生徒がプログラミングする際の留意点

大阪電気通信大学
教授・副学長　　兼宗進

「情報Iプログラミング」の解説動画

- 「情報Iプログラミング」の解説動画
  - [1]センサーライトを作ろう!（モデル化と外部機器を活用したシミュレーション ）
  - [2]100連ガチャをプログラムして作ろう!（アルゴリズムの基本とプログラミング ）
  - [3]公平な方法で発表順番を決めよう!（データ構造　外部プログラムの連携 ）
  - [4]天気予報表示マシーンを作ろう!（情報通信ネットワークを活用したプログラミング ）
- 高校の先生方（三井先生/鎌田先生）が説明
- わかりやすく、そのまま授業で活用できる内容
- 今日は動画の内容にも触れながら授業について確認
  - 情報Iで扱うプログラミングの基礎
  - プログラミング指導のポイント
  - プログラミングを利用した学習例

情報Iで扱うプログラミングの基礎
（Python言語を例に）

プログラミングの授業の流れ

- printで画面に文字を表示する。「''」か「""」で囲む（どちらでもよい）

```
print("ABC") → ABC
```

- 2回表示する。プログラムは上から順に実行される（順次実行）

```
print("ABC") → ABC
print("DEF")    DEF
```

- 「end=""」で改行せずに表示できる

```
print("ABC", end="") → ABCDEF
print("DEF")
```

数値を扱う

- printで表示
  `print(3)` → 3
- printで計算
  `print(3+4)` → 7
- 四則演算
  `print(3+4*2)` → 11
  `print(3+6/2)` → 6
  `print(5//3)` → 1
  `print(5%3)` → 2

数値変数を扱う

- 代入することで変数を作れる

```
a=3
print(a+2) → 5

a=3
b=4
print(a+b) → 7
```

条件分岐を使う

- ifで数値の大小比較(==, !=, >, >=, <, <=)

```
a=5
if a>3:
    print("3より大きい")
```
→ 3より大きい

- ifとelseの利用

```
a=2
if a>3:
    print("3より大きい")
else:
    print("3以下")
```
→ 3以下

論理演算(andとor)

- 両方成り立つ(and)

```
a=5
if a>3 and a<10:
    print("3より大きく10より小さい")
```

- どちらか成り立つ(or)

```
a=12
if a<3 or a>10:
    print("3より小さいか10より大きい")
```

キーボードから入力

- inputで文字を入力

```
str=input()
print(str, "です。") → ABC です。
```

- ifで入力文字を判定（hirakegomaを入力した例）

```
pw=input()
if pw=="hirakegoma":
    print("OK")        → OK
```

- inputで数値を入力（実行例は2を入れた場合）

```
str=input()
num=int(str)
print(num*2) → 4
```

処理の繰り返し(for)

- forで処理を繰り返せる

```
for i in range(5):
    print("*", end=" ") → * * * * *
```

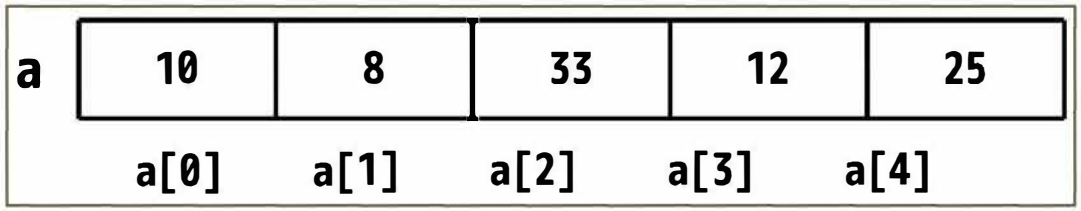
- 変数に何回目の実行かを入れられる(iに0,1,2,3,4が入る)

```
for i in range(5):
    print(i, end=" ") → 0 1 2 3 4
```

- 変数に入る値の範囲を指定できる(iに1,2,3,4,5が入る)

```
for i in range(1, 6):
    print(i, end=" ") → 1 2 3 4 5
```

リスト(配列)

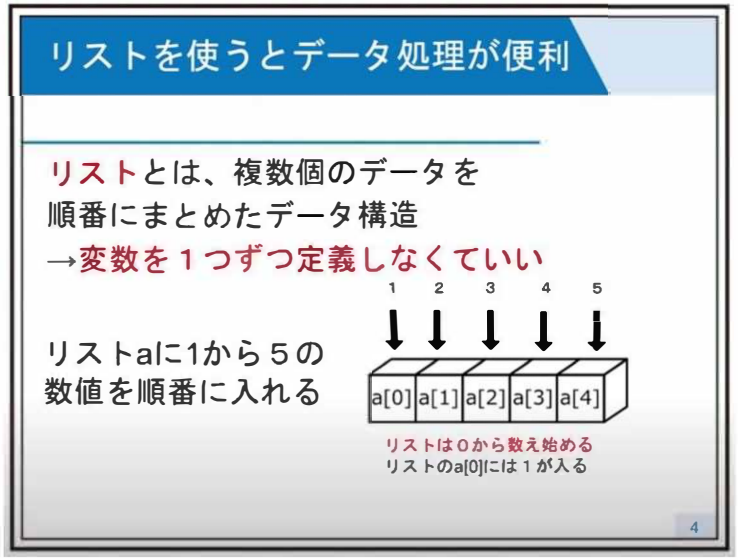


- 変数に複数の値を覚えておける

```
a=[10, 8, 33, 12, 25]
print(a)          → [10, 8, 33, 12, 25]
print(a[2])         → 33
```

- データを順に処理できる

```
a=[10, 8, 33, 12, 25]
for i in a:
    print(i*2, end=" ") → 20 16 66 24 50
```



関数

- 関数をdefで定義できる。この例ではp(5)で5個の*を表示している

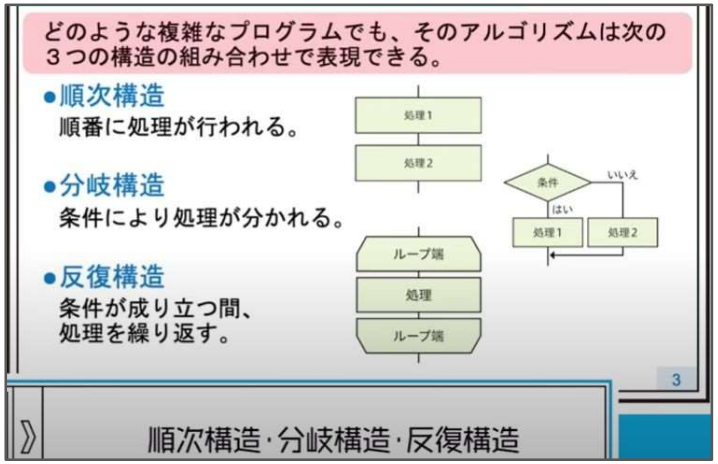
```
def p(n):
    for i in range(n):
        print("*", end="")
p(5)                    → *****
```

- 値をreturnで返せる。この例ではnijou(5)で5*5の結果を返している

```
def nijou(n):
    return n*n
r=nijou(5)
print(r)                → 25
```

情報Iで扱うプログラミングの内容

- プログラムを読み書きする上で必要な基礎を学ぶ
  - **変数、代入**
  - **条件分岐(if)、反復(for/while)**
  - **複数データ(リスト/配列)**
  - 関数定義



- 太字はプログラムの理解に必須。丁寧に学習したい

- 関数は意味を理解しておく。実用的なプログラム作成、情報II、入試で必要

プログラミング指導のポイント
（つまずきやすい点を中心に）

ポイント1： 進度の差、エラーの対処

- キー入力の速度に差、1文字の入力ミスでエラー
- 40人をできるだけ個別でなく全体で指導する工夫
- 生徒に伝える
  - エラーが起きたら、先生のサンプルと見比べよう
  - 隣の生徒のプログラムと見比べよう
  - 隣同士で教え合おう、早く終わった人は困っている人を助けてあげよう
- 起きやすい入力ミス
  - 閉じ忘れ 「`int(input()`」「`print("こんにちは)`」
  - 記号　　「`:`」の入れ忘れ、「`;`」などと間違える
  - 不等号　「`==`」を「`=`」と間違える
  - インデント

ポイント2：コンピュータの動きを考える

- わずか3行のプログラムだが生徒には意外と難しい？

```
a=[10, 8, 33, 12, 25]
for i in a:
    print(i*2)
```

- 実行すると「20 16 66 24 50」が表示される
- これだけを見ても動きを実感できない

- コンピュータになったつもりで、プログラムを指で指しながら動きを追うことで、右図のように実行されていることを理解できる

```
# このように実行される
a=[10, 8, 33, 12, 25]
i=10
print(i*2)
i=8
print(i*2)
i=33
print(i*2)
i=12
print(i*2)
i=25
print(i*2)
```

ポイント3： 学んだことの組み合わせ

- プログラムの部品の種類は多くない
- まずは、これらの意味を体験を通して理解しよう
  - **変数、代入、比較、条件分岐(`if`)、繰り返し(`for`,`while`)、リスト、関数**

- 実際のプログラムは、これらを組み合わせて使う
- 特に入れ子（ネスト）は重要
  - `if+if`
  - `for+if`
  - `for+for`

ifとifを組み合わせる

- 最初はifとelse

```
n=6
if n%2==0:
    print("偶数")
else:
    print("奇数")
```

- 次にifの中にif

```
n=6
if n%2==0:
    if n%3==0:
        print("6の倍数")
```

forとifを組み合わせる

- forで繰り返す0から4の数について、ifで条件を調べる

```
for i in range(5):
    if i>=3:
        print(i, end=" ") → 3 4
```

- forで繰り返す要素について、ifで条件を調べる

```
a=[10, 8, 33, 12, 25]
for i in a:
    if i>15:
        print(i, end=" ") → 33 25
```

forとforを組み合わせる(1)

- 「1から9までの表示」を9回繰り返す

```
for i in range(1,10):
    for j in range(1,10):
        print(j, end=" ")
    print()
```

→

```
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
```

- 「nの段の表示」を9回繰り返す

```
for i in range(1,10):
    for j in range(1,10):
        print(i*j, end=" ")
    print()
```

```
1 2 3 4 5 6 7 8 9
2 4 6 8 10 12 14 16 18
3 6 9 12 15 18 21 24 27
4 8 12 16 20 24 28 32 36
5 10 15 20 25 30 35 40 45
6 12 18 24 30 36 42 48 54
7 14 21 28 35 42 49 56 63
8 16 24 32 40 48 56 64 72
9 18 27 36 45 54 63 72 81
```

forとforを組み合わせる(2)

- 表示を空白1文字で区切ると縦が揃わない

```
for i in range(1,10):
    for j in range(1,10):
        print(i*j, end=" ")
    print()
```

```
1 2 3 4 5 6 7 8 9
2 4 6 8 10 12 14 16 18
3 6 9 12 15 18 21 24 27
4 8 12 16 20 24 28 32 36
5 10 15 20 25 30 35 40 45
6 12 18 24 30 36 42 48 54
7 14 21 28 35 42 49 56 63
8 16 24 32 40 48 56 64 72
9 18 27 36 45 54 63 72 81
```

- (練習問題)1桁なら前に空白を入れる

```
for i in range(1,10):
    for j in range(1,10):
        if i*j<10:
            print(" ", end="")
        print(i*j, end=" ")
    print()
```

```
 1  2  3  4  5  6  7  8  9
 2  4  6  8 10 12 14 16 18
 3  6  9 12 15 18 21 24 27
 4  8 12 16 20 24 28 32 36
 5 10 15 20 25 30 35 40 45
 6 12 18 24 30 36 42 48 54
 7 14 21 28 35 42 49 56 63
 8 16 24 32 40 48 56 64 72
 9 18 27 36 45 54 63 72 81
```

ifとforとprintを組み合わせることで、複雑なプログラムを作れる

ポイント4： ifやforが実行する範囲

- CやJavaScriptは「{ }」で範囲を指定。インデントはなくても動く

```
for (i=0; i<3; i++) {
    b=3;
    printf("%d", b);
    x=4;
}
b=5;
```

- Pythonはインデントで範囲を指定する

```
for i in range(3):
    b=3
    print(b)
    x=4
b=5
```

```
a=3
 b=4     # エラー（ifやforがないのに字下げ）
for i in range(3):
    b=3
   x=4   # エラー（上の行と揃っていない）
```

インデントの範囲

- インデントでforやifの実行範囲を指定

```
for i in range(3):
    b=3
    print(b)
    x=4
b=5
```

- インデントの深さで、どの実行範囲かを区別する
「b=3」は2段のインデント（if）、「x=4」は1段のインデント（for）

```
for i in range(3):
    if x<3:
        b=3
        print(b)
    x=4
b=5
```

ポイント5： データの種類(データ型)

- よく使うのは数値と文字列(数値は整数と実数がある)
- 'A'が文字列なのは明確だが、'2'が数でなく文字なのはわかりにくい

- inputの入力は文字。表示はできるが計算はできない(2を入力した例)

```
str=input()
print(str)
print(str+3)
```
→ エラー

- intで整数に変換して使う(2を入力した例)

```
str=input()
n=int(str)
print(n+3)
```
→ 5

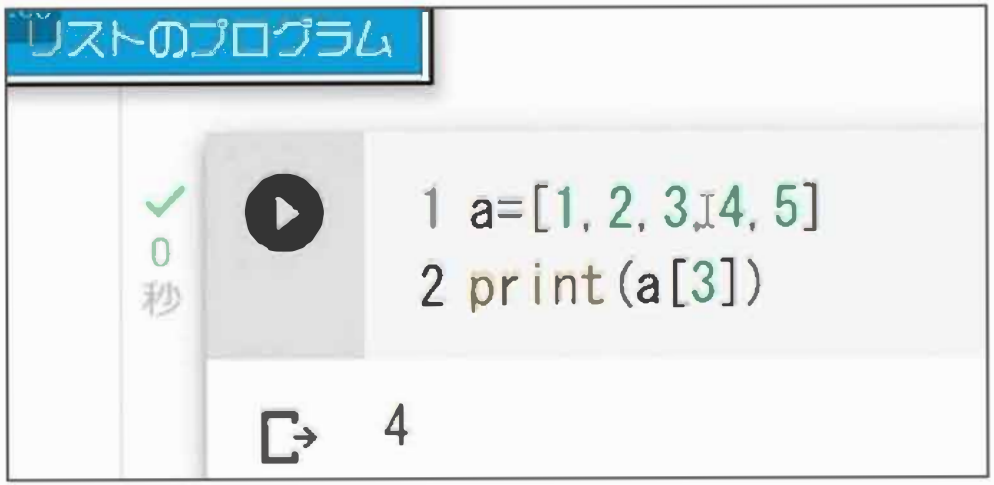
ポイント6： リスト(配列)

- 添字(インデックス)は0から始まる

```
a=[1, 2, 3, 4, 5]
print(a[3]) → 4
```

- 数値だけでなく、文字列も入れられる

```
te=['gu', 'choki', 'pa']
```



・リストと組み込みモジュールでじゃんけんを作る
→リストに「ぐー、ちょき、ぱー」
乱数の組み込みモジュールを使用

プログラム

```
import random
te=['gu','choki','pa']
a=random.randint(0,2)
print(te[a])
```

ポイント7： エラーの表示

- エラーになって実行できないことが続くと意欲に影響する
- 実行前に、編集画面にヒントが表示されていることがある
- エラーメッセージはプログラマ用のため、わかりやすくないことがある

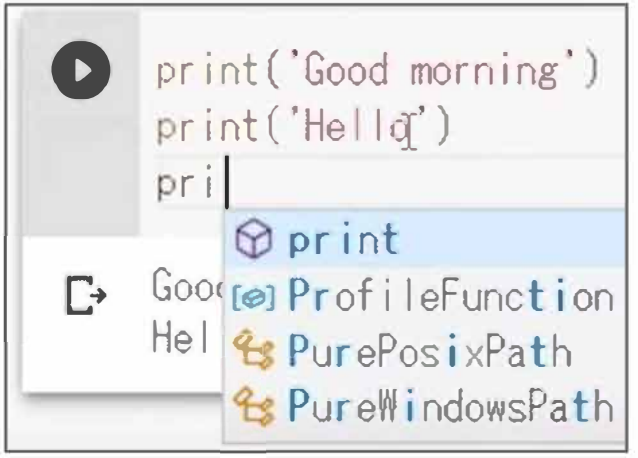
```
1 import random
2 for i i range(100):
3 a=random.randint(1,100)
4 print(a)
5 if(a==1):
6   print('atari')
7 else:
8   print('hazure')
```

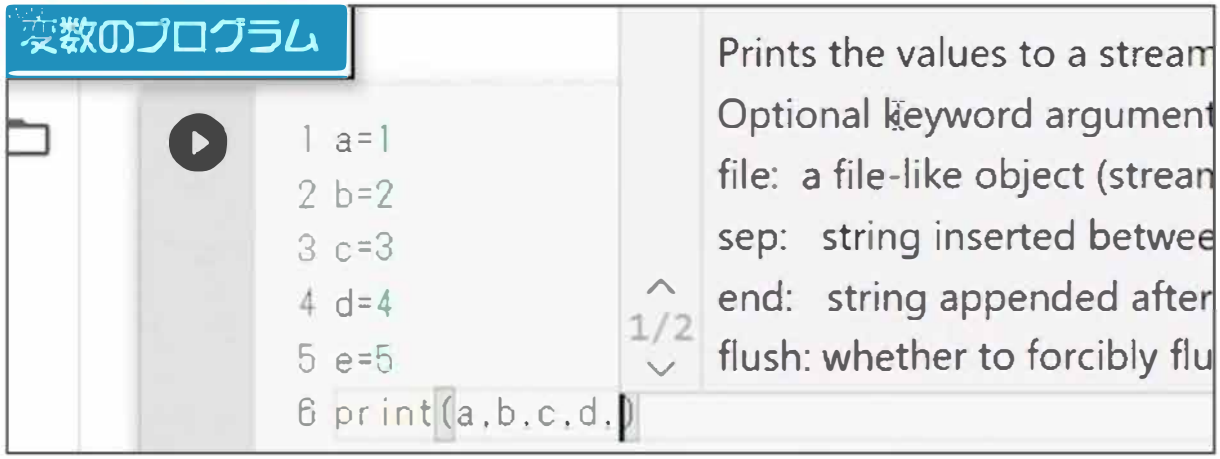
```
1 import random
2 for i i range(100):
3 ··a=random.randint(1,100)
4 ··print(a)
5 ··if(a==1):
6 ····print('atari')
7 ··else:
8 ····print('hazure')

File "<ipython-input-54-114d4bed7dbc>", line 2
  for i i range(100):
```

ポイント8： 入力補助機能

- 入力補助機能は活用すると便利
- プログラマ用のツールのため、習っていない機能を含めて多くの情報が表示されてしまうこともある





ポイント9： ブロックプログラミング

- 見てわかりやすい。文法エラーが起きない
- ブロックで考え方を学んだ後は、文字で記述するプログラムも紹介したい
- 以下は、LEDのプログラムをJavaScriptとPythonに書き直した例

JavaScript

```
1  basic.forever(function () {
2      if (input.lightLevel() < 50) {
3          basic.showLeds(`
4              # # # # #
5              # # # # #
6              # # # # #
7              # # # # #
8              # # # # #
9              `)
```

```
# Pythonプログラム
while(true):
  if bright()<50:
    LED(0,0,0,0,0,
        0,0,0,0,0,
        0,0,0,0,0,
        0,0,0,0,0,
        0,0,0,0,0)
```

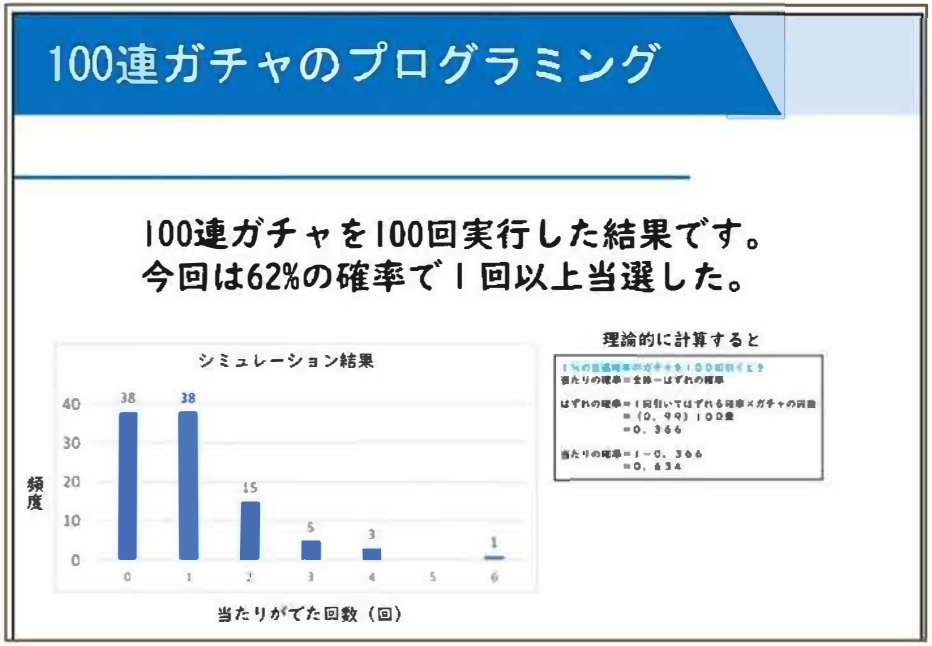
指導のポイント

- プログラムの理解は進度の差が大きい
  - クラス全体を指導しながら、つまずきやすいポイントを伝えたり、生徒同士の教え合いを活用したい
- 生徒が困るポイントを押さえておく
  - 典型的なパターンを知っておくことで対応が可能
  - プログラムという指令書でコンピュータが動く
  - 学んだ部品を組み合わせて意味のあるプログラムを作る
  - if/forの実行する範囲を理解する
  - データ型、リストの添字など

プログラミングを利用した学習例

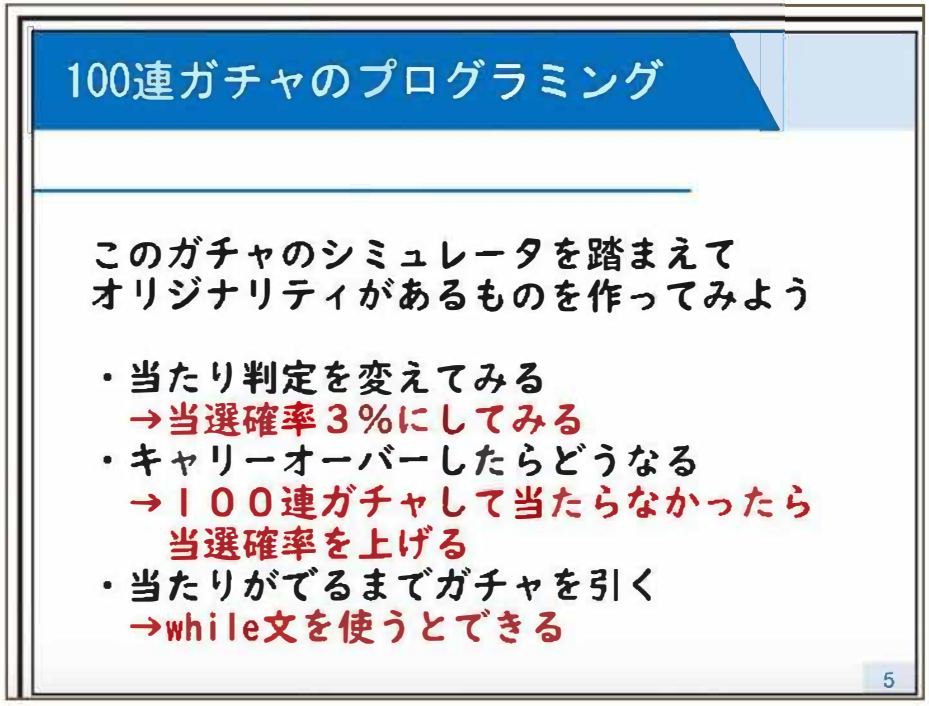
プログラミングを利用した学習の可能性

- プログラムの学習だけだと何に役に立つのかを理解しにくい
- プログラミングができるようになると、**他の単元と合わせて学習できる**
- 「100連ガチャのプログラミング」動画では、学んだプログラミングのスキルを**シミュレーションの学習**に利用していた



発展的な学習も可能

- 学んだプログラミングを利用して、**問題解決に結びつけたい**
- 解決したい問題を考えて、プログラムによる解決方法を検討する
- 発想しやすいように、いくつかの例を示すことは有効



学習環境の例(Bit Arrowなど)

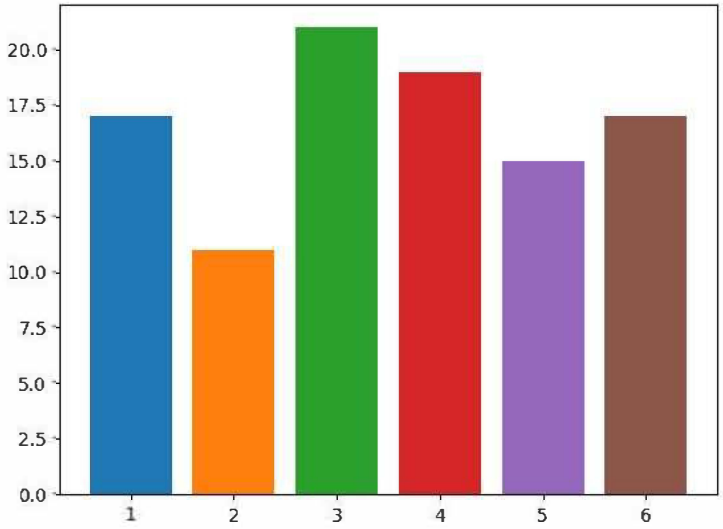
- いくつか授業用ツールを開発して公開しています。ご利用ください
- Bit Arrow:プログラミング学習ツール
  - ブラウザでプログラムを記述して実行
  - 3大学で共同開発(大阪電気通信大学、東京農工大学、明星大学)
  - 言語は、Python、DNCL、C、ドリトル、Processing、教育用JavaScript
  - 生徒へのプログラム配布、生徒プログラムのダウンロードなど
  - 無料で教員登録可能(「授業利用に向けた準備」から「教員登録フォーム」)
    - https://bitarrow.eplang.jp/
- サクセス:データベース学習ツール
  - 大阪電気通信大学、大阪大学で共同開発
    - https://saccess.eplang.jp
- Connect DB:データ活用ツール
  - 大阪電気通信大学高等学校と共同開発
    - https://cdb.eplang.jp

グラフによる可視化

- 棒グラフで表示する

```
import random
import matplotlib.pyplot as plt
deme=[]
for i in range(100):
    deme.append(random.randint(1,6))
for i in range(1,7):
    plt.bar(i, deme.count(i))
plt.show()
```

→



各種の画像処理

- Bit Arrowの素材管理で画像ファイルを登録しておく

- 画像を表示

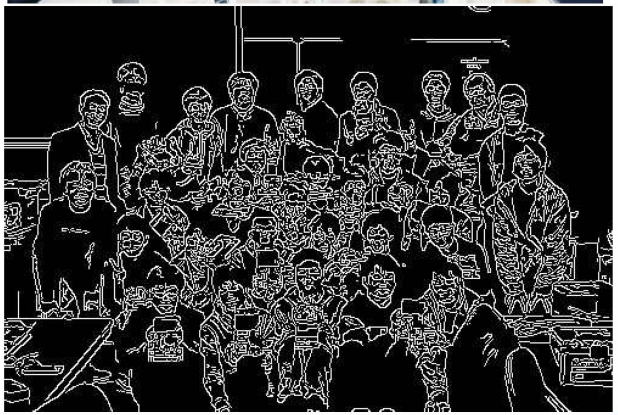
```
import cv2
img = cv2.imread("class/7-IMG_6564.jpg")
cv2.imshow("", img)
```

- 色を反転

```
rev = cv2.bitwise_not(img)
```

- 輪郭抽出

```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
canny = cv2.Canny(gray, 200, 100)
```



オープンデータの利用例

- 自治体（寝屋川市）の例
  - 利用条件は「CC BY」（原作者名を表記）

寝屋川市オープンデータについて

寝屋川市では、情報の利活用を推進し、地域の課題解決を図り、社会経済の発展に寄与するよう、市の保有する情報のオープンデータ化に取り組んでいます。

寝屋川市オープンデータサイトのデータの利用に関しては、CC（クリエイティブ・コモンズ）ライセンスによるCC－BYにより提供します。

データ名	カテゴリ	更新日	データ形式
AED設置個所一覧	安全・安心	2021年12月24日	csv
指定緊急避難場所一覧	安全・安心	2021年07月01日	csv
消防水利施設一覧	安全・安心	2021年07月01日	csv

	A	B	C	D	E	F	G	H
1	NO	名称	名称_カナ	住所	方書	緯度	経度	標高
2	1	東小学校	ヒガシショ	大阪府寝屋川市太秦		34.76139	135.6338	
3	2	第一中学校	ダイイチチ	大阪府寝屋川市高宮		34.76233	135.6317	
4	3	市民会館	シミンカイ	大阪府寝屋川市秦町		34.76288	135.6314	
5	4	東コミュニ	ヒガシコミ	大阪府寝屋川市高宮		34.76135	135.6313	
6	5	中央小学校	チュウオウ	大阪府寝屋川市初町		34.76379	135.6257	
7	6	寝屋川高等	ネヤガワコ	大阪府寝屋川市本町		34.7651	135.6259	
8	7	府立大学コ	フリツダイ	大阪府寝屋川市幸町		34.77043	135.629	
9	8	池田小学校	イケダショ	大阪府寝屋川市池田		34.7732	135.6171	
10	9	桜小学校	サクラショ	大阪府寝屋川市池田		34.77298	135.6115	
11	10	第二中学校	ダイニチュ	大阪府寝屋川市池田		34.76956	135.611	

  - CSV形式のテキストデータをBit Arrowに登録してプログラムから利用可能

避難場所のデータを地図に表示する

```
import pandas as pd
import folium

# 避難場所の位置情報を取得
shelters = pd.read_csv("class/272159_evacuation_space.csv", encoding="shift-jis")

# 地図作成
map = folium.Map(location=[shelters["緯度"][0], shelters["経度"][0]], zoom_start=12)

# 避難場所を1か所ずつマーク
for i in range(len(shelters)):
    latlng = shelters.loc[i, ["緯度", "経度"]].tolist()
    name = shelters.loc[i, "名称"]
    folium.Marker(latlng, popup=name).add_to(map)

# 地図表示
map.show()
```

全体のまとめ

- 今日の内容は「生徒がプログラミングする際の留意点」
- 情報Iのプログラミングは、次のことを学習
  - 基本的な考え方を学ぶ
    - 変数、代入、リスト、順次、分岐、反復
  - 学んだことを組み合わせてプログラムを作る
- 理解しにくい箇所や、実習で困りがちな箇所をご紹介
  - 全体に説明、教え合いを推奨
  - 困りそうなポイントを押さえておく
  - 文字で書いたプログラムをコンピュータが解釈して実行
- 慣れることは大切
  - 自分のアイデアでサンプルプログラムを拡張
  - シミュレーションやデータ活用などを含めてプログラミングを活用したい