, コンピュータをより効果的に活用でき, 人間の能力を増幅できる

アプリケーション

Word, Excel, Web ブラウザなど

自作のプログラムなど

自作の Python プログラム, Java プログラムなど

コンピュータ

プログラムが可能にすること



- **計算問題**：基本的な計算から高度な数学的な問題まで
- **データ処理**：データの整理，蓄積，分析，資格化
- **データ送受信**
- **人工知能**
- **グラフィックスやシミュレーション**

これらは，プログラムが可能にする一部.

アイデアを具現化し，作業を自動化し，新発見や創造の促進になる．これがプログラムの魅力．

演習

簡単なプログラムでも さまざまなことが可能

資料：29 ～ 32

【トピックス】

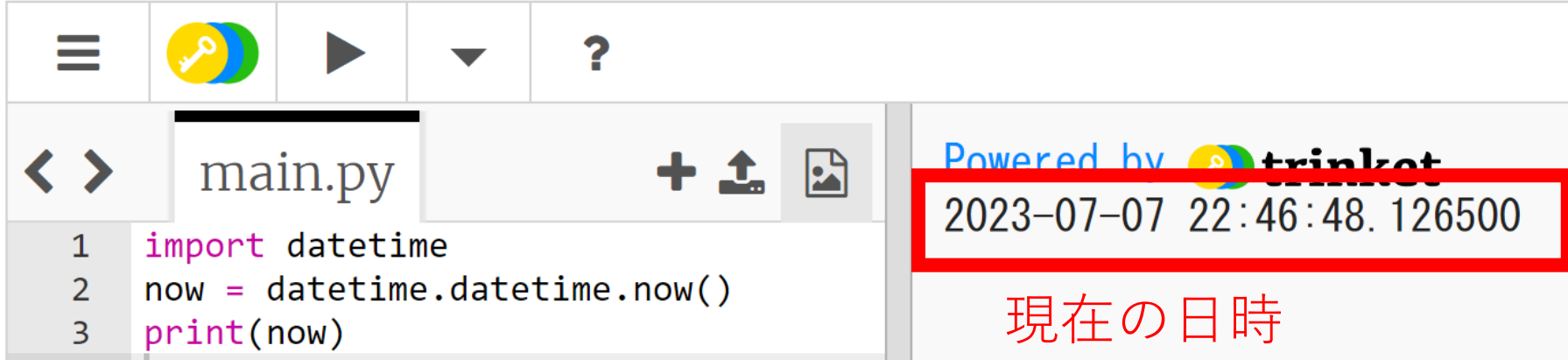
- trinket
- 現在の日時
- 平方根
- 円の面積
- 三角形の面積

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



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

Powered by trinket

2023-07-07 22:46:48.126500

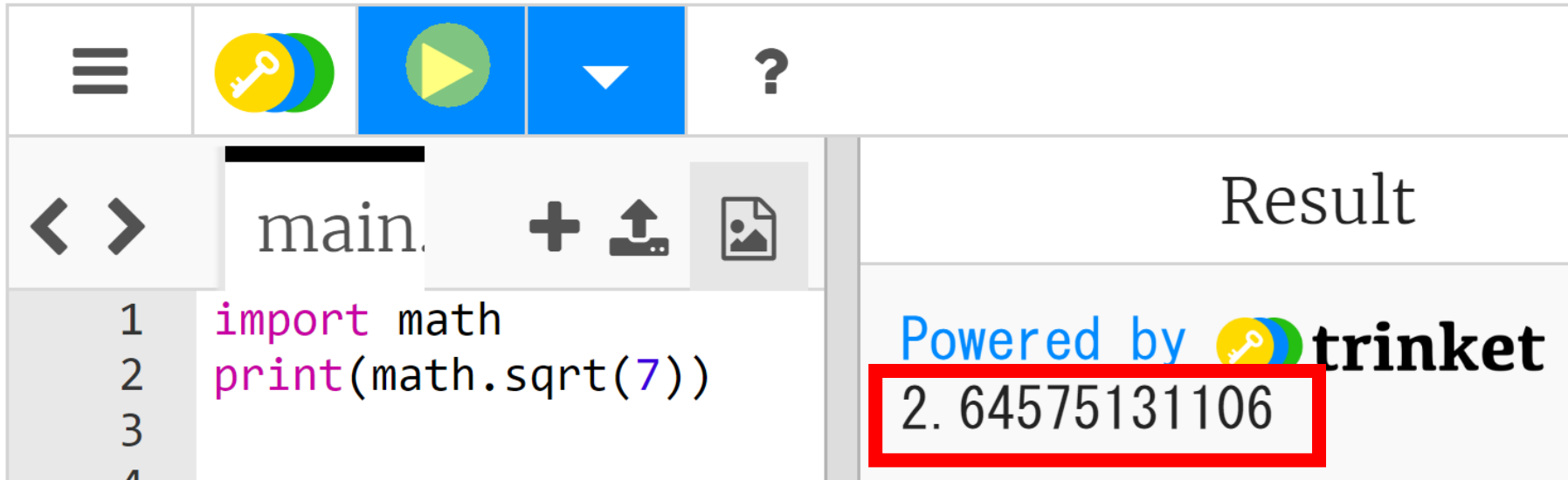
現在の日時

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



The screenshot displays the Trinket.io Python editor interface. The top navigation bar includes a menu icon, a key icon, a play button, a dropdown arrow, and a help icon. Below this, the file explorer shows a file named 'main.py'. The code editor contains the following Python code:

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

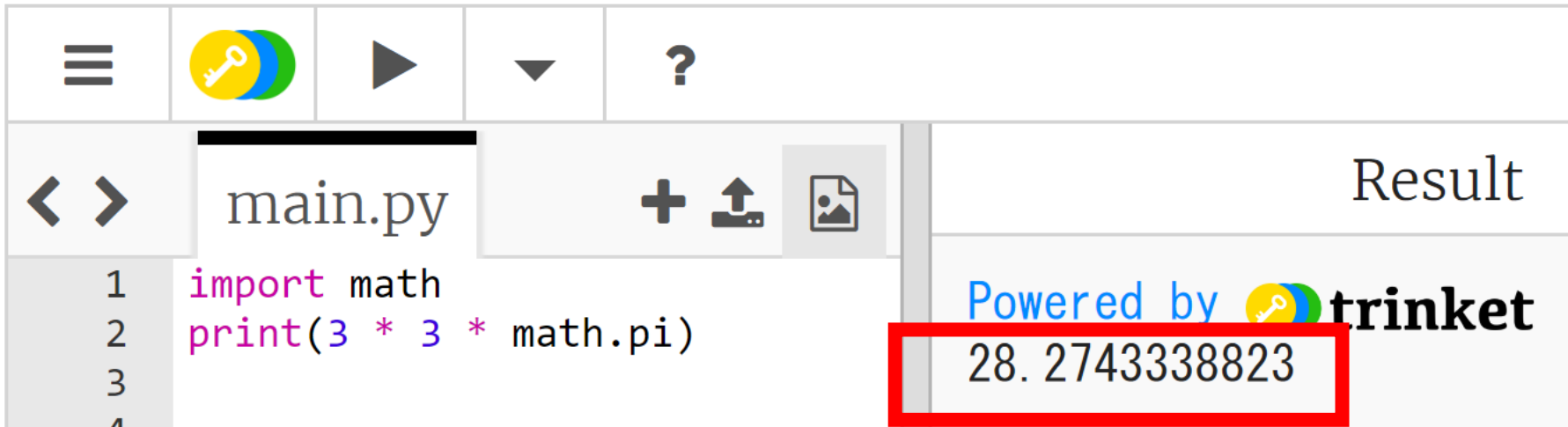
The right side of the interface shows the 'Result' section, which displays the output of the code execution: '2.64575131106'. The output is highlighted with a red rectangular box. The text 'Powered by trinket' is visible above the result.

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



The screenshot shows the Trinket.io Python editor interface. The top bar contains icons for a menu, a key, a play button, a dropdown arrow, and a question mark. Below the top bar, 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, there are icons for adding a new file, uploading a file, and inserting an image. The "Result" section on the right displays the output of the code:

Powered by  **trinket**

28. 2743338823

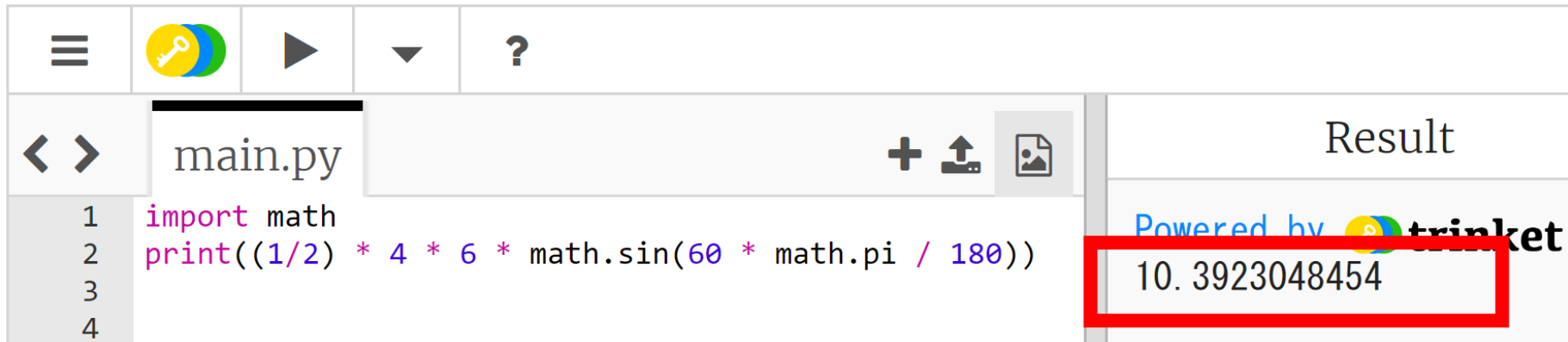
The result value "28. 2743338823" is highlighted with a red rectangular box.

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



The screenshot shows the Trinket.io Python editor interface. The top bar contains a menu icon, a key icon, a play button, a dropdown arrow, and a help icon. Below the bar, the file name 'main.py' is displayed. The code editor shows the following Python code:

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

On the right side, the 'Result' section displays the output of the code: '10.3923048454'. The result is highlighted with a red rectangular box. Above the result, it says 'Powered by trinket'.

4-5. 計算誤差

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「 $1 \div 3$ 」の計算

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

- コンピュータは「**0.333333333333333333**」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、「 $1 \div 3$ 」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

資料：37, 38

【トピックス】

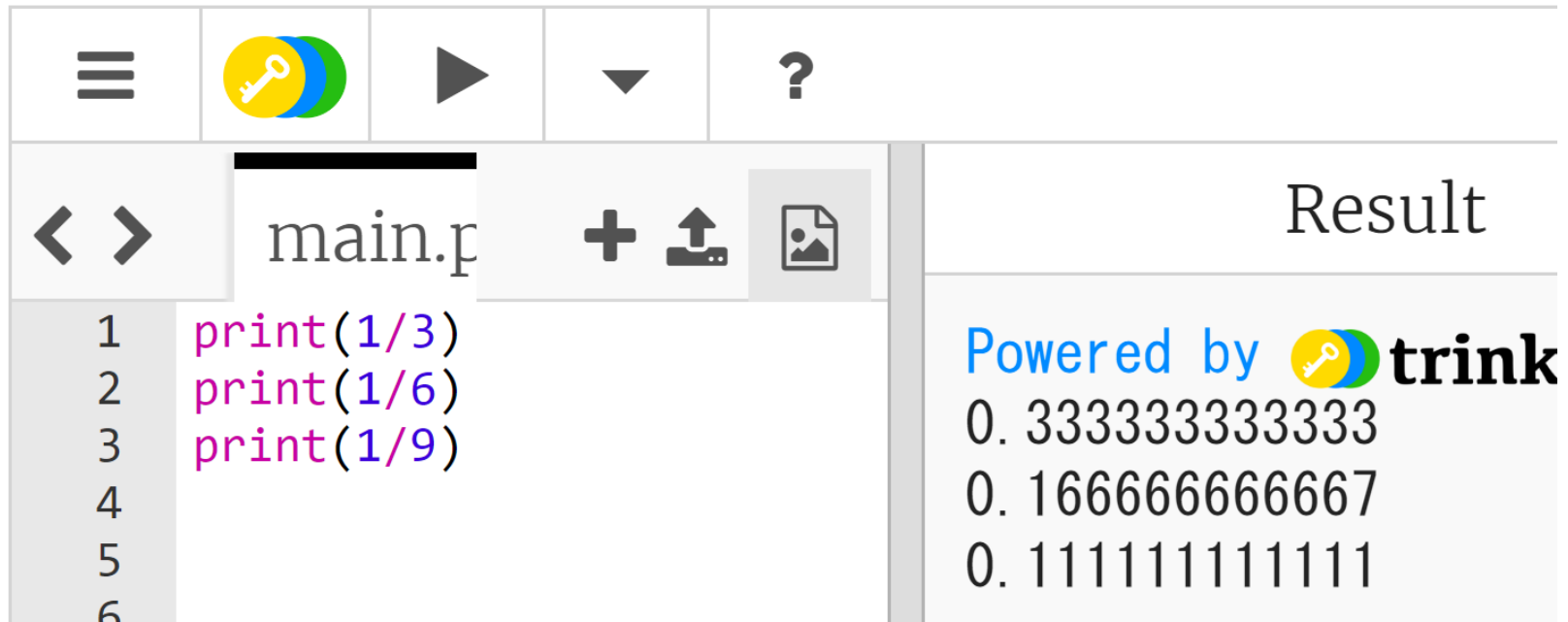
- ・ 計算誤差

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



The screenshot shows the Trinket.io Python editor interface. The top bar contains icons for a menu, a key, a play button, a dropdown arrow, and a help icon. Below the top bar, the file name 'main.p' is displayed. The code editor shows the following code:

```
1 print(1/3)
2 print(1/6)
3 print(1/9)
4
5
6
```

The right panel, titled 'Result', displays the output of the code:

```
Powered by  trink
0. 3333333333333
0. 1666666666667
0. 1111111111111
```

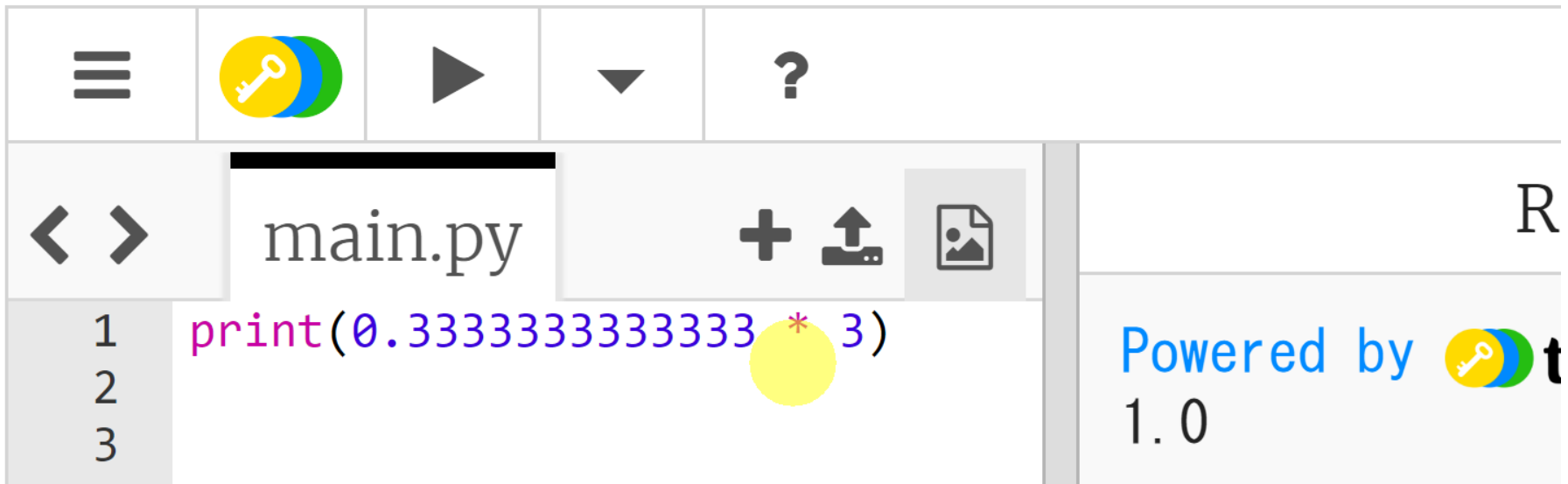
計算誤差がある

0.333333333333333 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

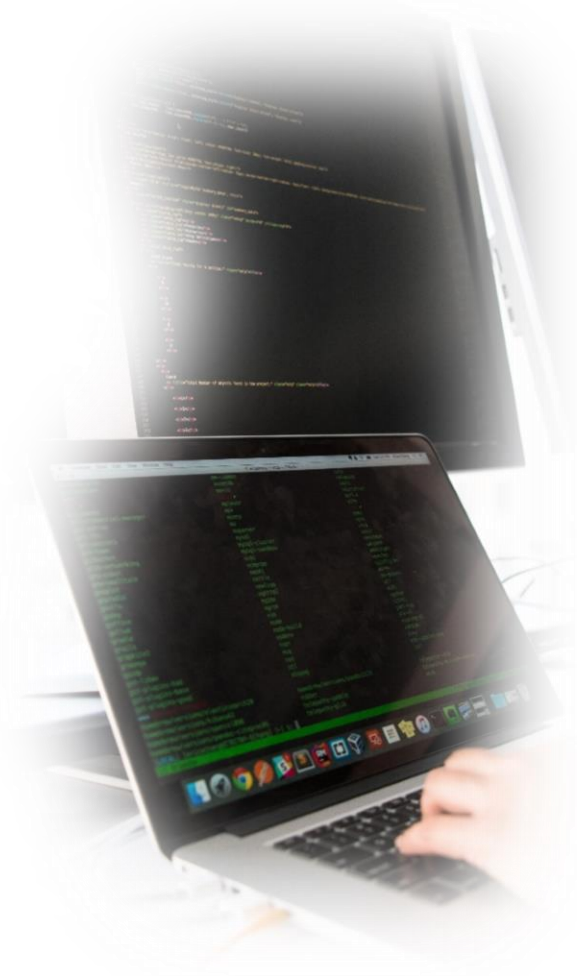
- 「コンピュータを使えば計算は完璧に正確」 - この思い込みは避けましょう。
- 例えば, 「 $1 \div 3$ 」の計算結果は, 完全な精度では表示できません. 少ないながらも誤差が含まれます.
- しかしこの微小な誤差は、多くの場合、**私たちの作業には十分な精度**
- 少しの誤差を許容すれば、**多くの計算が効率的に処理できる**という考え方も.

4-6. さまざまなプログラミング 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？



それぞれ
特徴があ
る

Python

どのコン
ピュータ
でも同じ
プログラ
ムが実行
可能

Java

シンプル
で、実行
も簡単.
初心者
にとって学
びやすい.

C / C++

コン
ピュータ
の性能を
最大限引
き出すた
めに適す
る.

R

データ処
理に特化
したコマ
ンド言語

SQL

データ
ベースに
特化した
コマンド
言語

**MATLAB /
Octave**

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

さまざま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)
```



```
main.py
1 a = [200, 400, 300]
2 for i in a:
3     print(i * 1.1)

Powered by trinket
220.0
440.0
330.0
```

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）

```
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

```
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

```
#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。

- **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．
「 $1 \div 3$ 」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．