繰り返し学ぶプログラミング

～３回に分けて段階的に学習活動を実施する～

世田谷学園中学校・高等学校
神藤　健朗（かんどう　たけあき）

← 2022年に使用した授業資料です。
QRコードを読み込んで確認してください。

自己紹介

世田谷学園 中学校 高等学校
SETAGAYA GAKUEN SCHOOL

神藤　健朗（かんどう　たけあき）

- **東京都内私立中高教諭として19年目（情報科）**
  - **世田谷学園中学校・高等学校 2021年～（3年目）**
    - 東京都市大学付属中学校・高等学校 2005年～（13年）
    - ドルトン東京学園開校準備＆中等部 2018年～（3年）
  - **芝浦工業大学 非常勤講師 2012年～（12年目）**
- **書籍等執筆**　日本文教出版 検定教科書
- **学歴**　数学教育（理学学士）⇒教育工学（学術修士）
- **職歴…コンピュータ関連の企業に約6年勤務**
  - 基幹業務システム開発（SE）1999年4月～
  - SE向け技術研修のインストラクター 2001年12月～
- **情報免許取得**
  - 一種免許状：北海道情報大学 科目等履修生（2004年）
  - （資格試験）：ソフトウェア開発技術者試験（2007年）
  - 専修免許状：放送大学大学院 修士専科生（2012年）

年間計画

～2022年度は以下の内容を実施しました～

世田谷学園 中学校 高等学校
SETAGAYA GAKUEN SCHOOL

1学期

4h｜過去の炎上・拡散事件

5h｜こんなことできません動画作成

2h｜n進法/論理演算
足し算/引き算/掛け算/小数

3h｜アルゴリズム
カップラーメンと並べ替え

2h｜ネットワーク基礎※
PC教室のネットワーク調査

1h｜n進法 補数表現

2学期

7h｜ガチャ＆とどラン
※一部の内容を1学期に実施

3h｜ピクトグラム作成
実験室のピクトグラムを作成

5h｜HTML+CSS
Webデザインの基本を確認

7h｜プログラミング※
ガチャ・おみくじ・じゃんけん

これらの合間に、
小テスト・ビーバーチャレンジ
などを行っています。

3学期

9h｜1～2学期の内容を復習
各1h｜さまざまなAI体験～個人情報とは～暗号化のしくみ～さまざまな圧縮の方法～ POSシステムのしくみ～自宅のネットワーク調査～音と画像のデジタル化～プログラミング※（2h）

2h｜プログラミング※
並べ替え・二分探索・n進法など

2h｜シミュレーション
待ち行列・感染モデル

1h｜n進法 浮動小数点/誤差

共通テストへの対応

世田谷学園 中学校 高等学校
SETAGAYA GAKUEN SCHOOL

～身近な面白い題材を積極的に取り入れた授業展開をすることが大切～

１．新教育課程による出題教科・科目　（7）情報　『情報Ⅰ』

日常的な事象や社会的な事象などを情報とその結び付きとして捉え，情報と情報技術を活用した問題の発見・解決に向けて探究する活動の過程，及び情報社会と人との関わりを重視する。

問題の作成に当たっては，**社会や身近な生活の中の題材**，及び**受験者にとって既知ではないもの**も含めた資料等に示された事例や事象について，情報社会と人との関わりや情報の科学的な理解を基に考察する力を問う問題などとともに，問題の発見・解決に向けて考察する力を問う問題も含めて検討する。

出典：独立行政法人大学入試センター　令和７年度大学入学者選抜に係る大学入学共通テスト問題作成方針（令和5年6月9日）
https://www.dnc.ac.jp/kyotsu/shiken_jouhou/r7/　（2023/7/19 18:26閲覧）

■知識及び技能

■思考力、判断力、表現力等

■学びに向かう力、人間性等

画像：かわいいフリー素材集いらすとや、https://www.irasutoya.com/

共通テストへの対応

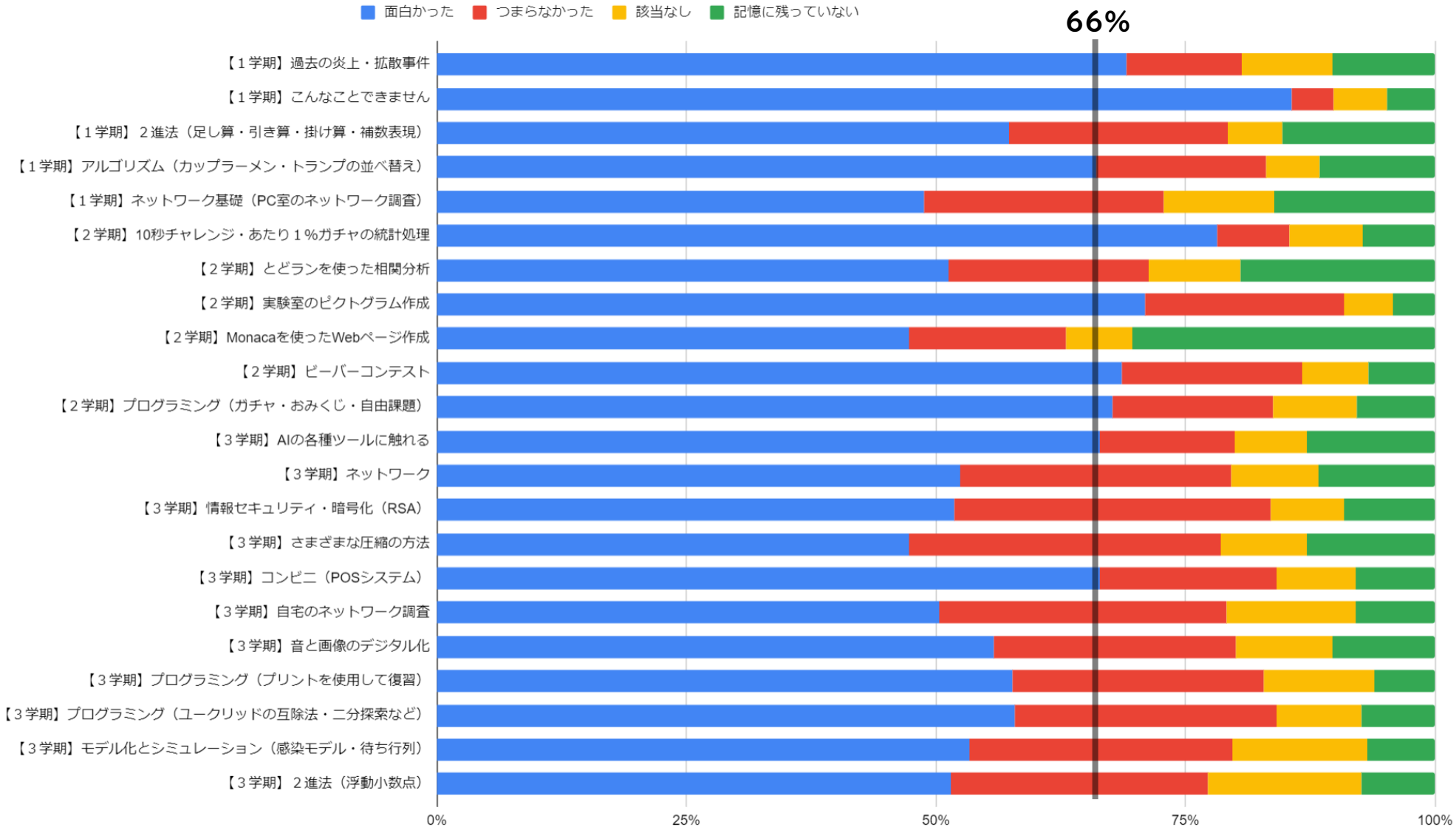
～情報Ⅰ（全12種）の教科書の索引に共通して収録された語句～

IoT	産業財産権	ビット
IPアドレス	シミュレーション	標本化
LAN	情報デザイン	ファイアウォール
OS	人工知能	符号化
POSシステム	知的財産権	プレゼンテーション
TCP/IP	著作権	フローチャート
圧縮	データ	プログラム
アルゴリズム	データベース	変数
解像度	テキストマイニング	メディア
階調	ドメイン名	メディアリテラシー
可逆圧縮	ハードウェア	モデル化
画素	パスワード	量子化
個人情報	非可逆圧縮	量的データ

出典：中園長新（麗澤大学）、高等学校「情報Ⅰ」教科書の索引に掲載された語句の傾向、2022PCカンファレンス口頭発表

授業アンケートの結果[N=165]

～生徒にとって身近な学習活動は面白かったという反応が多い～



まとめ

世田谷学園 中学校 高等学校
SETAGAYA GAKUEN SCHOOL

～身近な面白い題材を用いた授業を行い、生徒の記憶に残る授業～

- **詰め込んだ知識は、2年後（1年後）忘れている可能性大**
  - 忘れている前提で授業計画を立てる
  - 身近な面白い題材を使って授業し、知識につなげる方がお互いに幸せ
    - 稲垣先生（都立神代高校）『自分事』
  - 試験の場で「こんなことやったな」ってのを思い出せるような授業
- **生徒が定期的に復習するための工夫**
  - 1年生の定期テストを2年生（3年生）に配布する
    - 定期テストの過去問を1年生に配布
  - 大学入学共通テスト「情報Ⅰ」体験模試の受験を促す
- **情報収集として研究会への参加**
  - **8月上旬～中旬**　全国高等学校情報教育研究会全国大会（愛知大会）
  - **12月26日**　神奈川県情報科実践事例報告会
  - キミのミライ発見（河合塾）　https://www.wakuwaku-catch.net/

具体的な授業事例を紹介

例２：プログラミング

～「コンピュータとプログラミング」分野を実施したタイミング～

世田谷学園 中学校 高等学校
SETAGAYA GAKUEN SCHOOL

1学期

- 4h｜過去の炎上・拡散事件
- 5h｜こんなことできません動画作成
- 2h｜n進法/論理演算
  足し算/引き算/掛け算/小数
- 3h｜アルゴリズム
  カップラーメンと並べ替え
- 2h｜ネットワーク基礎※
  PC教室のネットワーク調査
- 1h｜n進法　補数表現

2学期

- 7h｜ガチャ＆とどラン
  ※一部の内容を1学期に実施
- 3h｜ピクトグラム作成
  実験室のピクトグラムを作成
- 5h｜HTML+CSS
  Webデザインの基本を確認
- 7h｜プログラミング※
  ガチャ・おみくじ・じゃんけん

これらの合間に、
小テスト・ビーバーチャレンジ
などを行っています。

3学期

- 9h｜1～2学期の内容を復習
  各1h｜さまざまなAI体験～個人情報とは～暗号化のしくみ～さまざまな圧縮の方法～ POSシステムのしくみ～自宅のネットワーク調査～音と画像のデジタル化～プログラミング※（2h）
- 2h｜プログラミング※
  並べ替え・二分探索・n進法など
- 2h｜シミュレーション
  待ち行列・感染モデル
- 1h｜n進法　浮動小数点/誤差

授業の流れ（計３時間）

～まずはじめに、アルゴリズムについて考える～

世田谷学園 中学校 高等学校
SETAGAYA GAKUEN SCHOOL

■学習活動を２時間で構成

カップラーメンのアルゴリズムを考える	カップラーメンのアルゴリズムを箇条書きにして記述する。（フローチャートは煩雑なので、まずは手順を表現する）
並べ替えのアルゴリズムを考える	トランプを使用して、10枚の並べ替え作業を考える。誰が読んでも正しく並べ替えが行えるような手順を記述する。

■教科書やWeb資料を使った情報整理を１時間で構成

並べ替えのアルゴリズムまとめ	様々な並べ替えのアルゴリズムを紹介する。自分のアルゴリズムと比較して手を動かして確認を行う。

1時間目
カップラーメンのアルゴリズム

アルゴリズムとは

- コンピューターが効率的に問題を解いたり、課題を解決したりするための処理手順。◇アルゴリズムをプログラミング言語を用いて具体的に記述したものがプログラム。
  - アルゴリズムとは - コトバンク (kotobank.jp)
- プログラムとは
  - 一般にはテレビ、ラジオなどの番組や催し物の番組表などをさすが、コンピュータ用語としては、コンピュータに行わせる処理の手順を、決められた形式（プログラム言語）に従って書き表したものをいう。
    - プログラムとは - コトバンク (kotobank.jp)
- 本日の動画
  - 流れを操る｜１０ｍｉｎ．ボックス　テイクテック｜NHK for School

本日の課題

～丁寧に書きすぎないよう注意し、最低限のポイントはおさえること～

- **ロボットが調理できるようにカップラーメンの作成手順を書き出してみよう。ロボットがミスなく調理できるように内容や手順に注意して記述してみよう。**
  - 作業１：やかんを使ってお湯を沸かす手順を記述する
  - 作業２：お湯がやかんに入っている状態でカップヌードルを作る
  - 作業３：お湯がやかんに入っている状態で赤いきつねを作る
    - ふたを開けると、スープの素と薬味（七味）が入っていることに注意が必要
  - 作業４（＋α）：お湯がやかんに入っている状態でカップ焼きそばを作る
    - ふたを開けると、かやく（乾燥野菜や肉）とソースが入っていることに注意が必要
- **評価基準**
  - 作業１～作業４それぞれについて以下の基準で評価を行う
    - 手順がもれなく書かれており、手順通りに行えば完成させることができる（or　ケガをしない）
    - 手順にもれがあり、その通りに作業を行っても完成させることができない（or　ケガをする）
    - 未完成（未入力に近い状態）
  - 低学年の弟や妹に調理手順を教えるようなイメージで書き出してみてください。

やかんにお水を入れる手順

～色々と表現の仕方がありますよね～

■細かすぎても大変

1. やかんのふたを取る
2. ふたをテーブルに置く
3. やかんを蛇口の下までもっていく
4. 水道の栓をひねって水を出す
5. やかんの半分まで水が入っているか確認
6. 半分まで水が入ったら、水道の栓をひねって水を止める
7. やかんをコンロの上に置く
8. テーブルの上のふたを取る
9. やかんにふたをする

■簡単すぎてもダメ

1. やかんに水を入れる

どの程度水を入れたらいいか
判断がつかない

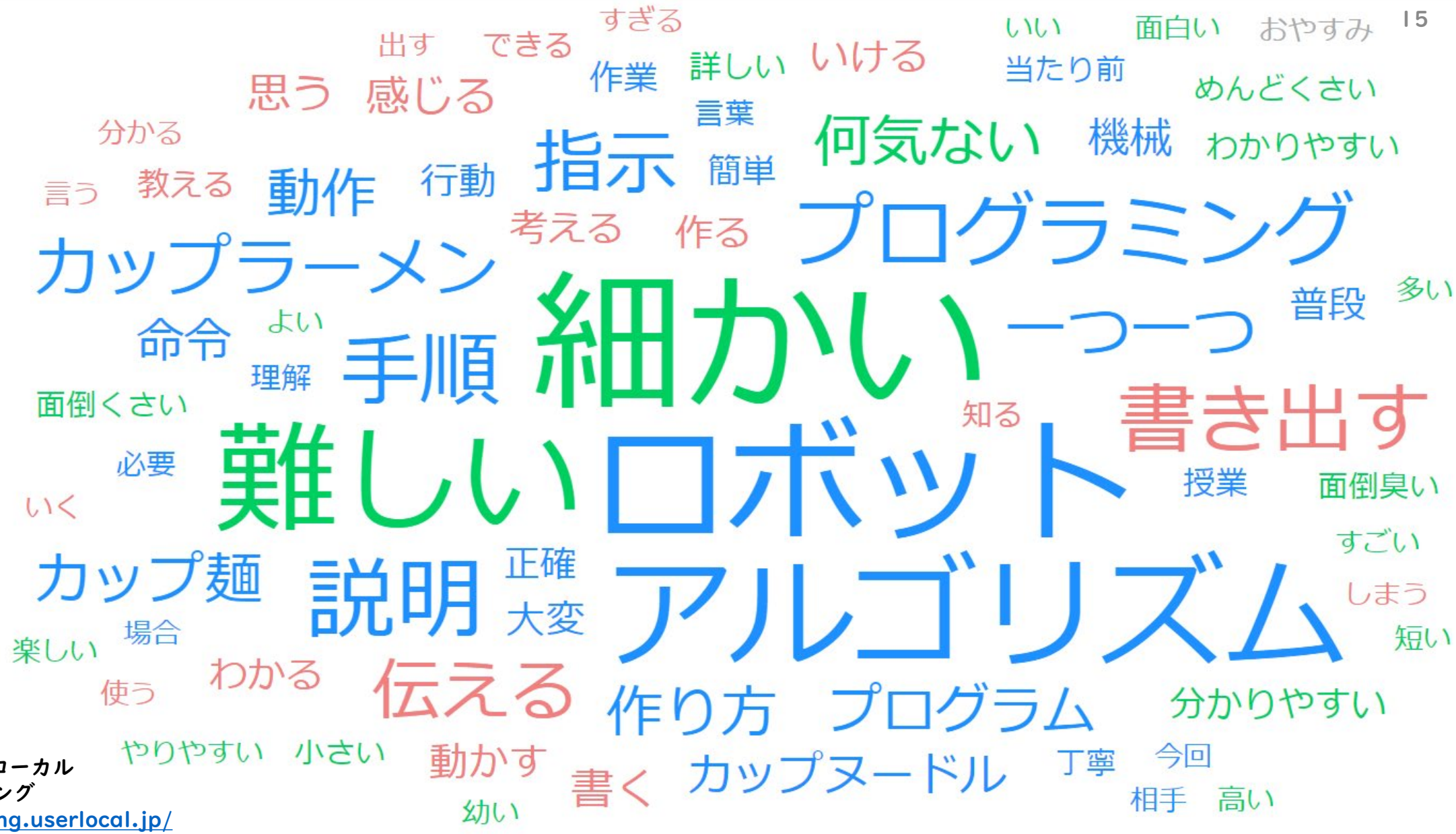
■必ず条件を入れるように！

1. やかんのふたを取る
2. やかんの半分まで水を入れる
3. やかんにふたをする

水の分量が具体的にわかるように
記載してください

1時間目：リフレクション

2022



出典：株式会社ユーザーローカル
AIテキストマイニング
https://textmining.userlocal.jp/

２時間目
並べ替えのアルゴリズム

お湯を沸かすアルゴリズム

～大きな流れは同じですが、細かさが異なります～

① やかんに溢れない程度に水を入れる。
② ガスコンロの上にやかんを置く。
③ ガスコンロを中火で点火する。
④ やかんから水蒸気が出たらSwitchを押して火を消す。

① やかんを右手で持つ
② やかんを蛇口の下のシンクに置いて、手を離す
③ 蛇口を反時計回りに90度回す
④ 水がやかんの3分の2入るまで待つ
⑤ 蛇口を時計回りに90度回す(水が止まるまで)
⑥ やかんをガスコンロの上に置く
⑦ やかんのふたを載せる
⑧ やかんが載っているガスコンロの側のスイッチを火が出るまで押さえて離す
⑨ 水が沸騰する音が聞こえるまで待つ
⑩ やかんが載っているガスコンロの側のスイッチを火が消えるまで押さえて離す

お湯を沸かすアルゴリズム

～コンピュータの処理は、「順次・分岐・繰り返し」の３種類～

① やかんに溢れない程度に水を入れる。
② ガスコンロの上にやかんを置く。
③ ガスコンロを中火で点火する。
④ やかんから水蒸気が出たらSwitchを押して火を消す。

順次
並んだ順番に処理を行う。1つの処理が終わったら、次の処理を行う。

条件分岐・判断分岐
〇〇なら、〇〇のとき、など条件によって判断を行う。その後、それぞれに対応する処理を行う。

繰り返し
〇分間、〇回、〇〇まで、などの条件に従って、処理を繰り返す。

並べ替えのアルゴリズムを考える

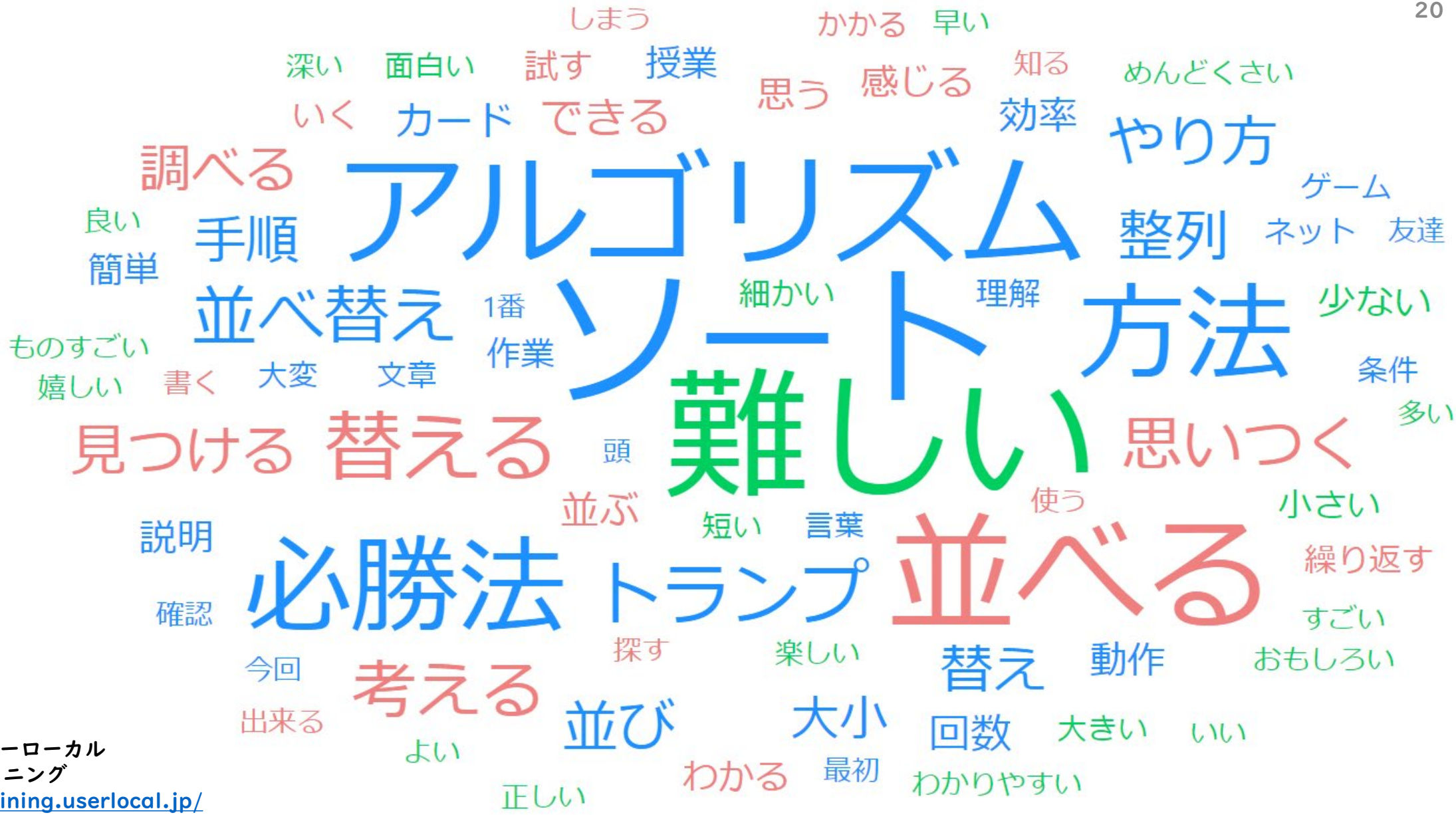
カードの枚数が変わっても並べ替えられるアルゴリズムを考えること

- **作業内容**
  - トランプを１０枚配るので、裏返した状態で横に並べる
  - 最終的にカードを大きい順（小さい順）になるように並べる
- **作業の条件**
  - 一度に表にできるのは２枚のみ（カードの数字を覚えてはいけない）
  - カードにしるしをつけてはいけない（ただし、分類するのは可能）
  - なるべく表に返す回数が少なくなるように
    - カード２枚を表にする作業を１回とカウントして、何回の操作で並べ替えできますか？
- **提出してほしいもの**
  - 自分（自分たち）の考えた並べ替えのアルゴリズムをレポートにまとめて提出
    - ほかの人が読んでも作業方法がわかるように手順（アルゴリズム）を記入すること
  - 自分で方法を編み出してもOK・ネットを参考にしてもOK
    - 評価は変わりません。あくまでも**正しく並べ替えられる手順**を導き出してください。

２時間目：リフレクション

世田谷学園 中学校 高等学校
SETAGAYA GAKUEN SCHOOL

2022



出典：株式会社ユーザーローカル
AIテキストマイニング
https://textmining.userlocal.jp/

３時間目
アルゴリズムまとめ

いろいろなソートアルゴリズム①

さまざまな方法があるので整理して紹介

- バブルソート
  - 泡が浮き上がるように並べ替えを行う方法
  - https://yutaka-watanobe.github.io/star-aida/1.0/algorithms/bubble_sort/anim.html
  - バブルソート - Wikipedia
- シェーカーソート
  - バブルソートを応用して、行ったり来たりしながら並べ替えを行う方法
  - https://yutaka-watanobe.github.io/star-aida/1.0/algorithms/shaker_sort/anim.html
  - シェーカーソート - Wikipedia
- 挿入ソート
  - 1つずつどこに並べればよいか確認しながら、順番に並べ替えを行う方法
  - https://yutaka-watanobe.github.io/star-aida/1.0/algorithms/insertion_sort/anim.html

いろいろなソートアルゴリズム②

高速な並べ替え方法について

- マージソート
  - ピタゴラスイッチの「しめじソート」の処理
  - 小さなグループで並べ替えを行い、小さなグループを併合（マージ）しながら大きくすることで効率よく並べ替えを行う方法
  - 比較回数は、$O(N \log_2 N)$
  - https://yutaka-watanobe.github.io/star-aida/1.0/algorithms/merge_sort/anim.html
  - マージソート - Wikipedia
- クイックソート
  - ピタゴラスイッチの「じゃがいもソート」の処理
  - 1つ基準を決めて、基準より大きいグループ・小さいグループに分ける操作を繰り返しながら並べ替えを行う方法
    - 基準の選び方によっては非効率になってしまう点に注意が必要
  - 比較回数は、$O(N \log_2 N)$
  - https://yutaka-watanobe.github.io/star-aida/1.0/algorithms/quick_sort/anim.html
  - クイックソート - Wikipedia

本日の作業

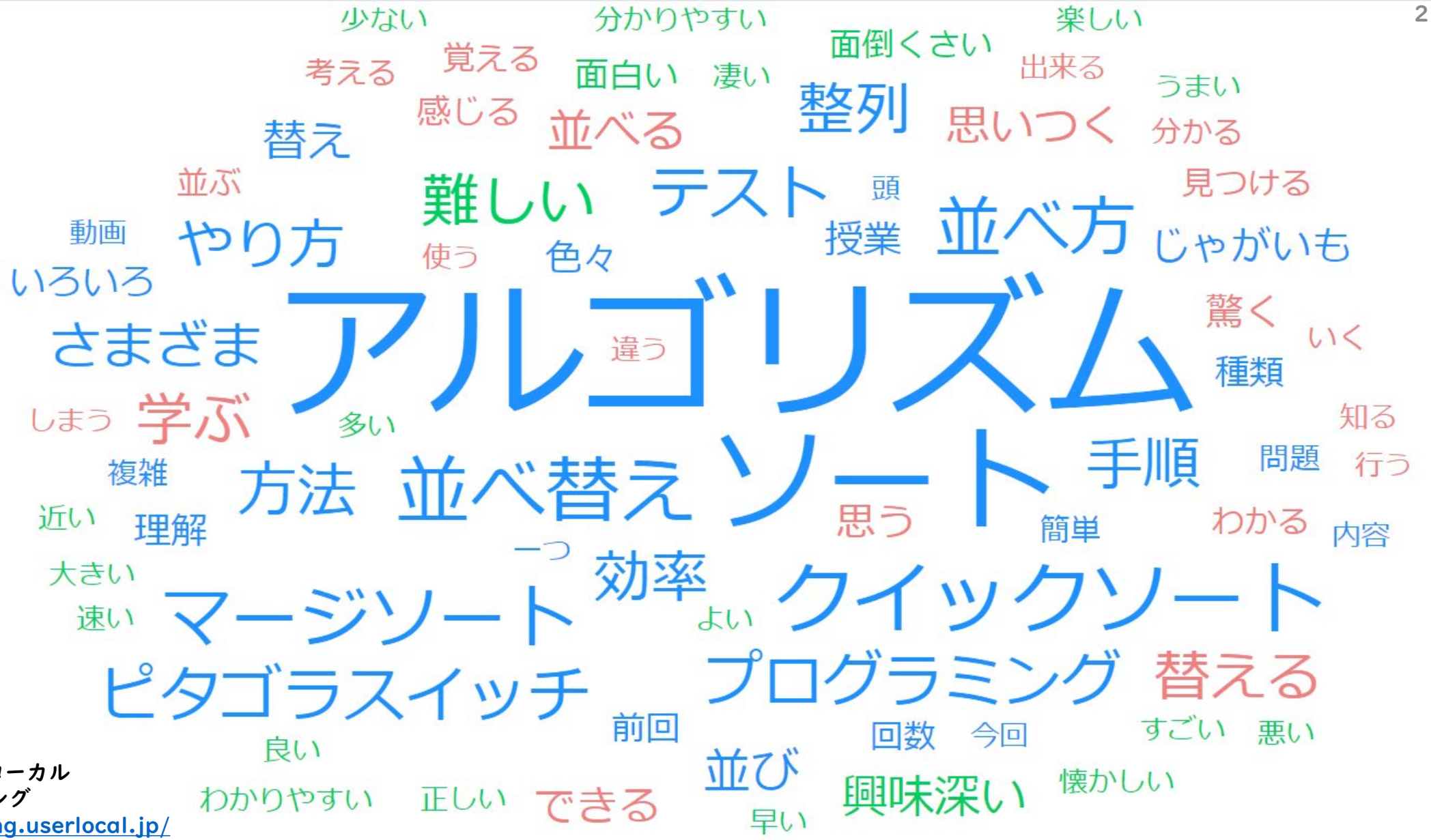
実際にいろいろな並べ替えの方法を試してみよう

- 整列（ソート）ゲームのWebページを使って、８枚のカードを何回で並べ替えることができるかチャレンジしてみよう
- 最小回数は何回にできるか、また最小回数になる並べ替えの方法はどのような方法かチャレンジしてみよう
  - https://javalab.org/ja/sorting_game_ja/

- 小テストの実施
  - アルゴリズムに関する小テストを実施する

３時間目：リフレクション

2022



出典：株式会社ユーザーローカル
AIテキストマイニング
https://textmining.userlocal.jp/

まとめ

～箇条書きでも手順をあらわすのは難しかった…～

- **手順をあらわすのに集中させた点が良かった**
  - フローチャートで書かせると、フローチャートの記述方法の正しい/間違えているに視点が行ってしまう
- **試行錯誤しながらも、ある程度正しいソートアルゴリズムを探し出すことができていた**
  - ３時間目の振り返りのときに、自分のアルゴリズムの検証を行うことができた
  - フローチャートの書き方に視点が行かず、手順に注力できたのは良かった
- **箇条書きでも手順をあらわすのは難しかった**
  - 手順の本質を考えさせることができてよかった
  - 保育園児の兄弟やロボットに手順を伝えるように細かく考えてみるように伝えることで、必要な情報と不要な情報を考えるように指示した
- **トランプを使用せず、以下のページを使って操作を記述させるのもあり**
  - https://javalab.org/ja/sorting_game_ja/
  - トランプの図柄や色に惑わされずに並べ替えの作業に注力できる

例２：授業の流れ（計7時間）

～Pythonを使ったプログラミング～

※PyTry https://pro-ktmr.github.io/pytry/

①あたり1%ガチャ（100連ガチャを200回）	データ分析の授業で扱った、あたり1%ガチャのプログラムを作成する	実行環境：PyTry、順次処理、繰り返し、条件分岐、配列（リスト）
②効率の良い硬貨の使い方（試作問題：第3問）	共通テスト「情報Ⅰ」試作問題：第3問を8つのステップで作成する	実行環境：PyTry、順次処理、繰り返し、条件分岐、関数
③おみくじ作成（演習）	2回の授業を振り返り、おみくじを引くプログラムを作成する	実行環境：PyTry、順次処理、繰り返し、条件分岐、配列（リスト）
④じゃんけん（基本形の作成）	じゃんけんの基本形を前回までと同様に4つのステップで作成し、その後基本形を改造することでオリジナルのプログラムを作成する（PyTryは双方向性のあるプログラムを作成できないため、後半はGoogle Colaboratoryを使用する）	実行環境：（前半）PyTry →（後半）Google Colaboratory 順次処理、繰り返し、条件分岐、配列（リスト）
⑤⑥基本形を改造してじゃんけんを作成（自由作成）		
⑦まとめ	副教材の問題を解きながら理解を深める	

1時間目
ガチャのプログラミング

ガチャのプログラミング

～順を追って作成すれば難しくなく簡単に作業をすることができます～

- **第1段階**
  - 乱数を使って１～１００までの任意の数字を表示する
- **第2段階**
  - 更に、任意の数字を１００回表示する（１００連ガチャにする）
- **第3段階**
  - 更に、「あたり」「はずれ」を表示する
- **第4段階、第5段階**
  - 更に、あたった回数を数えて表示する
- **第6段階**
  - 更に、１００連ガチャを２００回引いて最終的に１００回のうち何回当たったか集計を行う

PyTryの使用方法

https://pro-ktmr.github.io/pytry/

プログラムの実行ボタン

PyTry

実行

ソースコード コピー

整数1個：int(input()) 整数複数個：map(int, input().split()) 文字列：input() 整数リスト：list(ma

```
1  # 第1段階：乱数を使って１～１００までの任意の数字を表示する
2  import random
3
4  r = random.randint(1, 100)
5  print(r)
6
```

プログラムの入力領域

入力 ペースト

```
1
```

プログラムへ渡す値を入力する領域

出力

```
1  53
2
```

実行結果の表示領域
※エラーの表示領域

一部機能に制限はありますが、**エラーメッセージが日本語**に翻訳されるので、慣れるまでは非常に便利です。

PyTry をご利用いただきありがとうございます．近頃 PyTry を想定以上に

ガチャのプログラミング

～第１段階：１～１００のランダムな数字を出力する～

PyTry

実行

ソースコード　コピー

整数1個：int(input())　整数複数個：map(int, input().split())　文字列：

```
1  import random
2
3  r = random.randint(1, 100)
4  print(r)
5
```

入力　ペースト

1

必ず半角（英数モード）で入力
大文字小文字は区別する

```
1  import random
2
3  r = random.randint(1, 100)
4  print(r)
```

乱数の利用

～importを使用して、randomモジュールを読み込む～

■代表的な乱数の命令

- ■random.randint(開始値,終了値)
  - ■開始値から終了値までの整数を出力
- ■random.random()
  - ■０以上１未満の実数を出力

```
1   import random
2
3   r1 = random.randint(1, 100)
4   print(r1)
5   r2 = random.random()
6   print(r2)
```

出力

```
1   77
2   0.2757059200242023
3
```

■変数

- ■値や文字を保存しておくための名前を付けた箱

■代入

- ■変数に値や文字を保存する処理
  - ■変数 ＝ 値　と表記する

```
3   r1 = random.randint(1, 100)
```

  - ■数学の＝と異なり、左辺の値を右辺の変数に代入する
  - ■数学の＝は、＝＝で表現する

■出力

- ■print(出力したい変数や文字列)
  - ■変数を指定すると中身が出力される

ガチャのプログラミング

～第２段階：１～１００のランダムな数字を100回表示する～

PyTry

実行

ソースコード　コピー

整数1個：int(input())　整数複数個：map(int, input().split())　文字列：input()

```
1  import random
2
3  for i in range(100):
4      r = random.randint(1, 100)
5      print(r)
6
```

入力　ペースト

1

```
1  import random
2
3  for i in range(100):
4      r = random.randint(1, 100)
5      print(r)
```

半角空白でインデント（頭揃え）を設定する
※これがないと正しく実行できない！

繰り返し（反復構造）

～for文とwhile文を使うと、同じ命令を繰り返すことができる～

■繰り返したい部分はインデントでまとめる（字下げして頭をそろえる）

```
1    import random
2
3    for i in range(100):
4        r = random.randint(1, 100)
5        print(r)
6
```

コロンを忘れない！

インデント

「1～100の乱数を求めて出力する」
処理が100回繰り返される

■for 変数 in range(回数):

- ■回数に指定した数だけ繰り返される
  - ■補足：変数には、0,1,2,3,…,回数-1　の値が順番に代入される

ガチャのプログラミング

～第３段階：「あたり」「はずれ」を表示する～

PyTry

ソースコード　コピー

整数1個：int(input())　整数複数個：map(int, input().split())

```
1  import random
2
3  for i in range(100):
4      r = random.randint(1, 100)
5      print(r)
6      if r == 1:
7          print("あたり")
8      else:
9          print("はずれ")
10
```

ダブルクォーテーション(”)で囲まれた文字がそのまま表示される
※全角文字(日本語)も表示できる

```
1  import random
2
3  for i in range(100):
4      r = random.randint(1, 100)
5      print(r)
6      if r == 1:
7          print("あたり")
       else:
           print("はずれ")
```

条件分岐（選択構造）

～場合分けを行う方法～

■条件分岐（選択構造）

- ■比較演算の結果、真（True）の場合に実行したい処理、偽（False）の場合に実行したい処理を記述することができる

- ■記述方法
  - ■インデント□を忘れないように注意
  - ■□は半角空白（またはタブ：Tab）

```
if 比較演算:
□□比較演算がTrueのときの処理①
□□比較演算がTrueのときの処理②
else:
□□比較演算がFalseのときの処理①
□□比較演算がFalseのときの処理②
```

■比較演算

- ■2つの値の大小関係や等値関係を判定します
- ■結果は、真理値の真（True）または偽（False）で判定されます

AはB～	表記方法
等しい	A == B
等しくない	A != B
より大きい	A > B
より小さい	A < B
以上	A >= B
以下	A <= B

ガチャのプログラミング

世田谷学園 中学校 高等学校
SETAGAYA GAKUEN SCHOOL

～第４段階：あたった回数を数える変数を用意して、回数を管理する～

PyTry

ソースコード　コピー

整数1個：int(input())　整数複数個：map(int, input().split())

```
1   import random
2
3   atari = 0
4   for i in range(100):
5       r = random.randint(1, 100)
6       print(r)
7       if r == 1:
8           print("あたり")
9           atari = atari + 1
10      else:
11          print("はずれ")
12  print(atari)
13
```

```
1   import random
2
3   atari = 0
4   for i in range(100):
5       r = random.randint(1, 100)
6       print(r)
7       if r == 1:
8           print("あたり")
9           atari = atari + 1
10      else:
11          print("はずれ")
12  print(atari)
13
```

ガチャのプログラミング

～第５段階：カンマで区切ると複数の情報を表示（出力）できる～

PyTry

ソースコード　コピー

整数1個：`int(input())`　整数複数個：`map(int, input().split())`

```
1   import random
2
3   atari = 0
4   for i in range(100):
5       r = random.randint(1, 100)
6       if r == 1:
7           print(r, "あたり")
8           atari = atari + 1
9       else:
10          print(r, "はずれ")
11  print(atari, "回あたり")
12
```

```
1   import random
2
3   atari = 0
4   for i in range(100):
5       r = random.randint(1, 100)
6       if r == 1:
7           print(r, "あたり")
8           atari = atari + 1
9       else:
10          print(r, "はずれ")
11  print(atari, "回あたり")
12
```

ガチャのプログラミング

世田谷学園 中学校 高等学校
SETAGAYA GAKUEN SCHOOL

～第６段階：100連ガチャを200回引いた時の結果を集計する～

PyTry

ソースコード　コピー

整数1個：int(input())　整数複数個：map(int, input().split())　文字列：input()

```
1   import random
2
3   Kekka = [0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
4   for j in range(200):
5       atari = 0
6       for i in range(100):
7           r = random.randint(1, 100)
8           if r == 1:
9               atari = atari + 1
10      print(atari, "回あたり")
11      Kekka[atari] = Kekka[atari] + 1
12  print(Kekka)
13
```

```
1   import random
2
3   Kekka = [0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
4   for j in range(200):
5       atari = 0
6       for i in range(100):
7           r = random.randint(1, 100)
8           if r == 1:
9               atari = atari + 1
10      print(atari, "回あたり")
11      Kekka[atari] = Kekka[atari] + 1
12  print(Kekka)
13
```

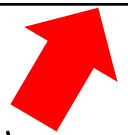
配列（リスト）

～複数のデータを一つにまとめて管理することができる仕組み～

■配列（リスト）

- ■部屋（**要素**）ごとにデータを管理できる
- ■部屋番号（**添字**）で位置を管理する

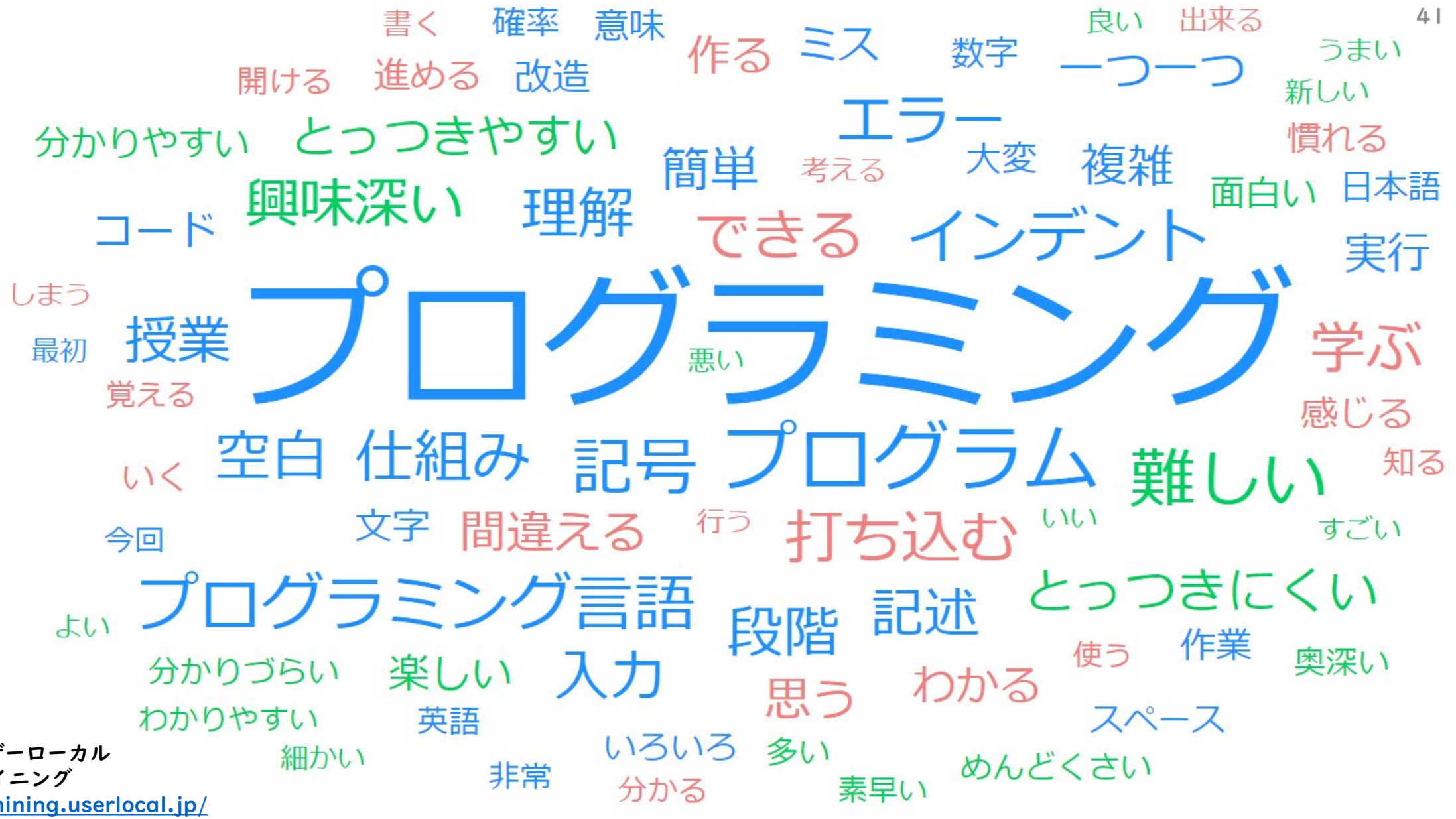
	0	1	2	3	4	5	6	7
Kekka	89	59	36	14	1	1	0	0



- ■配列への代入は添字で指定する
  - ■Kekka[4] = 3　　※4番の要素に3を代入
- ■表示したい場合はprintと組み合わせて使用する
  - ■print(Kekka)　　⇒　[89,59,36,14,1,1,0,0]　　※配列全部の要素を出力する
  - ■print(Kekka[3])　　⇒　14　　※3番目の要素のみ出力する
- ■要素の追加はappendを使用する
  - ■Kekka.append(5)　　←Kekkaの末尾に5を追加する

1時間目：リフレクション

2022



出典：株式会社ユーザーローカル
AIテキストマイニング
https://textmining.userlocal.jp/

２時間目
コインの枚数のプログラミング

コインの枚数のプログラミング

～試作問題で出題されたプログラミングを行ってみよう～

世田谷学園 中学校 高等学校 SETAGAYA GAKUEN SCHOOL

第３問　次の問い（**問１**～**３**）に答えよ。（配点　25）

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 円をちょうど支払う場合，支払う枚数はどうなりますか？

S：46 円を支払うには，10 円玉 4 枚，5 円玉 1 枚，1 円玉 1 枚という 6 枚で払い方が最小の枚数になります。

T：そうですね。一方，同じ 46 円を支払うのに，51 円を支払って釣り銭 5 円を受け取る払い方では，支払いに 2 枚，釣り銭に 1 枚で，合計 3 枚の硬貨のやり取りになります。こうすると交換する硬貨の枚数の合計が最小になりますね。

S：これが上手な払い方ですね。

T：そうです。このように，客と店が交換する硬貨の合計が最小となる枚数，すなわち「最小交換硬貨枚数」の計算を考えましょう。

S：どうやって考えればいいかなぁ。

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。

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 $x - y$

問２　次の文章の空欄［オ］～［コ］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。

S：まずは，関数「**枚数**(金額)」のプログラムを作るために，与えられた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を考えてみます。もう少しヒントが欲しいなぁ。

T：金額に対して，高額の硬貨から使うように考えて枚数と残金を計算していくとよいでしょう。また，金額に対して，ある額の硬貨が何枚まで使えて，残金がいくらになるかを計算するには，整数値の商を求める演算『÷』とその余りを求める演算『%』が使えるでしょう。例えば，46 円に対して 10 円玉が何枚まで使えるかは［オ］で，その際にいくら残るかは［カ］で求めることができますね。

S：なるほど！あとは自分でできそうです。

S さんは，先生（T）との会話からヒントを得て，変数 **kingaku** に与えられた目標の金額（100 円以下）に対し，そ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を考えてみた（図 1）。ここでは例として目標の金額を 46 円としている。

配列 **Kouka** に硬貨の額を低い順に設定している。なお，配列の添字は **0** から始まるものとする。最低額の硬貨が 1 円玉なので **Kouka[0]** の値は **1** となる。

先生（T）のヒントに従い，高額の硬貨から何枚まで使えるかを計算する方針で，**(4)**～**(6)** 行目のような繰返し文にした。この繰返しで，変数 **maisu** に支払いに使う硬貨の枚数の合計が計算され，変数 **nokori** に残りいくら支払えばよいか，という残金が計算される。

実行してみると［ア］が表示されたので，正しく計算できていることが分かる。いろいろな例で試してみたが，すべて正しく計算できていることを確認できた。

コインの枚数のプログラミング

～順を追って作成すれば難しくなく簡単に作業をすることができます～

- **第1段階**
  - 50円玉を使う枚数と残金を求める
- **第2段階**
  - 50円玉と10円玉を使う枚数と残金を求める
- **第3段階**
  - 更に、「合計枚数」と「最終残金」を表示する
- **【省略可能】第4段階（クイズ：第３段階から何を変更すれば対応できるか？）**
  - 更に、5円玉と1円玉を使う枚数と残金を求める
- **第5段階**
  - 配列（リスト）と繰り返しを用いて50円玉～1円玉までの使用枚数を求める
- **第6段階（クイズ：第５段階から何を変更すれば対応できるか？）**
  - さらに、500円玉と100円玉も含めて使用枚数を求める
- **第7段階**
  - 関数を作成して効率よく処理する
- **第8段階**
  - 共通テストの問題に沿ったプログラムを作成する

PyTryの使用方法

https://pro-ktmr.github.io/pytry/

プログラムの実行ボタン

PyTry　実行

ソースコード　コピー

整数1個：int(input())　整数複数個：map(int, input().split())　文字列：input()　整数リスト：list(ma

```
1  # 第1段階：乱数を使って１～１００までの任意の数字を表示する
2  import random
3
4  r = random.randint(1, 100)
5  print(r)
6
```

プログラムの入力領域

入力　ペースト

```
1
```

プログラムへ渡す値を入力する領域

出力

```
1  53
2
```

実行結果の表示領域
※エラーの表示領域

一部機能に制限はありますが、**エラーメッセージが日本語**に翻訳されるので、慣れるまでは非常に便利です。

コインの枚数のプログラミング

～第１段階：50円玉を使う枚数と残金を求める～

世田谷学園 中学校 高等学校
SETAGAYA GAKUEN SCHOOL

PyTry

ソースコード　コピー

整数1個：int(input())　整数複数個：map(int, input().split())　文字列：input()

```
1  # 第１段階：50円玉を使う枚数と残金を求める
2  kingaku = 90
3  nokori = kingaku
4  coin = 50
5  print(coin, "円の枚数", nokori // coin)
6  print("残金", nokori % coin)
7
```

・必ず半角（英数モード）で入力
・大文字小文字は区別する
・ダブルクォーテーション（”）で囲まれた文字は全角入力OK
・シャープ（#）で書かれた行はコメント（プログラムとして実行されない説明文）


```
1  # 第１段階：50円玉を使う枚数と残金を求める
2  kingaku = 90
3  nokori = kingaku
4  coin = 50
5  print(coin, "円の枚数", nokori // coin)
6  print("残金", nokori % coin)
7
```

変数の利用と算術演算記号

～記憶させるために変数を用意して、記憶したい値を代入する～

- 変数
  - 値や文字を保存しておくための名前を付けた箱
- 代入
  - 変数に値や文字を保存する処理
    - 変数 = 値　と表記する
    - 左辺の値を右辺の変数に代入する

```
2   kingaku = 90
3   nokori = kingaku
4   coin = 50
```

    - 数学の＝は、＝＝で表現する
- 出力
  - print(出力したい変数や文字列)
    - 変数を指定すると中身が出力される
    - カンマで区切って複数の情報を出力

- 算術演算記号
  - 数学記号との違いに注意

	記号	使用例	計算結果
足し算（＋）	＋	7+3	10
引き算（－）	-	7-3	4
掛け算（×）	*	7*3	21
割り算（÷）	/	7/3	2.33…
割り算の商	//	7//3	2
割り算の余り	%	7%3	1
数値の〇乗	**	7**3	343

コインの枚数のプログラミング

～第２段階：50円玉と10円玉を使う枚数と残金を求める～

PyTry 実行

ソースコード コピー

整数1個：int(input()) 整数複数個：map(int, input().split()) 文字列：input() 整数リスト：li

```
1  # 第２段階：50円玉と10円玉を使う枚数と残金を
2  kingaku = 90
3  nokori = kingaku
4  # 50円玉を使う枚数
5  coin = 50
6  print(coin, "円の枚数", nokori // coin)
7  print("残金", nokori % coin)
8  nokori = nokori % coin
9  # 10円玉を使う枚数
10 coin = 10
11 print(coin, "円の枚数", nokori // coin)
12 print("残金", nokori % coin)
13 nokori = nokori % coin
14
```

入力 ペースト

```
1
```

出力

```
1  50 円の枚数 1
2  残金 40
3  10 円の枚数 4
4  残金 0
5
```

コインの枚数のプログラミング

～第３段階：さらに「合計枚数」と「最終残金」を表示する～

ソースコード　コピー

整数1個：int(input())　整数複数個：map(int, input().split())　文字列：input()　整数

```
1   kingaku = 90
2   maisu = 0
3   nokori = kingaku
4   # 50円玉を使う枚数
5   coin = 50
6   print(coin, "円の枚数", nokori // coin)
7   maisu = maisu + nokori // coin
8   print("残金", nokori % coin)
9   nokori = nokori % coin
10  # 10円玉を使う枚数
11  coin = 10
12  print(coin, "円の枚数", nokori // coin)
13  maisu = maisu + nokori // coin
14  print("残金", nokori % coin)
15  nokori = nokori % coin
16  # 硬貨を使う枚数
17  print("合計枚数", maisu)
18  print("最終残金", nokori)
```

入力　ペースト

【重要】7行目と13行目の変数への代入の方法
左辺の計算結果を右辺に代入する

左辺：現在の枚数＋コインを使う枚数
7行目：0枚 ＋ 90円÷50円
13行目：1枚 ＋ 40円÷10円

出力

```
1   50 円の枚数 1
2   残金 40
3   10 円の枚数 4
4   残金 0
5   合計枚数 5
6   最終残金 0
7
```

コインの枚数のプログラミング

～第４段階：50円玉～1円玉までの使用枚数を求める～

ソースコード　コピー

整数1個：int(input())　整数複数個：map(int, input().split())　文字列：input()　整数リスト：li

```
12  coin = 10
13  print(coin, "円の枚数", nokori // coin)
14  maisu = maisu + nokori // coin
15  print("残金", nokori % coin)
16  nokori = nokori % coin
17  # 5円玉を使う枚数
18  coin = 5
19  print(coin, "円の枚数", nokori // coin)
20  maisu = maisu + nokori // coin
21  print("残金", nokori % coin)
22  nokori = nokori % coin
23  # 1円玉を使う枚数
24  coin = 1
25  print(coin, "円の枚数", nokori // coin)
26  maisu = maisu + nokori // coin
27  print("残金", nokori % coin)
28  nokori = nokori % coin
29  # 硬貨を使う枚数
30  print("合計枚数", maisu)
31  print("最終残金", nokori)
32
```

ソフトウェアキーボードで作業すると大変なので、
第5段階の作業を行っても問題ありません。

同じことを繰り返しているだけ!
⇒繰り返しを使えばもっとプログラムを短くできる

出力

```
1   50 円の枚数 1
2   残金 48
3   10 円の枚数 4
4   残金 8
5   5 円の枚数 1
6   残金 3
7   1 円の枚数 3
8   残金 0
9   合計枚数 9
10  最終残金 0
```

コインの枚数のプログラミング

～第５段階：配列と繰り返しを使って効率よく処理する～

世田谷学園 中学校 高等学校
SETAGAYA GAKUEN SCHOOL

```
1   Kouka = [50, 10, 5, 1]
2   kingaku = 98
3   maisu = 0
4   nokori = kingaku
5
6   # 配列と繰り返しを使って効率
7   for i in range(4):
8       coin = Kouka[i]
9       print(coin, "円の枚数"
10      maisu = maisu + nokor
11      print("残金", nokori %
12      nokori = nokori % coi
13
14  # 硬貨を使う枚数
15  print("合計枚数", maisu)
16  print("最終残金", nokori)
17
```

```
1   Kouka = [50, 10, 5, 1]
2   kingaku = 98
3   maisu = 0
4   nokori = kingaku
5
6   # 配列と繰
7   for i in range(4):
8       coin = Kouka[i]
9       print(coin, "円の枚数", nokori // coin)
10      maisu = maisu + nokori // coin
11      print("残金", nokori % coin)
12      nokori = nokori % coin
```

「変数iの値を0～3に変更しながら繰り返す」

繰り返し（反復構造）

～for文とwhile文を使うと、同じ命令を繰り返すことができる～

■繰り返したい部分はインデントでまとめる（字下げして頭をそろえる）

```
 7  for i in range(4):
 8      coin = Kouka[i]
 9      print(coin, "円の枚数", nokori // coin)
10      maisu = maisu + nokori // coin
11      print("残金", nokori % coin)
12      nokori = nokori % coin
```

コロンを忘れない！

インデント

■for 変数 in range(回数):

■回数に指定した数だけ繰り返される

■補足：変数には、0,1,2,3,…,回数−1　の値が順番に代入される

赤枠の処理が4回繰り返される
※変数iに0,1,2,3が順番に代入される

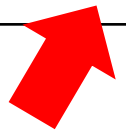
配列（リスト）

～複数のデータを一つにまとめて管理することができる仕組み～

■配列（リスト）

- ■部屋（**要素**）ごとにデータを管理できる
- ■部屋番号（**添字**）で位置を管理する

Kouka	0	1	2	3
	50	10	5	1



- ■配列への代入は添字で指定する
  - ■Kouka[3] = 2　　※3番の要素に2を代入
- ■表示したい場合はprintと組み合わせて使用する
  - ■print(Kouka)　　⇒　[50,10,5,1]　　※配列全部の要素を出力する
  - ■print(Kouka[1])　　⇒　10　　※1番目の要素のみ出力する
- ■要素の追加はappendを使用する
  - ■Kouka.append(5)　　←Koukaの末尾に5を追加する

コインの枚数のプログラミング

～第６段階：500円玉～１円玉までの使用枚数を求める～

世田谷学園 中学校 高等学校
SETAGAYA GAKUEN SCHOOL

```
1   Kouka = [500, 100, 50, 10, 5, 1]
2   kingaku = 128
3   maisu = 0
4   nokori = kingaku
5
6   # 配列と繰り返しを使って効率よく処理する
7   for i in range(6):
8       coin = Kouka[i]
9       print(coin, "円の枚数", nokori // coin)
10      maisu = maisu + nokori // coin
11      print("残金", nokori % coin)
12      nokori = nokori % coin
13
14  # 硬貨を使う枚数
15  print("合計枚数", maisu)
16  print("最終残金", nokori)
17
```

第6段階で行った作業
・配列の中身を追加
・繰り返しの回数を変更

出力

```
1   500 円の枚数 0
2   残金 128
3   100 円の枚数 1
4   残金 28
5   50 円の枚数 0
6   残金 28
7   10 円の枚数 2
8   残金 8
9   5 円の枚数 1
10  残金 3
```

コインの枚数のプログラミング

～第７段階：関数を作成して効率よく処理する～

世田谷学園 中学校 高等学校
SETAGAYA GAKUEN SCHOOL

```python
def Mai(kingaku):
    Kouka = [500, 100, 50, 10, 5, 1]
    maisu = 0
    nokori = kingaku
    # 配列と繰り返しを使って効率よく処理する
    for i in range(6):
        coin = Kouka[i]
        print(coin, "円の枚数", nokori // coin)
        maisu = maisu + nokori // coin
        print("残金", nokori % coin)
        nokori = nokori % coin
    return maisu


# 硬貨を使う枚数
print("合計枚数", Mai(752))
```

出力

```
500 円の枚数 1
残金 252
100 円の枚数 2
残金 52
50 円の枚数 1
残金 2
10 円の枚数 0
```

コインの枚数のプログラミング

～第８段階：共通テストの問題に沿ったプログラムを作成する～


```python
def Mai(kingaku):
    Kouka = [500, 100, 50, 10, 5, 1]
    maisu = 0
    nokori = kingaku
    # 配列と繰り返しを使って効率よく処理する
    for i in range(6):
        coin = Kouka[i]
        maisu = maisu + nokori // coin
        nokori = nokori % coin
    return maisu


kakaku = 46
min_maisu = 100
for tsuri in range(100):
    shiharai = kakaku + tsuri
    maisu = Mai(shiharai) + Mai(tsuri)
    if maisu < min_maisu:
        min_maisu = maisu
print("合計枚数", min_maisu)
```

【注意】関数の中のprint命令はすべて削除する
⇒大量に出力されるため実行できない

```python
kakaku = 46
min_maisu = 100
for tsuri in range(100):
    shiharai = kakaku + tsuri
    maisu = Mai(shiharai) + Mai(tsuri)
    if maisu < min_maisu:
        min_maisu = maisu
print("合計枚数", min_maisu)
```

条件分岐（選択構造）

～場合分けを行う方法～

■条件分岐（選択構造）

■比較演算の結果、真（True）の場合に実行したい処理、偽（False）の場合に実行したい処理を記述することができる

■記述方法

- インデント□を忘れないように注意
- □は半角空白（またはタブ：Tab）

```
if 比較演算:
□□比較演算がTrueのときの処理①
□□比較演算がTrueのときの処理②
else:
□□比較演算がFalseのときの処理①
□□比較演算がFalseのときの処理②
```

■比較演算

■2つの値の大小関係や等値関係を判定します

■結果は、真理値の真（True）または偽（False）で判定されます

AはB～	表記方法
等しい	A == B
等しくない	A != B
より大きい	A > B
より小さい	A < B
以上	A >= B
以下	A <= B

２時間目：リフレクション

2022

出典：株式会社ユーザーローカル
AIテキストマイニング
https://textmining.userlocal.jp/

まとめ

世田谷学園 中学校 高等学校
SETAGAYA GAKUEN SCHOOL

～構文を整理（まとめ）せずに演習や自由作成はハードルが高かった～

- **PyTryの日本語エラーは初学者に向いていた**
  - エラーメッセージをよく読んで確認してで済んだ
  - 典型的なエラーパターンの提示を事前にすれば、より効果が高いと考えられる
  - Google ColaboratoryよりもPyTryを好んで使用している生徒が多かった
- **ステップを踏んで作成するのは良かった**
  - 動画解説を聞きながら自分のペースで作業できた
    - 解説動画なしで一斉型だと半分程度しか消化できない
  - 動画を見ずに「入力⇒実行」して満足する生徒がいた
- **演習や自由作成はハードルが高かった**
  - 3時間目にまとめを入れたほうが効果的かもしれない
    - 変数・繰り返し・条件分岐の確認を徹底することが大切
  - このタイミングで③～⑥はやらなくても良いかも

①あたり1%ガチャ（100連ガチャを200回）

②効率の良い硬貨の使い方（試作問題：第3問）

⑦まとめ

③おみくじ作成（演習）

④じゃんけん（基本形の作成）

⑤⑥基本形を改造してじゃんけんを作成（自由作成）

３学期
典型的なアルゴリズムに関するプログラミング

次のプログラムを作成

～復習内容を使って様々なプログラムを作成する～

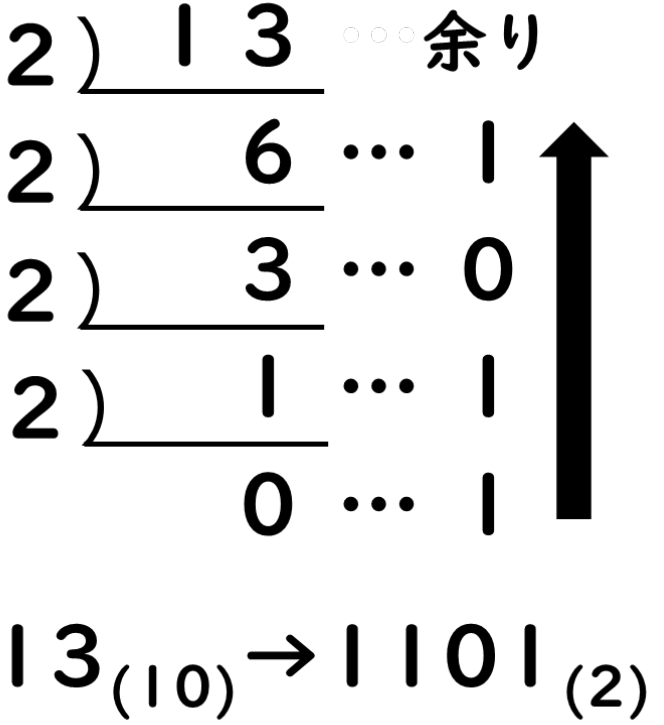
- **2進法に変換するプログラム**
  - 2学期期末試験で出題した内容を改めて確認する
  - 試験ではfor文を使用したが、今回はwhile文を使用する
- **並べ替えのプログラム**
  - プログラミングプリントNo4の内容を確認する
  - 2重ループ（入れ子構造の繰り返し）の理解を深めることが大切
- **最大公約数を求めるプログラム**
  - 単純な繰り返しで求めるプログラムを作成する。
  - その後、ユークリッドの互除法を使って作成し、効果の違いを確認する。
- **リスト（配列）の中にある数値を検索するプログラム**
  - 単純にリストの先頭から一致する数値を探して一致したら終了する
  - 二分探索を用いて効率よく一致する数値を検索する

2進法に変換するプログラム

～2学期期末試験で出題した内容をあらためて確認します～

■2進法の変換方法

- 10進法の13を2進法に変換するためには、右図のように、まず13を2で割って商の6と余り1を求める。先ほど求めた商の6を2で割って商の3と余り0を求め、さらに商の3を2で割って商の1と余り1を求め、商の1を2で割って商の0と余り1を求める。商が0になったら終了し、余りを逆順に並べた1101が求める答えとなる。



■プログラムの作成方法として

- ■元の数が0になるまで与えられた値を2で割り続ける
  - ■商と余りを表示し続ける

2進法へ変換するプログラムを作成

～順を追って作成する～

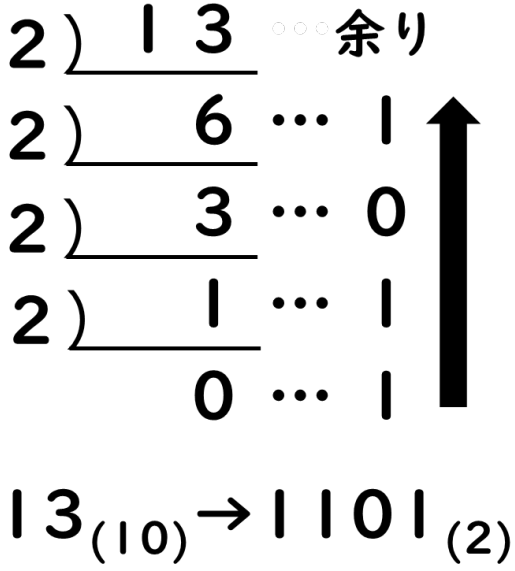
- **第1段階**
  - 与えられた値を0になるまで2で割り続け、商と余りを表示する
- **第2段階**
  - 第1段階でも十分だが、13だったら1101と表示したいので、表示できるようにプログラムを修正する
- **第3段階**
  - 数値をいろいろ変更してプログラムを実行する
  - 16進法へ変換するにはどうすればよいか考えてみる

出力

```
1    13 6 1
2    6 3 0
3    3 1 1
4    1 0 1
5
```

出力

```
1    1101
2
```



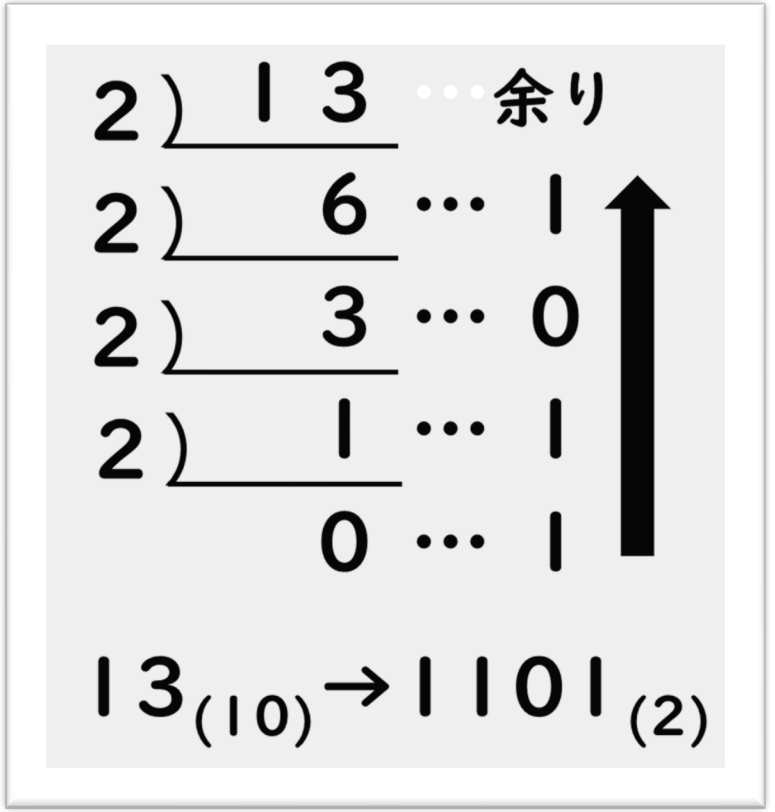
$13_{(10)} \rightarrow 1101_{(2)}$

２進法へ変換するプログラム

～第１段階：２で割り続けた商と余りを表示する～


```
1    N = 13
2    while N > 0:
3        print(N, N // 2, N % 2)
4        N = N // 2
5
```

2行目：
Nが0より大きい間は3～4行目の処理を実行する
3行目：
Nの値、Nを2で割った商、Nを2で割った余りを表示する
4行目：
Nを2で割った商をNに代入して次の桁の計算処理をする



2進法へ変換するプログラム

～第2段階：2進法を正しく表示できるように赤字部分を修正する～


```
1    N = 13
2    BIT = ""
3    while N > 0:
4        BIT = str(N % 2) + BIT
5        N = N // 2
6    print(BIT)
7
```

出力

```
1    1101
2
```

2行目：2進法への変換結果を保存する変数BITを用意する。
3行目：Nが0より大きい間は3～4行目の処理を実行する。
4行目：Nを2で割った余りを文字列に変換する（str関数を使用して変換）。文字列に変換した余りと変数BITの値を結合した結果を変数BITに代入しなおす。
5行目：Nを2で割った商をNに代入して次の桁の計算処理をする。
6行目：変換結果を表示する。

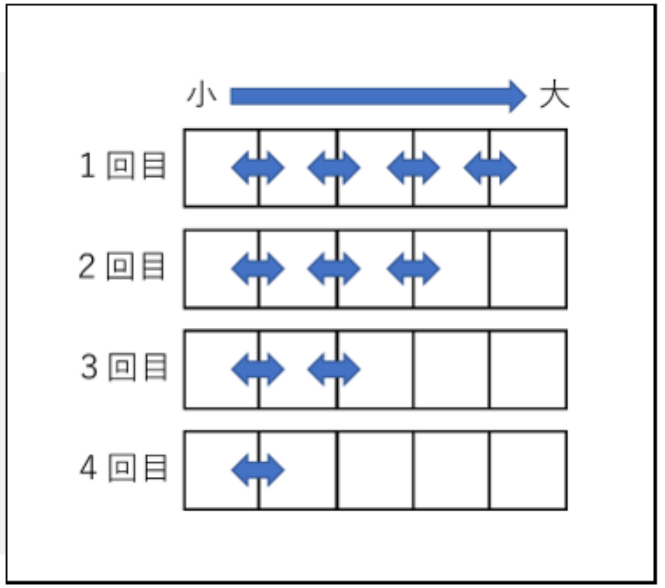
並べ替えのプログラム

- **第1段階**
  - 2重ループを作成して、2つの変数の動きを確認する
- **第2段階**
  - 繰り返しの開始値と終了値を調整して意図する動きに調整する
- **第3段階**
  - 値の交換を実装して完成させる

▶ 例題「並べ替え」

与えられたリストの要素を昇順に並べ替えるプログラムを作成する。

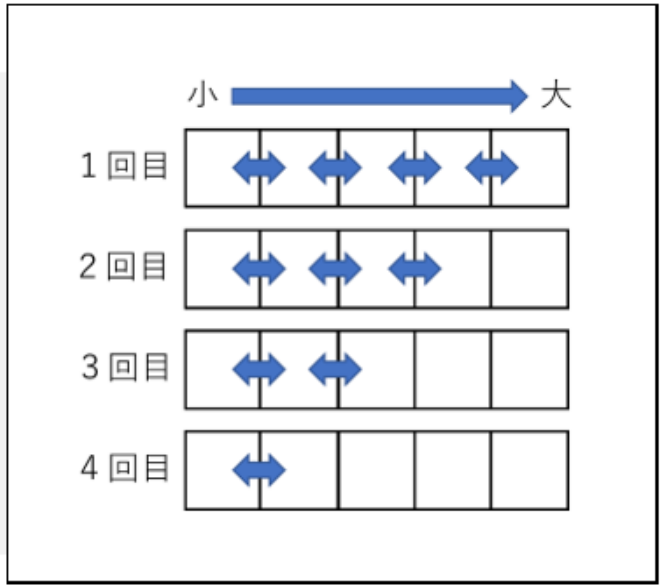
右図のように、リストの左から隣り合う要素の大小関係を比較し、左の要素が大きかったら右の要素と交換する作業を繰り返して並べ替え操作を行う。これを実現するプログラムを作成しなさい。



並べ替えのプログラム

～第1段階：2重ループの動きを確認する～

```
1    data = [5, 4, 3, 2, 1]
2    for i in range(5):
3        for j in range(5):
4            print(i, j)
5
```



出力

```
0 0
0 1
0 2
0 3
0 4
1 0
1 1
1 2
1 3
1 4
2 0
2 0
2 1
2 2
2 3
2 4
3 0
3 1
3 2
3 3
3 4
4 0
4 0
4 1
4 2
4 3
4 4
```

並べ替えのプログラム

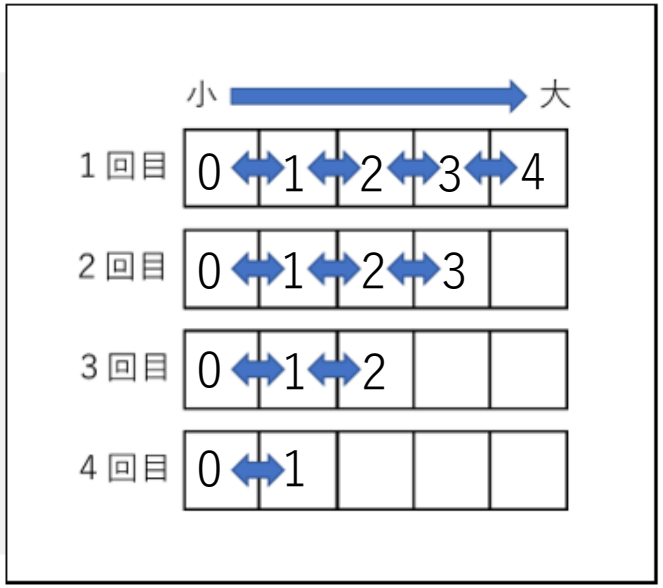
～第2段階：繰り返し回数の調整を行う～

```
1    data = [5, 4, 3, 2, 1]
2    for i in range(5):
3        for j in range(5 - i):
4            print(i, j)
5
```

```
1    data = [5, 4, 3, 2, 1]
2    for i in range(5):
3        for j in range(4 - i):
4            print(i, j)
5
```

出力

```
1    0 0    6    1 0    11    2 1
2    0 1    7    1 1    12    2 2
3    0 2    8    1 2    13    3 0
4    0 3    9    1 3    14    3 1
5    0 4    10   2 0    15    4 0
                        16
```



出力

```
1    0 0    6    1 1
2    0 1    7    1 2
3    0 2    8    2 0
4    0 3    9    2 1
5    1 0    10   3 0
                11
```

並べ替えのアルゴリズム

～第3段階：値の交換を実装して完成させる～


```
1    data = [5, 4, 3, 2, 1]
2    for i in range(5):
3        for j in range(4 - i):
4            if data[j] > data[j + 1]:
5                tmp = data[j + 1]
6                data[j + 1] = data[j]
7                data[j] = tmp
8        print(data)
9
```

4行目：配列の隣同士(data[j]とdata[j+1])を比較し、右(data[j+1])のデータが小さいか判断する。

5～7行目：右の数値が小さい時は、隣同士の数値を交換する。交換方法は以下の図を参照すること。

aに相当： data[j+1]

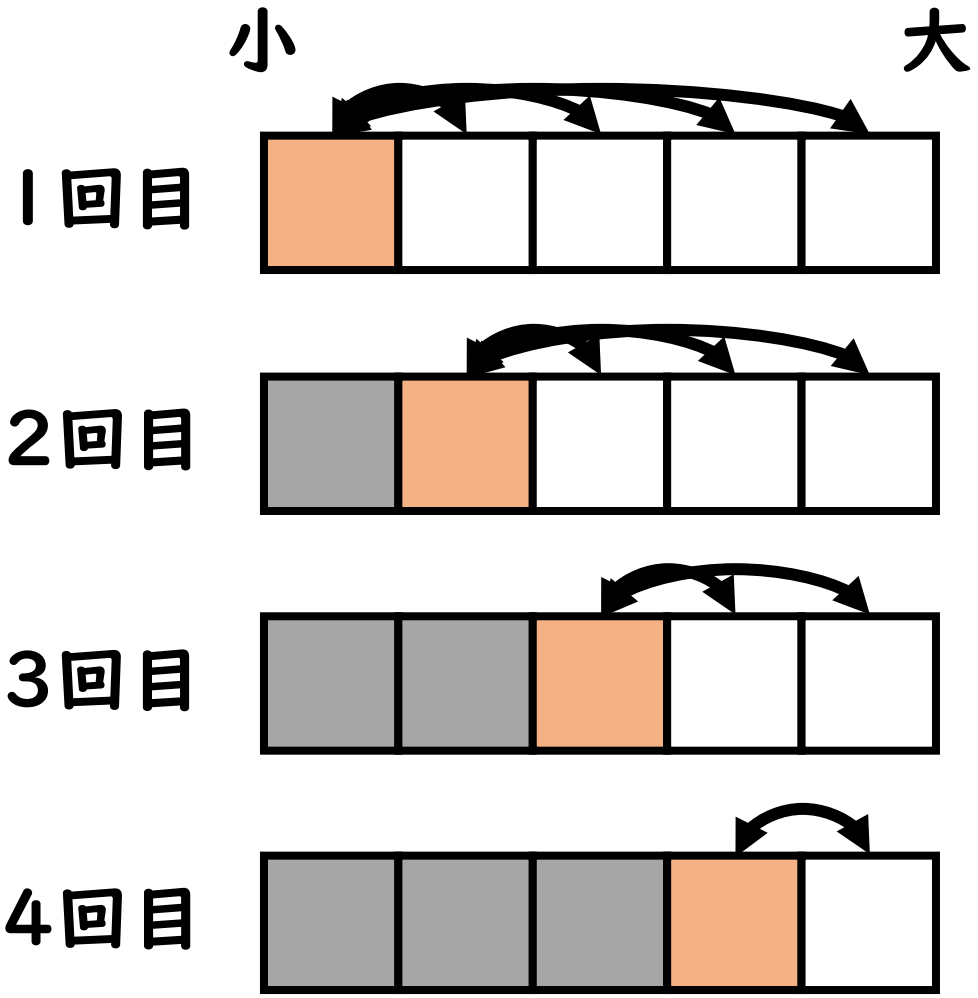
bに相当： data[j]

※2行目のrange(5)は、range(4)でも問題ありません。理由を考えてみてください。

並べ替えのアルゴリズム

～発展：以下の交換ルールだったらどうしますか？～

■先頭から順番に小さい値を見つけて、見つかったら交換する



【ヒント】
rangeの記述方法を工夫することで繰り返しの方法を制御することが可能になる。

```
# iの値は2,3,4と変化する
for i in range(2,5):
    print(i)

# iの値は0,2,4と2ずつ変化する
for i in range(0,5,2):
    print(i)
```

並べ替えのアルゴリズム

～発展：答えは何種類もありますが、一つの例を示します～

```
1    data = [5, 4, 3, 2, 1]
2    for i in range(5):
3        for j in range(i + 1, 5):
4            if data[i] > data[j]:
5                tmp = data[j]
6                data[j] = data[i]
7                data[i] = tmp
8        print(data)
9
```

最大公約数を求めるプログラム

～ユークリッドの互除法を使うと効率よく求められます～

- **作業１**
  - 割る数を確認するためのプログラムを作成する
  - 0で割り算、最大の数まで割ることができないなど、繰り返しの数に注意が必要
- **作業２**
  - 作業１で生成した数値を使って2つの数を割り算し、最大公約数を求める
- **作業３**
  - ユークリッドの互除法を用いて最大公約数を求める

最大公約数を求めるプログラム

～作業１：割る数を正しく設定できているか確認する～

```
1    a = 9
2    b = 21
3    for i in range(a):
4        print(i)
5
```

出力

```
1    0
2    1
3    2
4    3
5    4
6    5
7    6
8    7
9    8
10
```

・繰り返しが0からスタートすると、0で割り算をすることになってしまい不都合が発生する。
・繰り返しの最後が8となっており、少なくとも9まで割り算しないと正しく最大公約数を求めることができない。

```
1    a = 9
2    b = 21
3    for i in range(1, a + 1):
4        print(i)
5
```

出力

```
1    1
2    2
3    3
4    4
5    5
6    6
7    7
8    8
9    9
10
```

・rangeの開始値を1にすることで、0からスタートしないように設定できる。
・rangeの終了値をa+1にすることで、上記例で9まで割り算することができるように設定できる。

最大公約数を求めるプログラム

～作業２：2つの数を割り算して最大公約数を求める～


```
1    a = 9
2    b = 21
3    GCD = 1
4    for i in range(1, a + 1):
5        if a % i == 0 and b % i == 0:
6            GCD = i
7    print(GCD)
8
```

3行目：最大公約数を保存する変数GCDを作成する。初期値を1とする。

5行目：2つの数字（変数aと変数b）を変数iで割り算し、両方とも余りが0なら公約数となる。

6行目：5行目の条件がTrueのとき、最大公約数を保存する変数GCDに変数iの値を代入する。

※さらに大きな数値で割れることがわかったら、GCDの値は更新されるため、最終的に最大公約数が保存されることになる

変数aと変数bに非常に大きな数値を入れて実行してみてください。このプログラムでは非常に大きな数同士の最大公約数を求めることができないことを確認できます。

最大公約数を求めるプログラム

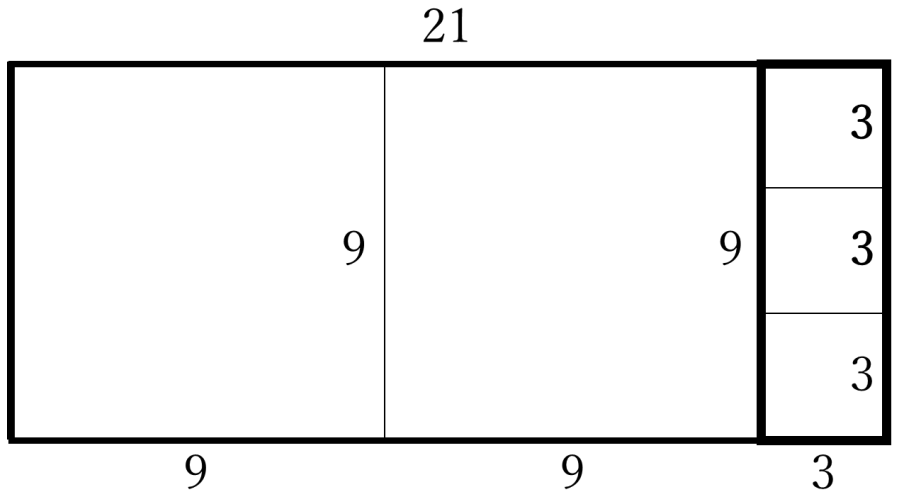
～作業３：ユークリッドの互除法を用いて最大公約数を求める～

■ユークリッドの互除法

21×9の長方形に正方形のタイルを敷き詰める方法を考えることで、最大公約数を求める。

21×9の長方形は、9×9の正方形を２つ（21÷9＝2あまり3）敷き詰めることができ、余った長方形は9×3となる。この9×3の長方形は、3×3の正方形を3つ（9÷3=3余り0）敷き詰めることができる。このとき、長方形が余ることはない（すべて同じ大きさの正方形になる）ので、3が最大公約数となる。

このように、長方形の長辺a÷短辺bの余りrの関係を利用して、長辺と短辺を短辺と余りに入れ替えながら余りが0になるまで繰り返したときの短辺bが最大公約数となる。



長辺 a	短辺 b	余り r
21	9	3
9	3	0

最大公約数を求めるプログラム

～作業３：ユークリッドの互除法を用いて最大公約数を求める～


```
1    a = 21
2    b = 9
3    while a % b != 0:
4        r = a % b
5        a = b
6        b = r
7    print(b)
8
```

プログラムの中では変数aと変数bの値を更新しながら処理を行っている

3行目：変数aの値を変数bで割り算し、割り切れなければ変数bは最大公約数ではないので4行目～6行目の処理を実行する。割り切れるまで繰り返す。

4行目：変数aを変数bで割った余りを変数rに代入する。

5行目：変数bの値を変数aに代入する。

6行目：変数rの値を変数bに代入する。

※左下の表で考えると、4行目は「余りrの3を求める処理」、5行目は赤矢印の代入処理、6行目は青矢印の代入処理に該当します。

長辺 a	短辺 b	余り r
21	9	3
9	3	0

変数aと変数bに非常に大きな数値を入れて実行してみてください。ユークリッドの互除法の効果がわかります。

数値を検索するプログラム

- 第1段階
  - 配列の先頭から順番に数値を検索する
- 第2段階
  - 二分探索を用いて一致する数値を検索する

数値を検索するプログラム

～作業１：配列の先頭から順番に数値を検索するプログラムを作成～


```
1   a = [3, 5, 11, 15, 24, 29, 35, 38, 39, 44, 49]
2   ans = 44
3   isYes = False
4   for i in range(len(a)):
5       if a[i] == ans:
6           isYes = True
7           break
8   if isYes == True:
9       print("Yes")
10  else:
11      print("No")
12
```

リストaの中に、変数ansの値があるかどうか検索する。

3行目：値が見つかったかどうかを管理する変数isYesを作成し、Falseを設定する。（見つかっていない状態とする。）

4行目：リストの先頭から末尾まで検索するための繰り返し処理を作成する。

5行目～7行目（一致する値の検索処理）：リストのi番目の要素（a[i]）と変数ansの値が一致していたら変数isYesの値をTrueに設定してbreakを実行し検索処理（繰り返し処理）を終了する。

8行目～11行目（検索結果の表示処理）：変数isYesの値を確認し、Trueなら値が見つかっているのでYesを出力し、Falseなら値が見つかっていないのでNoを出力する。

※この検索処理は最短で1回目、最大で11回目に値が見つかる。

数値を検索するプログラム

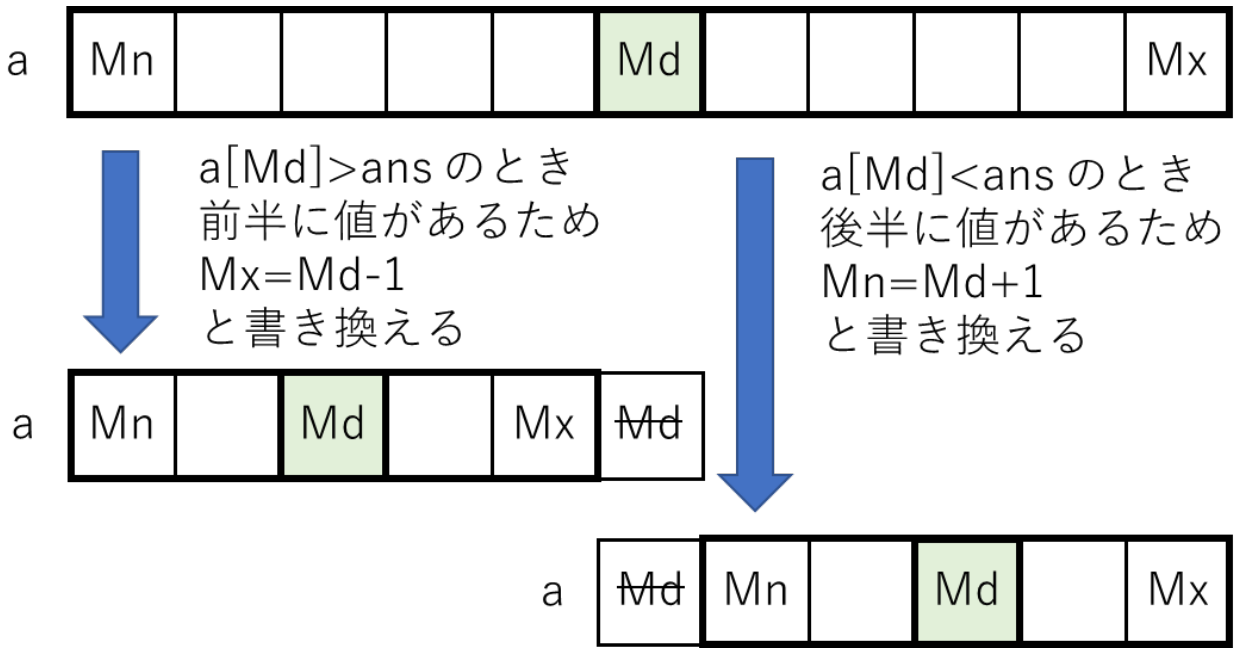
～作業２：二分探索を用いて一致する数値を検索する～

■二分探索

二分探索は右図のように、検索範囲を半分に絞りながら該当する値が見つかるまで繰り返し処理を行う。

たとえば、検索対象となるリストの要素数が100万件であっても、つねに20回程度の検索処理で終了する。

作業１で示した線形探索の場合は最大100万回の検索処理が行われることから、二分探索の効率のよさがわかる。



数値を検索するプログラム

～作業２：二分探索を用いて一致する数値を検索する～


```
1   a = [3, 5, 11, 15, 24, 29, 35, 38, 39, 44, 49]
2   ans = 44
3   isYes = False
4   Mn = 0
5   Mx = len(a) - 1
6   while Mn <= Mx:
7       Md = (Mn + Mx) // 2
8       if a[Md] == ans:
9           isYes = True
10          break
11      elif a[Md] > ans:
12          Mx = Md - 1
13      else:
14          Mn = Md + 1
15  if isYes == True:
16      print("Yes")
17  else:
18      print("No")
19
```

リストaの中に、変数ansの値があるかどうか検索する。

3行目：値が見つかったかどうかを管理する変数isYesを作成し、Falseを設定する。

4行目～5行目：検索対象となるリストの範囲の最小値Mnと最大値Mxを管理する変数を定義し、初期値を代入する。

6行目：最小値Mnと最大値Mxの大小関係が正しい間繰り返し処理を実行する。

7行目：最小値Mnと最大値Mxの中央の値（平均値）を計算し、変数Mdに代入する。

8行目～10行目：中央の値（a[Md]）と変数ansの値が一致していたら、変数isYesの値をTrueに設定してbreakを実行し、検索処理（繰り返し処理）を終了する。

11行目～12行目：中央の値（a[Md]）よりも変数ansの値が小さかったら、最大値Mxの値を、Md-1に置き換える。

13行目～14行目：中央の値（a[Md]）よりも変数ansの値が大きかったら、最小値Mnの値を、Md+1に置き換える。このように、検索対象の範囲を絞りながら最終的に検索値が見つかるまで繰り返す（検索対象の値が見つからない場合は、Mn>Mxとなるため繰り返し処理が終了する）。

15行目～18行目：最後に変数isYesの値を確認し、Trueなら値が見つかっているのでYesを出力し、Falseなら値が見つかっていないのでNoを出力する。

1時間目：リフレクション

2022

出典：株式会社ユーザーローカル
AIテキストマイニング
https://textmining.userlocal.jp/

逆ポーランド記法（後置記法）

～コンピュータで計算処理するときに都合が良い表現方法です～

- **普段目にしている数学の計算式は中置記法**
  - 演算子（計算記号）を数値の間に表記する方法
- **逆ポーランド記法（後置記法）は演算子を後ろに記載する方法**
  - 左から順に読み込んでいくと計算できるため、コンピュータで計算処理するときに都合が良い表記法

中置記法	逆ポーランド記法
3 ＋ 4	3 4 ＋
（ 1 ＋ 2 ） × 3	1 2 ＋ 3 ×
3 ×（ 2 ＋ 1 ）	3 2 1 ＋ ×
（3＋4）×（1－2）	3 4 ＋ 1 2 － ×

- **変換手順**　※計算順序に従って、順を追って逆ポーランド記法にすることで表記を変換できる

（1＋2）×3　⇒　1 2 ＋ ×3　⇒　1 2 ＋ 3 ×

3 ×（2＋1）　⇒　3 × 2 1 ＋　⇒　3 2 1 ＋ ×

問題演習

～中置記法⇔後置記法の変換と、後置記法の計算ができるように！！～

①中置記法：８÷（１＋３）×４ を後置記法であらわすと？

②中置記法：（ａ－ｂ）＋（ｃ＋ｄ）×ｅ を後置記法であらわすと？

③後置記法：６ ２ １ ＋ ÷ ５ ３ － × の計算結果は？

④後置記法：ａ ｂ × ｃ － ｄ ｅ ＋ ÷ ｆ ＋ を中置記法であらわすと？

⑤次の３条件を満たす式を１つ、後置記法で解答欄に書きなさい

- 計算結果が９である。
- 数値として使える数は、１ ２ ３ ４ ５ ６ ７に限る。ただし、同じ数を２回以上用いてはならない。また、２桁以上の数として用いてはならない。
- 演算子（＋ － × ÷）の中から３種類以上を用いること。同じ演算子を２回以上用いても良い。

■以上の問題をGoogleフォームで回答してください。

問題演習（解答）

～あわてず丁寧に順を追って考えてください～

計算順に従って後置記法に置き換える

①８÷（１＋３）×４

８÷　１３＋　×４

８１３＋÷　×４

８１３＋÷４×

②（ａ－ｂ）＋（ｃ＋ｄ）×ｅ

ａｂ－　＋　ｃｄ＋　×ｅ

ａｂ－　＋　ｃｄ＋ｅ×

ａｂ－ｃｄ＋ｅ×＋

前から順に処理をする

③６２１＋÷５３－×

６３÷５３－×

２５３－×

２２×

４

問題演習（解答）

～中置記法に戻したものは（）でくくってみると判断しやすい～

④ａ ｂ × ｃ － ｄ ｅ ＋ ÷ ｆ ＋

（ａ×ｂ） ｃ － ｄ ｅ ＋ ÷ ｆ ＋

（（ａ×ｂ）－ｃ） ｄ ｅ ＋ ÷ ｆ ＋

（ａ×ｂ－ｃ） （ｄ＋ｅ） ÷ ｆ ＋

（（ａ×ｂ－ｃ）÷（ｄ＋ｅ）） ｆ ＋

（（ａ×ｂ－ｃ）÷（ｄ＋ｅ））＋ｆ

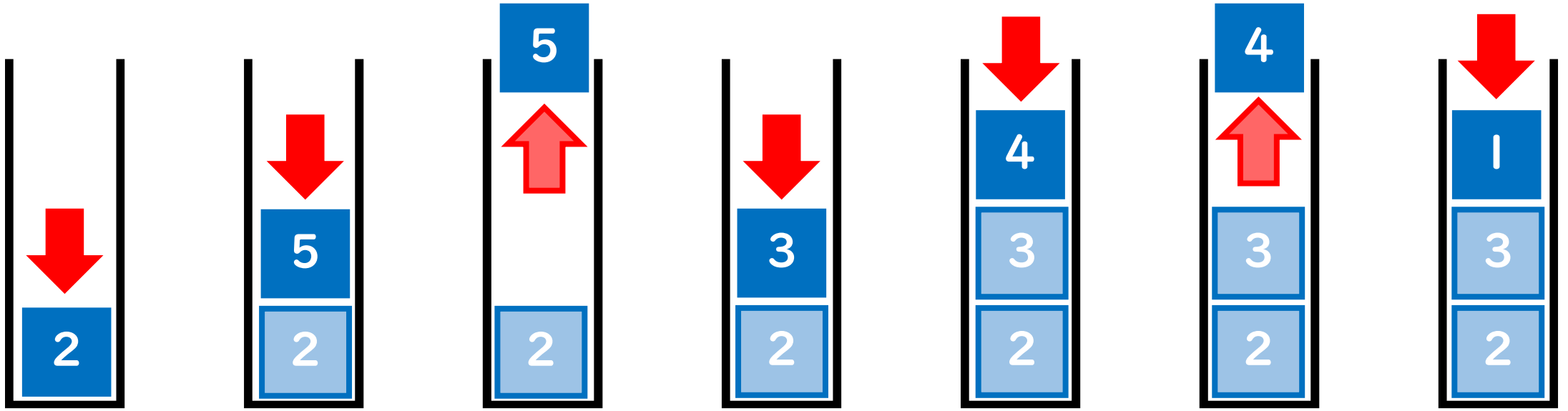
（ａ×ｂ－ｃ）÷（ｄ＋ｅ）＋ｆ

スタックについて

～後に入れたものを先に出す（Last-In First-Out）仕組みです～

- **スタックとは**
  - データを格納する入れ物の一種
  - 入ってきたデータを順番に格納して**保存する（push）**
  - 取り出すときは、最後に格納したデータから順に**取り出す（pop）**
- **たとえばスタックが空の状態から以下の処理を行うと…**

push(2) ⇒ push(5) ⇒ pop ⇒ push(3) ⇒ push(4) ⇒ pop ⇒push(1)



演習問題２

■push(3)⇒pop⇒push(5) ⇒ push(１) ⇒ pop ⇒ push(6) ⇒ push(7) ⇒ pop ⇒ pop ⇒ push(2) ⇒ push(4)という手順で操作を行うと、最終的にスタックの中のデータはどのようになっているか次の中から一つ選びなさい。

■ある手順で操作を行うと、最終的にスタックが次の状態になった。

３６４

※左側の数字が最初に入れたデータで、右に行くほど後から入れたデータになっているものとする。このようになる手順として適切なものを２つえらびなさい。

■以上の問題をGoogleフォームで回答してください。

後置記法の計算処理を行う

～スタックを使って計算処理を行う～

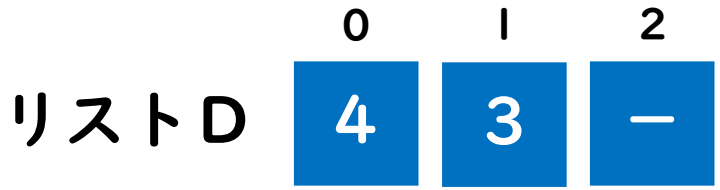

```python
# スタックに相当するdeque型を読み込む
from collections import deque

# 空白で区切られた後置記法のデータを読み込む
# 入力欄のデータをリスト(配列)に変換する
D = list(map(str, input().split()))
# スタックを定義する
ANS = deque()
# 後置記法のデータを順番に読み取るため、
# 後置記法のデータの長さ分だけ繰り返す
for i in range(len(D)):
    # 計算記号が来たらスタックからデータを２つ
    # 取り出して、計算した結果をスタックに戻す
    if D[i] == "+":
        n2 = ANS.pop()
        n1 = ANS.pop()
        ANS.append(n1 + n2)
    elif D[i] == "-":
        n2 = ANS.pop()
        n1 = ANS.pop()
        ANS.append(n1 - n2)
    elif D[i] == "*":
        n2 = ANS.pop()
        n1 = ANS.pop()
        ANS.append(n1 * n2)
    elif D[i] == "/":
        n2 = ANS.pop()
        n1 = ANS.pop()
        ANS.append(n1 / n2)
    else:
        # 計算記号以外のとき(数字のとき)は
        # スタックに追加する
        ANS.append(int(D[i]))
print(ANS[0])

```

入力 ペースト

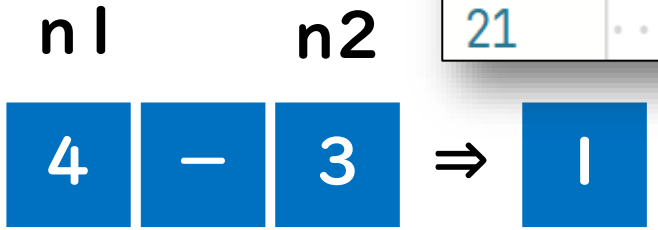
```
1  4 3 -
```

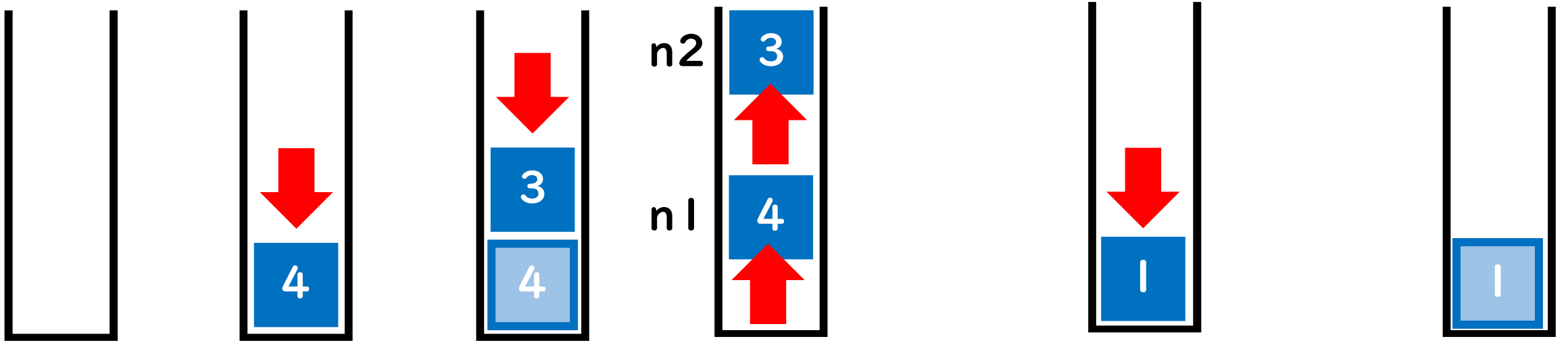
後置記法のプログラム

～図解によるイメージ～




```
19        n2 = ANS.pop()        89
20        n1 = ANS.pop()
21        ANS.append(n1 - n2)
```





スタック
ANS

1文字目
数字は保存

2文字目
数字は保存

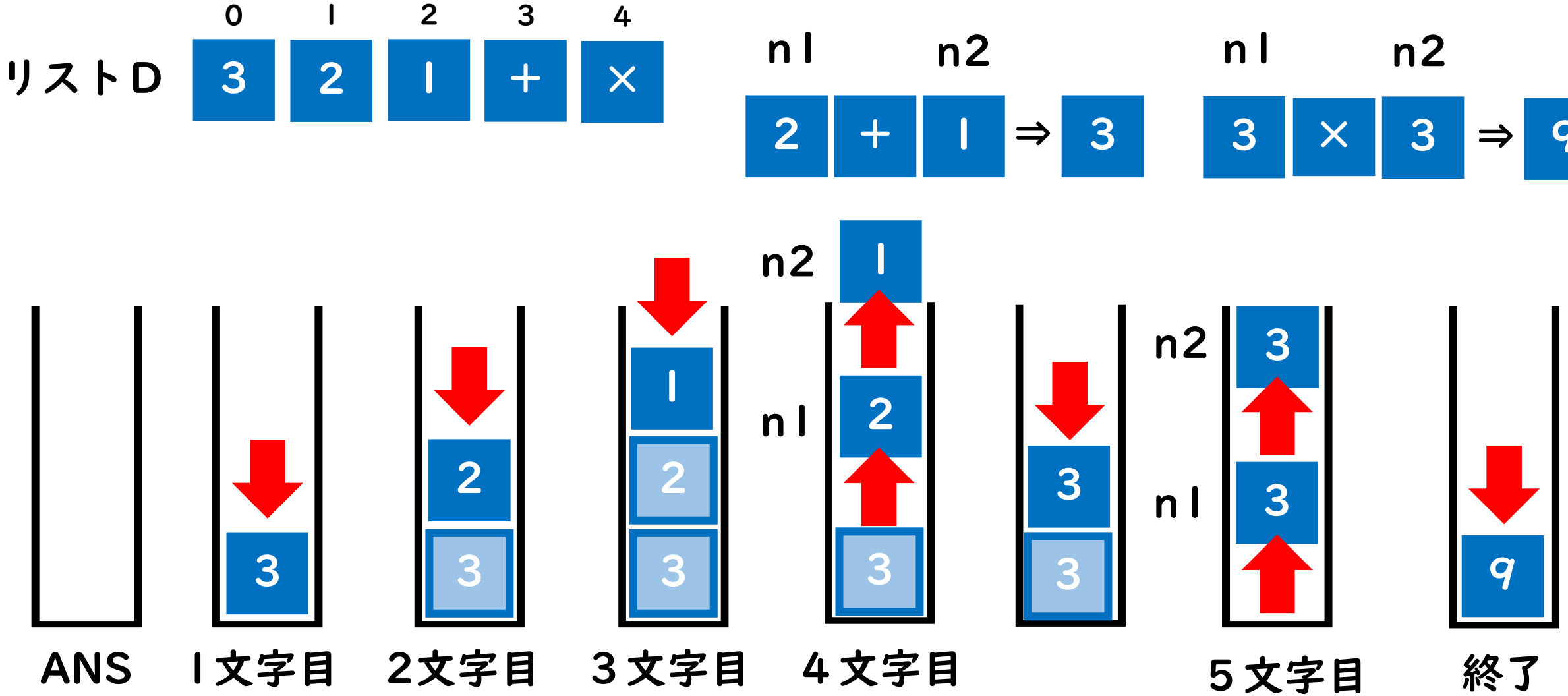
3文字目
演算子は２つ
数字を取り出す

計算が終わったら
スタックに数字を保存

終了

後置記法のプログラム

～図解によるイメージ～



発展：後置記法を中置記法に変換

～仕組みを把握すれば、ちょっとした工夫で作成できます～

■足し算・引き算の計算を行うとき

- ■（）をつけて計算順が正しく反映されるように式を生成する
- ■下の例で（）をつけないと式がどのようになるか考えてみること！

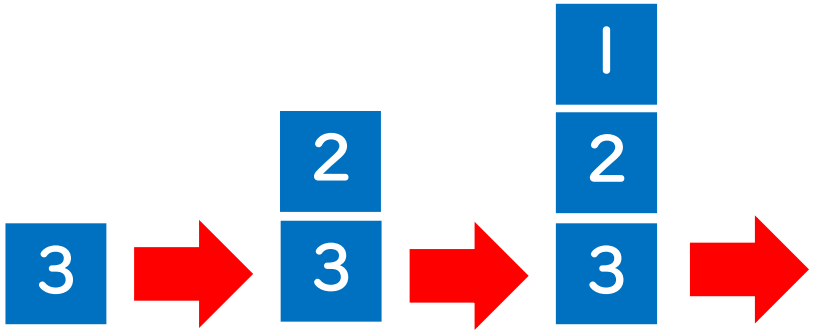
■掛け算・割り算の計算を行うとき

- ■四則演算のときに先に計算するのでそのまま問題ない

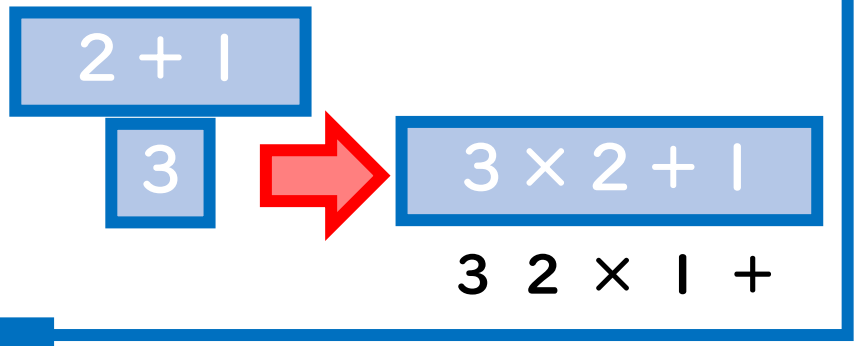
■文字の結合（文字同士であることが大前提）

- ■＋の記号を使って文字を結合することができる

リストD　3　2　1　＋　×



（）がないと、おかしなことに



発展：後置記法を中置記法に変換

～仕組みを把握すれば、ちょっとした工夫で作成できます～


```python
# スタックに相当するdeque型を読み込む
from collections import deque

# 空白で区切られた後置記法のデータを読み込む
# 入力欄のデータをリスト(配列)に変換する
D = list(map(str, input().split()))
# スタックを定義する
ANS = deque()
# 後置記法のデータを順番に読み取るため、
# 後置記法のデータの長さ分だけ繰り返す
for i in range(len(D)):
    # 計算記号が来たらスタックからデータを２つ
    # 取り出して、計算した結果をスタックに戻す
    if D[i] == "+" or D[i] == "-":
        n2 = ANS.pop()
        n1 = ANS.pop()
        ANS.append("(" + n1 + D[i] + n2 + ")")
    elif D[i] == "*" or D[i] == "/":
        n2 = ANS.pop()
        n1 = ANS.pop()
        ANS.append(n1 + D[i] + n2)
    else:
        # 計算記号以外のとき(数字のとき)は
        # スタックに追加する
        ANS.append(D[i])
print(ANS[0])
```

入力　ペースト

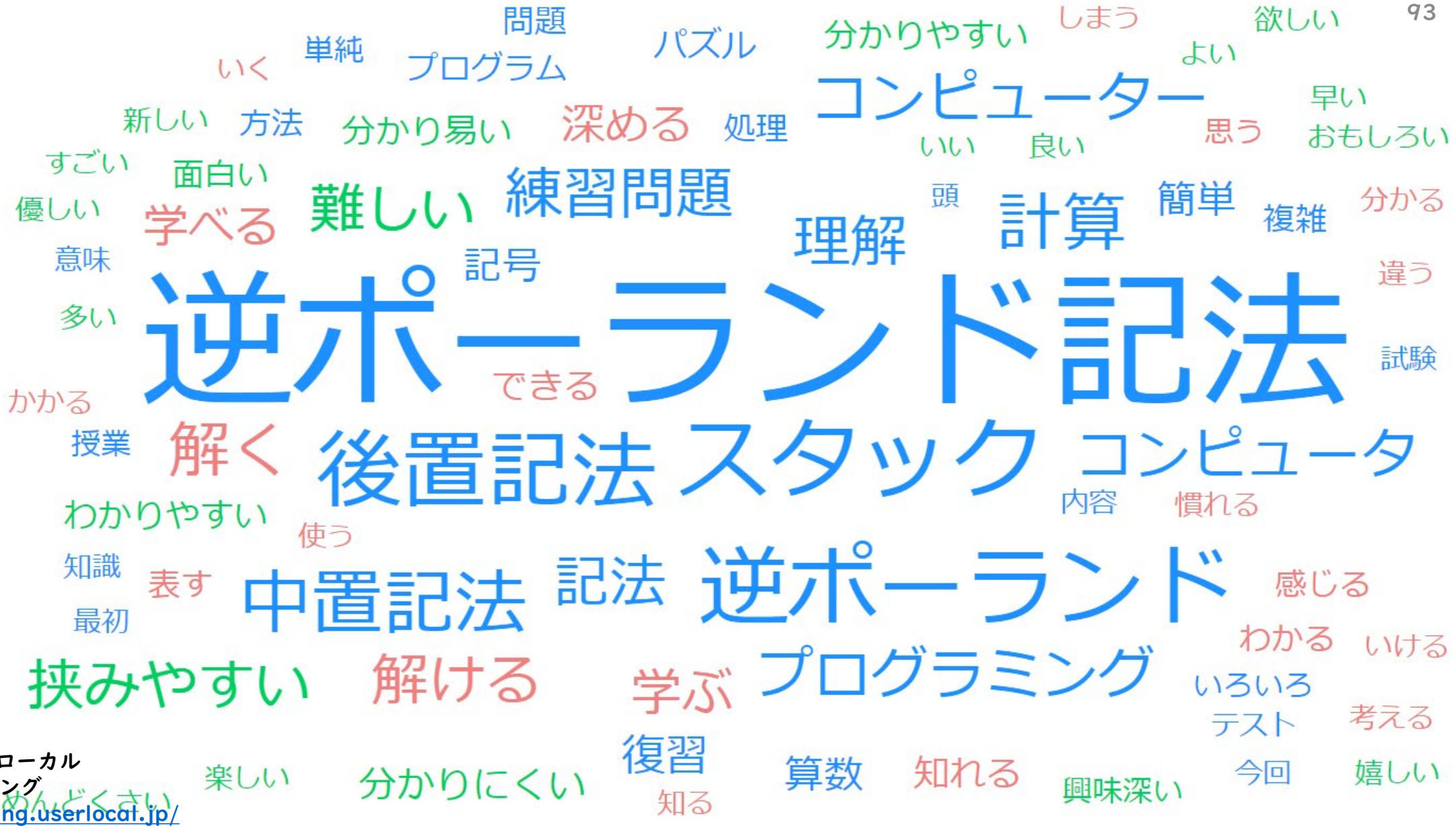
```
1
8 1 3 + *
```

出力

```
1
8*(1+3)
```

２時間目：リフレクション

2022



出典：株式会社ユーザーローカル
AIテキストマイニング
https://textmining.userlocal.jp/

まとめ

世田谷学園 中学校 高等学校
SETAGAYA GAKUEN SCHOOL

～身近な面白い題材を用いた授業を行い、生徒の記憶に残る授業～

- **詰め込んだ知識は、2年後（1年後）忘れている可能性大**
  - 忘れている前提で授業計画を立てる
  - 身近な面白い題材を使って授業し、知識につなげる方がお互いに幸せ
    - 稲垣先生（都立神代高校）『自分事』
  - 試験の場で「こんなことやったな」ってのを思い出せるような授業
- **生徒が定期的に復習するための工夫**
  - 1年生の定期テストを2年生（3年生）に配布する
    - 定期テストの過去問を1年生に配布
  - 大学入学共通テスト「情報Ⅰ」体験模試の受験を促す

- **情報収集として研究会への参加**
  - **8月上旬～中旬**　全国高等学校情報教育研究会全国大会（愛知大会）
  - **12月26日**　　神奈川県情報科実践事例報告会
  - キミのミライ発見（河合塾）　https://www.wakuwaku-catch.net/