アシアル情報教育研究所

大学入学共通テスト「情報」対策研修
2024年 第二部

アシアル情報教育研究所
所長 岡本雄樹

本研修の目的と時間割

- 第二部(10:30～11:45)
  - 情報Ⅰ・プログラミング領域の実習
    - おみくじアプリ（3倍速相当）
    - 試作問題(問3)解説と実習
  - WaPEN@Asail紹介

おみくじアプリ（3倍速相当）

実習環境・Monaca Education

(配布されたアカウントを使う場合)Monacaのログイン

1.教育版公式サイトURLにアクセス

https://edu.monaca.io/

2.ログインを選択

3. アカウントカードの情報をフォームに入力

(補足)後からSSO

- 配布されたアカウントに対して後からGoogleやMicrosoftアカウントを連携することもできます。
- 連携すれば毎回Monacaのパスワードを入力しなくて済みます。

第1章 アプリ開発入門

Monaca ダッシュボードとプロジェクト

ログインとダッシュボードの表示

- プラン管理
  - 右上のアイコンを経由してプラン管理を行えます。有料プランを利用するためのアクティベーションコードが適応できます。
- 開発中のアプリはプロジェクト単位で管理されます。
  - プロジェクトは画面左側に一覧表示されます。

Monacaクラウド IDE

- IDE(Integrated Development Environment)は統合開発環境の意味です。

エディター

- プログラムを記述するためのパネルです。

index.html

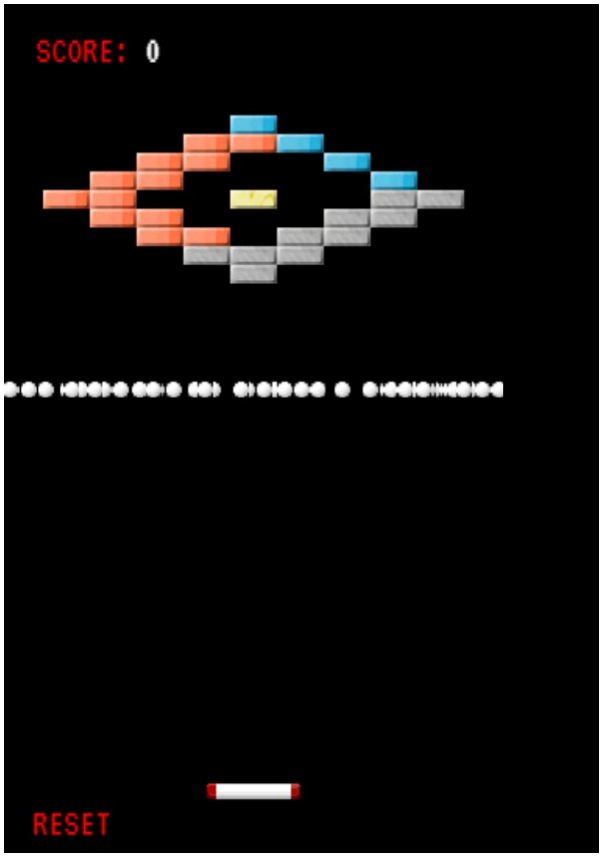
```
1  <!DOCTYPE HTML>
2  <html>
3  <head>
4      <meta charset="utf-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=no">
6      <meta http-equiv="Content-Security-Policy" content="default-src * data: gap: content: https://ssl.
       gstatic.com; style-src * 'unsafe-inline'; script-src * 'unsafe-inline' 'unsafe-eval'">
7      <script src="components/loader.js"></script><script src="classic.js"></script>
8      <link rel="stylesheet" href="components/loader.css">
9      <link rel="stylesheet" href="css/style.css">
10     <script>
11     </script>
12 </head>
13 <body>
14 </body>
15 </html>
```

メニューバー

- 保存
  - エディタの編集内容を保存します。
- 元に戻す＆やり直し
  - 編集内容を一つ戻したり戻した編集内容をやり直したりします。
- 編集
  - エディタの編集に関する機能があります。
- 表示
  - IDEの表示に関する機能があります。
- プロジェクト
  - プロジェクトの公開などに関する機能があります。
- ツール
  - お絵かきや録音などの機能があります
- 授業
  - 授業でよく使う、課題提出に関する機能などのショートカットがまとまっています。

プレビュー

- プログラムの実行結果が表示されます。
  - ※こちらはブロック崩しプロジェクトの例です。

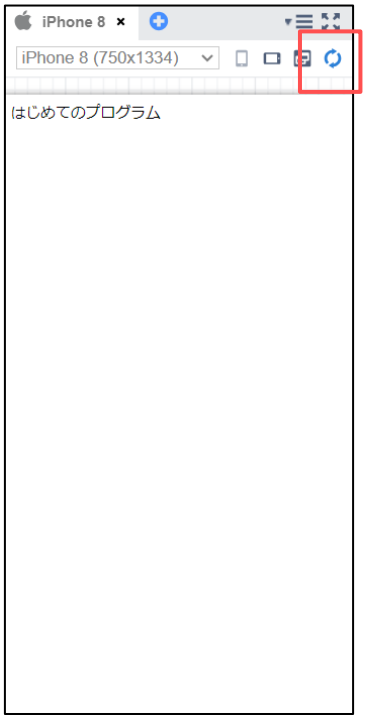


はじめてのプログラム

- <body>タグの中に「はじめてのプログラム」と記述します。
- メニューバーの［保存］アイコンをクリックします。

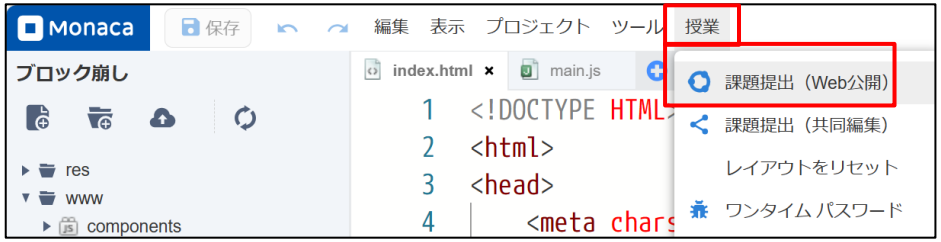
プレビューの更新

- ［保存］と連動してプレビュー画面が自動更新されます。
- 自動で更新されない場合
  - プレビュー画面右上の円状の矢印ボタンをクリックして下さい。



Web公開機能とは

- プロジェクトをWeb公開する機能です。
- プレビューはログイン中しか確認できないが、Web公開すればログイン不要で友達や先生にも作品を見せることができます。
- 注意点として、公開を「On」にして「適応する」必要があります。



アプリ制作教材（おみくじ）

■ アプリ制作教材（おみくじ）

- アプリ制作教材とは
  - アプリ制作（カスタマイズ）を通じて基礎を学習できる教材
  - コマ数は2～4コマ程度
- 教材の例：おみくじ
  - 変数・乱数・分岐・繰り返し・関数を学習可能
  - 1コマ目：おみくじアプリを動かす（変数・乱数・分岐）
  - 2コマ目：結果の追加（乱数・分岐）
  - 3コマ目：100回連続で引く(ガチャ対応)①（関数・繰り返し）
  - 4コマ目：100回連続で引く(ガチャ対応)②関数・繰り返し

1コマ目・おみくじアプリの仕組みを理解する

■ おみくじアプリを動かしてみよう

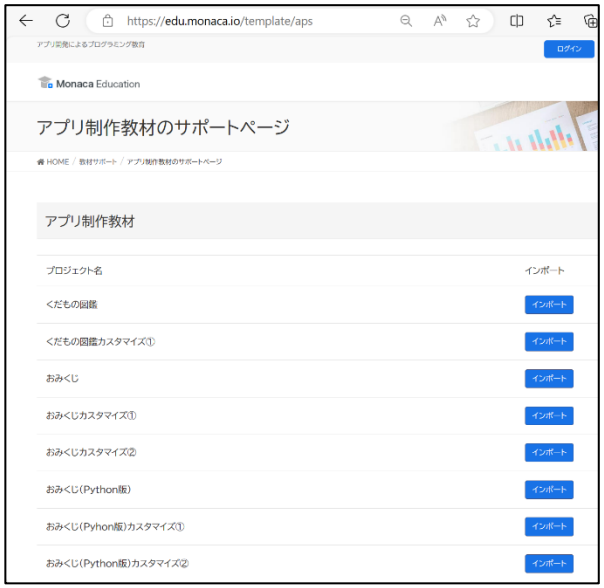
サポートページから
おみくじ(Python) をクリック

Monacaプロジェクトとしてインポートして IDEで開く

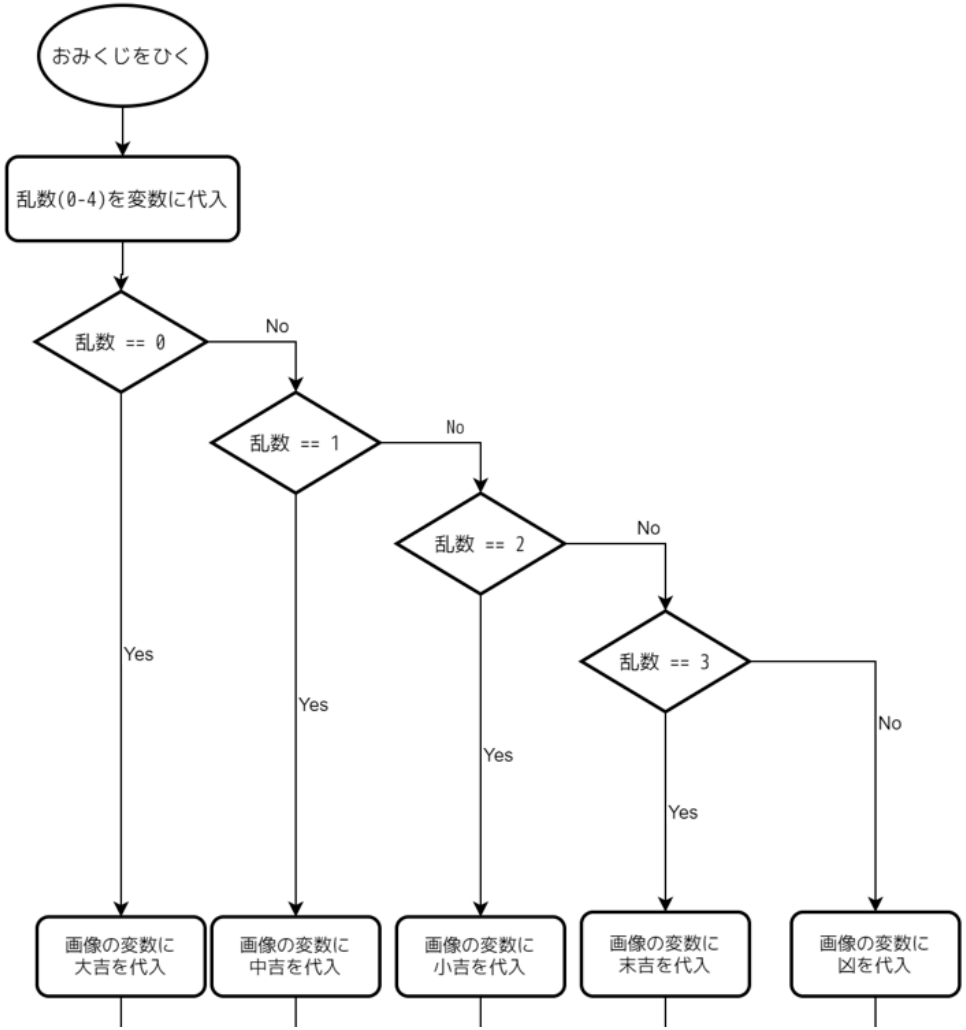
おみくじアプリとして
動作することを確認する

■ 学習者用・サポートページについて

└ インポート経由で入手可能です



■ おみくじアプリのフローチャート(抜粋)



■ おみくじアプリのソースコード(抜粋)

```
import random

# おみくじの処理
def play(e):
# 0から4までの整数をランダムに作り、変数noに代入する
  no = random.randint(0,4)
  if no == 0:
    image_name = 'daikichi.png'
  elif no == 1:
    image_name = 'chuukichi.png'
  elif no == 2:
    image_name = 'shoukichi.png'
  elif no == 3:
    image_name = 'suekichi.png'
  else:
    image_name = 'kyou.png'

  # 画面にメッセージを表示する
  alert('おみくじが出ました！さて結果は？')

  # 画像と文字列の差し替え
  document['omikuji'].src = 'img/' + image_name
  document['playBtn'].textContent = 'やりなおす'

# ボタンがクリックされたら、おみくじの処理を呼び出すように設定する
document['playBtn'].bind('click',play)
```

①乱数で結果を生成する

②結果に応じて画像を変える

③結果の表示

■ 動作確認：イベント・乱数・変数・分岐

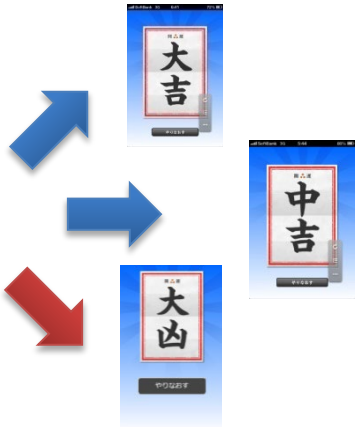
- ボタンを押下するたびに結果が変わることを確認する。
- 乱数の値に応じて条件分岐していることを確認する。
- random.ranint(0,4) の4の部分を2や10などに変更する
  - 『大吉』や『凶』がでやすくなること確認
    - else文の役割を確認
- ポイント
  - else文は、ifやelse ifの条件に当てはまらなかった場合に実行される
    - 乱数部分の値を4以上の数字にした場合はelse文が呼ばれる確率が高くなる。

■ 動作確認：イベント・関数

- ボタンが押されたときにプログラムが実行されることを確認
- play()関数の記述を確認
  - おみくじのプログラムの本体はplay()関数に書かれている
  - ボタン押下などのイベントに応じてplay()関数が呼び出されている

2コマ目・おみくじの結果を追加してみよう

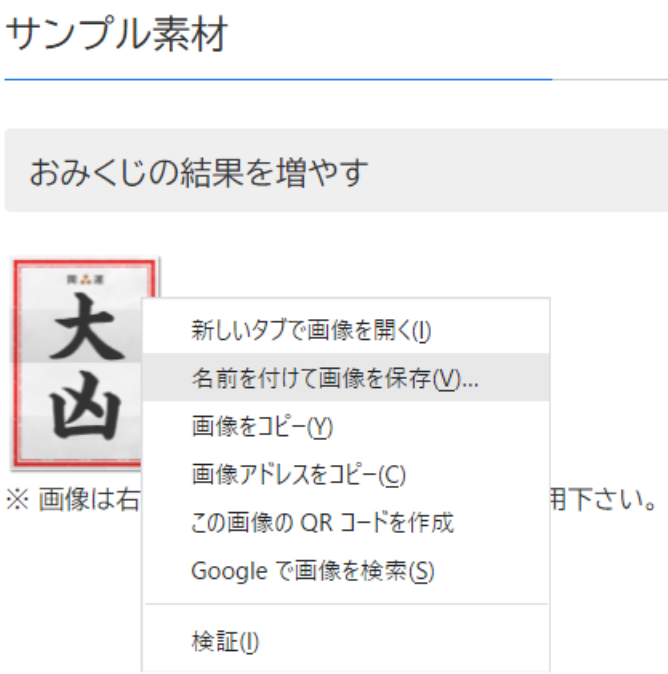
■ おみくじの結果を追加してみよう



- 乱数の範囲を1大きくする
- 乱数の結果によって
  『daikyo.png』が選ばれるようにする

■ 画像の取得

└ 何らかの方法で画像を入手して下さい



■ 画像のアップロード

└ /www/imgフォルダ内に画像をアップロード

└ 『ドラッグアンドドロップ』または『ファイル選択』

■ おみくじアプリのソースコード(抜粋)

```
import random

# おみくじの処理
def play(e):
# 0から5までの整数をランダムに作り、変数noに代入する
  no = random.randint(0,5)
  if no == 0:
    image_name = 'daikichi.png'
  elif no == 1:
    image_name = 'chuukichi.png'
  elif no == 2:
    image_name = 'shoukichi.png'
  elif no == 3:
    image_name = 'suekichi.png'
  elif no == 4:
    image_name = 'kyou.png'
  else:
    image_name = 'daikyou.png'
```

①乱数の範囲を0～5としたいので『5』に変更

②乱数が5以上なら大凶に変更

■ 実習

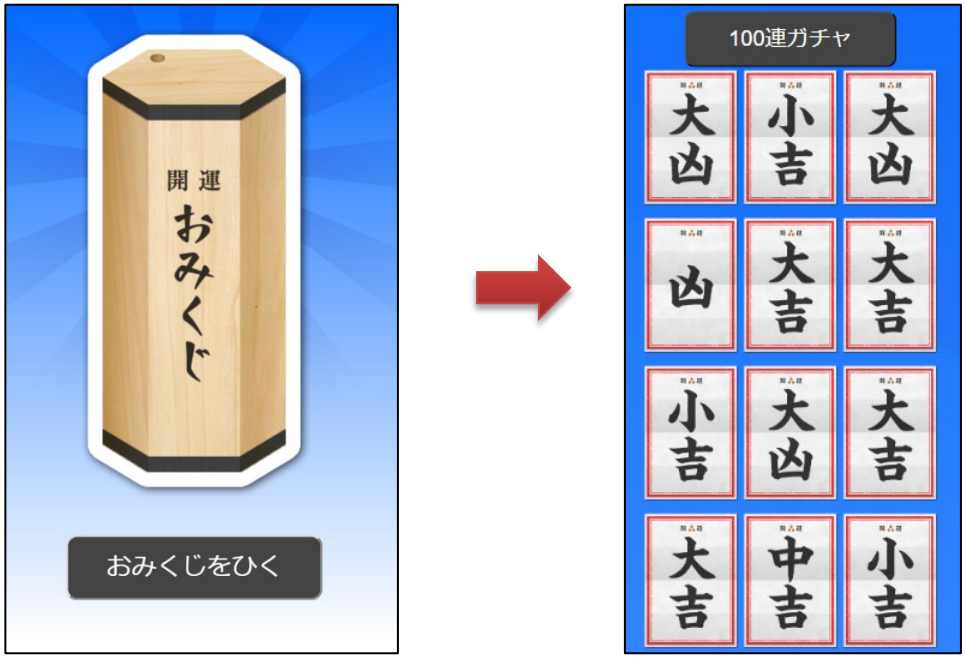
- 大凶の画像をアップロードする
- おみくじの結果に大凶を追加するカスタマイズを行う
  - 答え合わせ機能を使うと簡単
- 大凶が表示されるまでおみくじをひく

3コマ目・おみくじを100回ひいてみよう

■ 導入

- 繰り返しによるカスタマイズを行います
  - 関数の有用性を確認したり乱数や変数の仕組みを再確認して下さい

■ おみくじを100回ひいてみよう



大凶がでるまでボタンを押すのは大変なので、繰り返しの構文を利用して沢山のおみくじをひいてみましょう。

プログラムを変更（play関数の修正）

```
    # 画面にメッセージを表示する
##    alert('おみくじが出ました！さて結果は？')
    # 画像と文字列の差し替え
##    document['omikuji'].src = 'img/' + image_name
##    document['playBtn'].textContent = 'やりなおす'
```

①一部のプログラムをコメントアウト

```
    document['result'].innerHTML += '<img src="img/' + image_name + '">'
```

②ボタンの下にある結果を表示するための領域に、画像を追記する形で表示

プログラムを変更(gacha関数の追加)

```
# ガチャの処理
def gacha(e):
    daikichi = 0
    h = range(100)
    for i in h:
        kekka = play(e)
```

③play関数を複数回呼ぶ関数を追加

プログラムを変更(ボタンが押された時の処理)

```
document['playBtn'].bind('click',gacha)
```

④ボタンの文言と押したときに呼び出す関数を変更

プログラムを変更(HTML側のボタンの変更)

⑤ボタンの名称を変更

```
    <button id="playBtn">100連ガチャ</button>
```

4コマ目

導入

- 繰り返しと条件分岐を組み合わせたプログラムに挑戦します
  - 分岐と変数を応用して特定の場合の回数を数えることにも挑戦します

プログラムを変更（play関数の修正）

①関数を実行した時の戻り値として乱数を返せるように変更

```
return no;
```

プログラムを変更(gacha関数の修正)

```
def gacha(e):
    daikichi = 0
    loop = 100
    h = range(loop)
    for i in h:
        if (play(e) == 0) :
            daikichi = daikichi + 1
    document['message'].textContent = str(loop) + "回おみくじを引いたら大吉が" +
str(daikichi) + "回でました"
```

②大吉の数を数えるための変数を0で初期化
繰り返しの回数を変数に代入して利用するように変更

③play関数の結果が0なら大吉として値を1増やす

④おみくじを引いた回数と大吉の回数をボタン上のメッセージ領域に表示

試作問題3問に関連したプログラミング実習

問3：最小硬貨交換枚数

- 「釣り銭の計算」自体はよくある題材
  - よくあるケースは「客」が払う枚数を最小化させる
  - 51円を払うのには「50円1枚」と「1円1枚」
- プログラミングの基礎を応用させる題材として最適
- 試作問題の問いは「客」と「店」の枚数の最小化
  - 「初見」で「時間内」に解くのは難しい

問1　次の生徒（S）と先生（T）の会話文を読み，空欄 ア に当てはまる数字をマークせよ。また，空欄 イ ～ エ に入れるのに最も適当なものを，後の解答群のうちから一つずつ選べ。ただし，空欄 ウ ・ エ は解答の順序は問わない。

S：この前，お客さんが 460 円の商品を買うのに，510 円を払って，釣り銭を 50 円受け取っていたのを見て，授業で勉強したプログラミングで，そんな「上手な払い方」を計算するプログラムを作ってみたいと思いました。
T：いいですね。まず，「上手な払い方」とは何かを考える必要がありますね。
S：普通は手持ちの硬貨の枚数を少なくするような払い方でしょうか。
T：そうですね。ただ，ここでは，客が支払う枚数と釣り銭を受け取る枚数の合計を最小にする払い方を考えてみませんか？ 客も店も十分な枚数の硬貨を持っていると仮定しましょう。また，計算を簡単にするために，100 円以下の買い物とし，使う硬貨は 1 円玉，5 円玉，10 円玉，50 円玉，100 円玉のみで 500 円玉は使わない場合を考えてみましょう。例えば，46 円をち

出典：令和７年度大学入学共通テスト試作問題「情報」の6-2-I_試作問題『情報Ⅰ』※令和4年12月23日一部修正.pdf

問3：最小硬貨交換枚数

- 客が支払う枚数と釣り銭を受け取る枚数の合計を最小にする
- 問題中の例：46円を支払う場合

客	店	硬貨交換枚数
1円　x 1枚 5円　x 1枚 10円 x 4枚	0	6枚
50円 x 1枚	1円 x 4枚	5枚
50円 x 1枚 1円　x 1枚	5円 x 1枚	3枚

問3：関数 枚数（金額）

- このアルゴリズムは基礎に近いので事前に学習しておきたい
- 機能：最小の枚数を計算する関数
  - 最小硬貨交換枚数ではない
  - 客か店、どちらか片方の立場での最小枚数
- 引数：金額
- 戻り値：枚数

T：ここでは，次の関数のプログラムを作り，それを使う方法を考えてみまし

ょう。目標の金額を釣り銭無くちょうど支払うために必要な最小の硬貨枚数を求める関数です。

【関数の説明と例】

枚数(金額)… 引数として「金額」が与えられ，ちょうどその金額となる硬貨の組合せの中で，枚数が最小となる硬貨枚数が戻り値となる関数。
例：8 円は「5 円玉が 1 枚と 1 円玉が 3 枚」の組合せで最小の硬貨枚数になるので，**枚数**(8)の値は 4 となる。

出典：令和７年度大学入学共通テスト試作問題「情報」の6-2-I_試作問題『情報Ⅰ』※令和4年12月23日一部修正.pdf

問3：関数 枚数（金額）の実行例

- 枚数（8）の戻り値は4
  - 5円 + 1円 x 3枚
- 枚数（46）の戻り値は6
  - 10円 x 4枚 + 5円 + 1円
- 枚数(51)の戻り値は2
  - 50円 x 1円
- 枚数(5)の戻り値は1
  - 5円

【実習】関数枚数の処理を考えてみよう※別名：金種計算

- 硬貨(金種)
  - 1, 5, 10, 50, 100
- 処理のポイント
  - 大きな金種から割っていく
- 金種計算のポイント
  - 枚数（金額）は、全ての金種の枚数の合計を返す

Pythonでの実装例（試作問題の回答を参考）

- 配列のソートや配列の数を数える命令は出てこない
  - 配列の値を大きい順に利用していく
  - ループでは配列の値が大きい順に処理していく

```
import math

Kouka = [1,5,10,50,100]
kingaku = 46
maisu = 0
nokori = kingaku
for i in range(4,-1,-1):
    maisu = maisu + math.floor( nokori/Kouka[i] )
    nokori = nokori % Kouka[i]
print(maisu)
```

Monaca Educationで動かす場合の例：https://anko.education/archives/2979

試作問題

- Python版とあまり変わりません

```
(1) Kouka = [1,5,10,50,100]
(2) kingaku = 46
(3) maisu = 0, nokori = kingaku
(4) i を [キ] ながら繰り返す：
(5)     maisu = [ク] + [ケ]
(6)     nokori = [コ]
(7) 表示する(maisu)
```

図1　目標の金額ちょうどになる最小の硬貨枚数を計算するプログラム

[キ] の解答群

⓪ 5から1まで1ずつ減らし　① 4から0まで1ずつ減らし
② 0から4まで1ずつ増やし　③ 1から5まで1ずつ増やし

[ク] の解答群

⓪ 1　① maisu　② i　③ nokori

[ケ]・[コ] の解答群

⓪ nokori ÷ Kouka[i]　① nokori % Kouka[i]
② maisu ÷ Kouka[i]　③ maisu % Kouka[i]

出典：令和７年度大学入学共通テスト試作問題「情報」の6-2-I_試作問題『情報Ⅰ』※令和4年12月23日一部修正.pdf

問3：最小硬貨交換枚数を求める方法の考え方

- 客が枚数（51）払い、店が枚数（5）返す
  - 2枚+1枚＝3枚
- x = 商品の価格 = (x + y) + (x − y)
  - 46 = (46 + 0) + (46 − 0)
  - 46 = (46 + 99) + (46 − 99)

T：これは，例えば，**枚数**(46)＝［ア］と計算してくれるような関数です。これを使って最小交換硬貨枚数の計算を考えてみましょう。例えば，46 円支払うのに，51 円払って 5 円の釣り銭を受け取る払い方をした場合，客と店の間で交換される硬貨枚数の合計は，この関数を使うと，どのように計算できますか？

S：［イ］で求められますね。

T：一般に，商品の価格 x 円に対して釣り銭 y 円を 0, 1, 2, . . . と変化させて，それぞれの場合に必要な硬貨の枚数の合計を

枚数(［ウ］)＋**枚数**(［エ］)

と計算し，一番小さな値を最小交換硬貨枚数とすればよいのです。

S：なるほど。それで，釣り銭 y はいくらまで調べればよいでしょうか？

T：面白い数学パズルですね。まあ，詳しくは今度考えるとして，今回は 100 円以下の商品なので y は 99 まで調べれば十分でしょう。

［イ］の解答群

⓪ **枚数**(51) ＋ **枚数**(5)　① **枚数**(46) ＋ **枚数**(5)

② **枚数**(51) − **枚数**(5)　③ **枚数**(46) − **枚数**(5)

［ウ］・［エ］の解答群

⓪ x　① y　② $x + y$　③ $x - y$

出典：令和７年度大学入学共通テスト試作問題「情報」の6-2-1_試作問題『情報Ⅰ』※令和4年12月23日一部修正.pdf

問3：最小硬貨交換枚数の処理

- 金種計算（関数枚数）を客と店で2回使う
- 100円以下の商品・金種は100円までという前提
- ループ処理で0～99回シミュレーションする
  - └ もし最小なら変数min_maisuにmaisuを格納

```
(1) kakaku = 46
(2) min_maisu = 100
(3) [サ] を [シ] から 99 まで 1 ずつ増やしながら繰り返す：
(4)     shiharai = kakaku + tsuri
(5)     maisu = [ス] + [セ]
(6)     もし [ソ] < min_maisu ならば：
(7)         [タ] = [ソ]
(8) 表示する(min_maisu)
```

図２　最小交換硬貨枚数を求めるプログラム

サ, ソ, タ の解答群

⓪ maisu　① min_maisu　② shiharai　③ tsuri

シ の解答群

⓪ 0　① 1　② 99　③ 100

ス, セ の解答群

⓪ 枚数(shiharai)　① 枚数(kakaku)　② 枚数(tsuri)
③ shiharai　④ kakaku　⑤ tsuri

出典：令和７年度大学入学共通テスト試作問題「情報」の6-2-I_試作問題『情報Ⅰ』※令和4年12月23日一部修正.pdf

まとめ

- 問題をじっくり読めば解ける問題ではある
  - 関数枚数が穴埋めとはいえ用意されている
  - 『%』と『/』の違いも対話文で示されている
- 本番でじっくり読む時間があれば･･･
  - プログラミング入門レベルの基本文法は抑えておきたい
  - 入門レベルのみじかなアルゴリズムにも慣れておきたい

WaPEN@Asail紹介

■ 大学入学共通テスト実習対策講座

- 共通テスト手順記述標準言語(DNCL)とは
- 共通テスト対策実習の実施方法
- 画面・操作の説明

■ 共通テスト手順記述標準言語(DNCL)とは

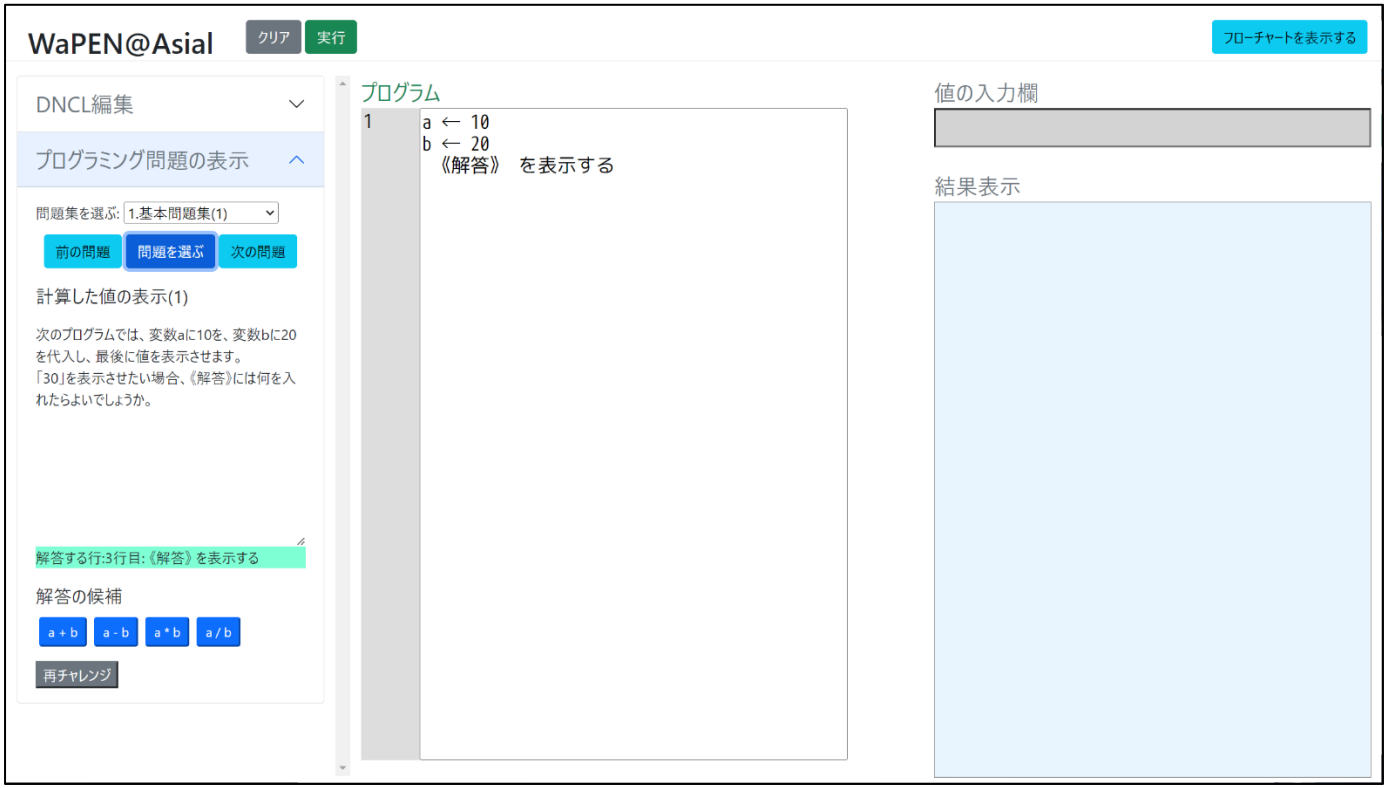
- 「大学入学共通テスト」で使われてきた「擬似言語」
  - 大学入試センターにて仕様が公開されている(2022)
    - 共通テスト手順記述標準言語（DNCL）の説明
    - 大学入学共通テスト
- DNCL終了のお知らせ
  - 令和7年度大学入学共通テスト　試作問題『情報Ⅰ』
    - 最新の試作問題では少し言語の文法が変わっています
    - 名称もDNCLではないものになります
      - 「共通テスト用プログラム表記」

■ WaPEN@Asialの文法について

- 2024年度現在、DNCLの仕様に準拠しています
  - 現時点で変更予定はありません
    - 予めご了承ください
  - 理由
    - 情報科の目的は擬似言語を学ぶことでは無いため
    - 本番でどのような言語仕様になるか不明なため
    - 最新の試作問題で示されている例が限定的
    - 新仕様の需要があれば対応を検討します

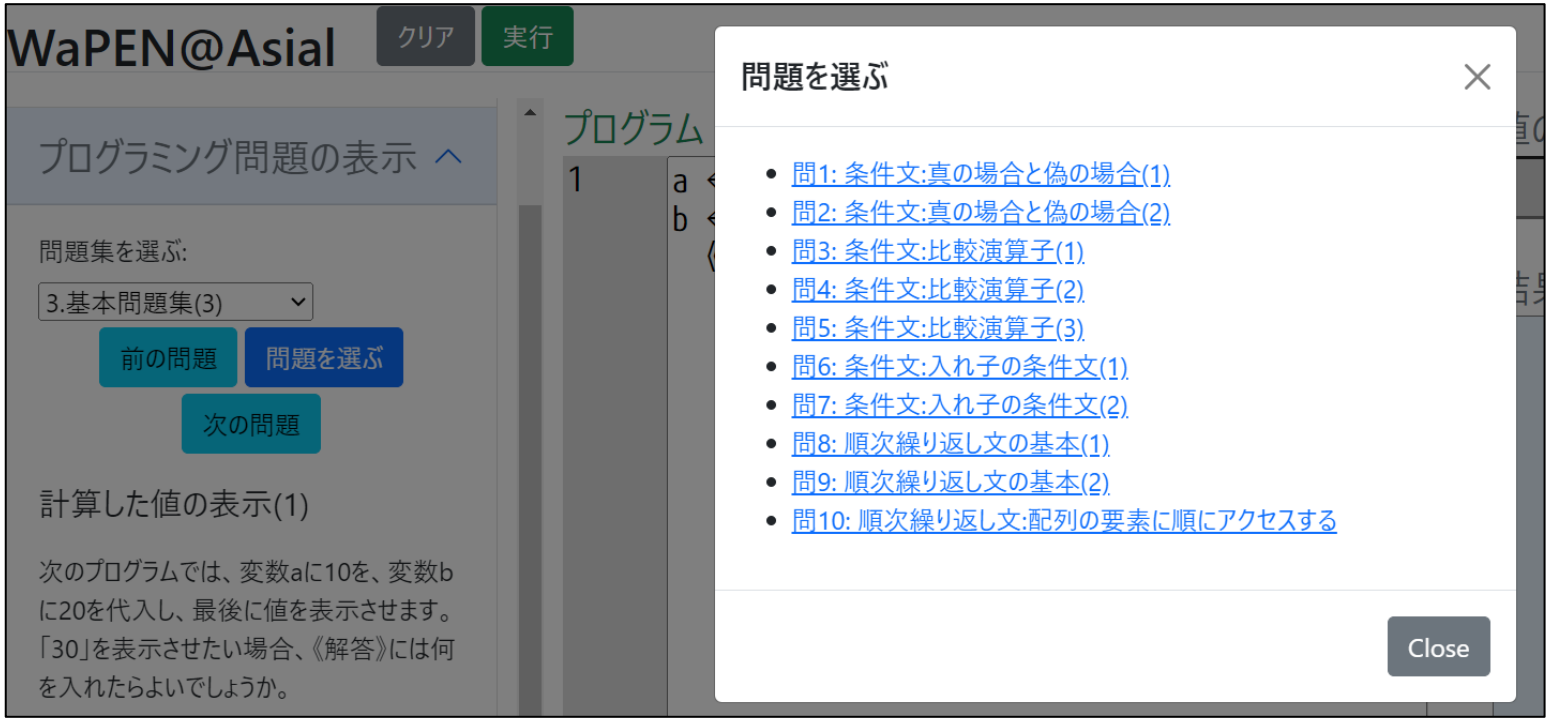
■ WaPEN@Asialの紹介

- ブラウザ上で動作(iPadも一応動作(デザイン崩れます))
- 問題が60問、組み込まれています(2024年度～)



■ 問題集が搭載されています

- 基礎問題40問、応用問題20問
- 入門レベルの文法や普遍的なアルゴリズムが題材



WaPEN@Asialの使い方

■ WaPEN@Asialの使い方（導線）

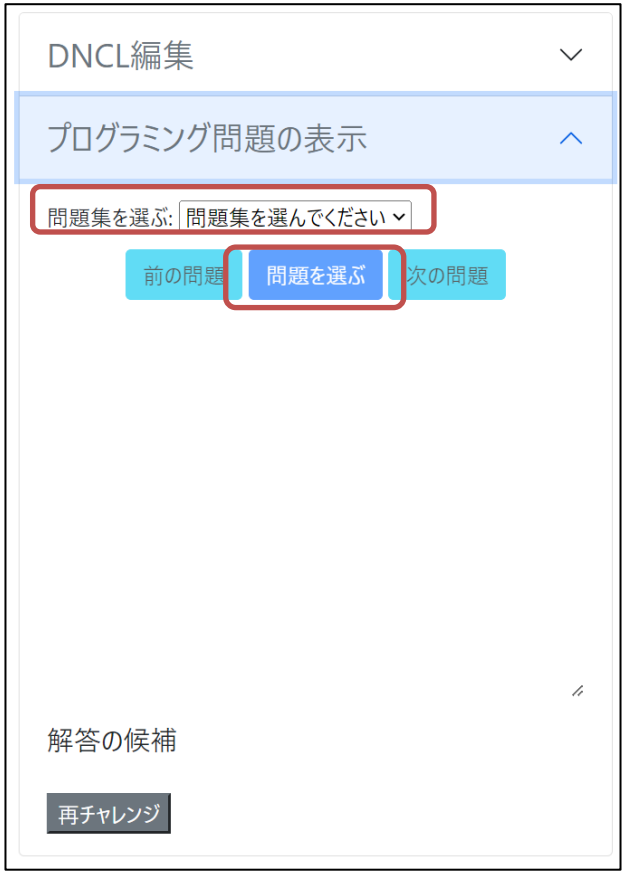
- クラウドIDEのリンク集から辿ってください
  - 生徒→自習用教材→大学入学共通テストDNCL対策教材（要パスワード）→WaPEN@Asialを利用する
    - 2024年6月時点の導線です
    - パスワードは「dncl」
- 夏休み期間中はリンク集から直接アクセスとする予定です
  - 生徒→WaPEN@Asial

■ WaPEN@Asialの使い方（基本的な流れ）

- 「プログラミング問題の表示」を選択

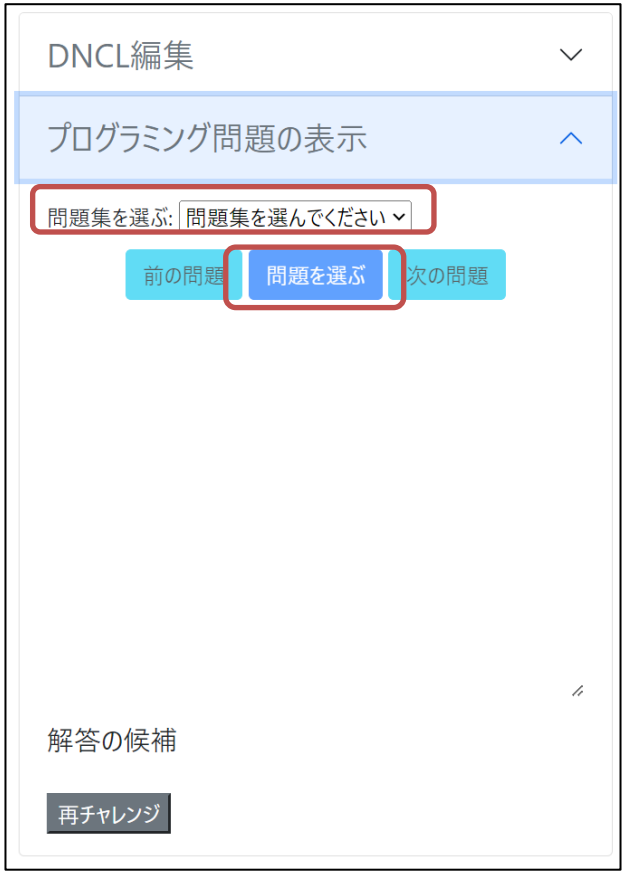
■ WaPEN@Asialの使い方（基本的な流れ）

- 「問題集を選ぶ」で問題集を選択
- 「問題を選ぶ」で問題を選択



■ WaPEN@Asialの使い方（基本的な流れ）

- 「問題集を選ぶ」で問題集を選択
- 「問題を選ぶ」で問題を選択



■ WaPEN@Asialの使い方（問題集の例）

・ 問題を選択します

■ WaPEN@Asialの使い方（問題集の例）

・ 問題文が表示されます

■ WaPEN@Asialの使い方（問題集の例）

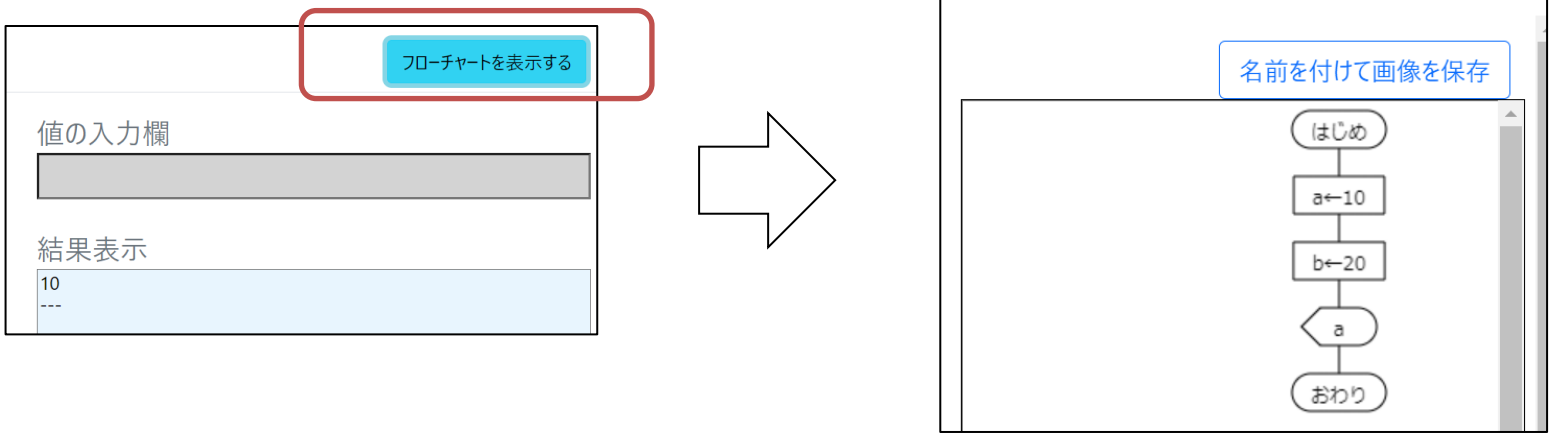
・ 解答の候補を選択するとソースコードに反映されます

■ WaPEN@Asialの使い方（問題集の例）

- 実行するとソースコードが実行されます
  - 結果表示には結果が表示されます

■ WaPEN@Asialの使い方（問題集の例）

- 「フローチャートを表示する」ことも可能です



【参考】プログラミング言語と文法について

■ プログラミング言語と文法について

- 手続き型言語では「基本三構造」が重要
  - 順次・分岐・繰り返し
  - これらを駆使すれば殆どのアルゴリズムが処理できます
- 入試対策として理解を深めておきたいポイント
  - 多少文法が揺れても「読める」力を身につける
    - 理想は、作品作りなどを通じて「使う」こと

■ 変数

- 値を持つ方法

	WaPEN@Asial	試作問題	Python
文法	□ ← □	□ = □	□ = □
例	Kingaku ← 77	Kingaku = 77	Kingaku = 77

■ 配列

- 複数の値を持つ方法

	WaPEN@Asial	試作問題	Python
文法	□ ← [□,□,□]	□ = [□,□,□]	□ = [□,□,□]
例	Kouka ← [1,5,10,50,100]	Kouka = [1,5,10,50,100]	Kouka = [1,5,10,50,100]

■ 分岐(最も単純な分岐)

・ 条件に一致したときに処理を実施します

	WaPEN@Asial	試作問題	Python
文法	もし □ ならば ｜ □ を実行する	もし □ ならば: └ □	if □ : □
例	nenrei ← 10 もし nenrei < 18 ならば ｜"未成年" を表示する を実行する	nenrei = 10 もし nenrei < 18 ならば: └ 表示する(未成年)	nenrei = 10 if nenrei < 18: print("未成年")

■ 分岐(2方向の分岐)

・ 条件に一致しなかったときにも別の処理を実施します

	WaPEN@Asial	試作題	Python

文法	もし □ ならば
	｜ □
	を実行し,そうでなければ
	｜ □
	を実行する
		????	if □ :
			  □
			else:
			  □

例	nenrei ← 10
	もし nenrei < 18 ならば
	｜ "未成年" を表示する
	を実行し,そうでなければ
	｜ "成人" を表示する
	を実行する
		????	nenrei = 10
			if nenrei < 18 :
			  print("未成年")
			else:
			  print("成人")

■ 繰り返し(for)

・一定の回数の範囲で処理を繰り返すパターン

	WaPEN@Asial	試作問題	Python
文法	□ を □ から □ まで □ ずつ増やしながら□ ｜□ を繰り返す	□を□から□まで1ずつ増やしながら繰り返す: └□ を実行する	for □ in range(□, □, □): □
例	gakunen を 1 から 3まで 1 ずつ増やしながら□ ｜gakunen を表示する を繰り返す	gakunen を 1 から 3まで 1 ずつ増やしながら: └ 表示する(gakunen)	for gakuen in range(1, 3, 1): print(gakunen)

■ 繰り返し(while)

・ 条件に一致している限りずっと処理を繰り返すパターン

	WaPEN@Asial	試作問題	Python
文法	□ の間,□ ｜ □ を繰り返す	?[?][?][?]	while: 　□
例	nenrei ← 0 jyumyou ← 100 nenrei < jyumyouの間,□ ｜nenrei ← nenrei + 1 ｜"誕生パ□ティ□"を表示する を繰り返す	?[?][?][?]	nenrei = 0 jyumyou = 100 while nenrei < jyumyou: print("誕生パ□ティ□")

おわりに

■ アンケートのお願い

・アンケートのご協力をお願いいたします

・https://forms.gle/tikBYwnp7Ahsik7Q9

■ トライアルキャンペーンのご案内

- Monaca Educationを授業でお試し頂けます

- https://edu.monaca.io/trial_2024about