

新型コロナウイルスへの対応

elementaryCS-
1st

Naoyuki
Nagatou

- 講義は第一，第二ともオンラインで行います
- 何らかの理由で講義に参加できなかったとしても，心配しないでください
 - ネットワークや機器の不調
 - 自身が感染，などなど
- 何某か対応するので焦らずゆっくり落ち着いて参加してください
- PC の記憶領域をかなり圧迫するので講義の録画は各自でする必要はありません
- こちらから動画視聴ための URL を別途お送りします.

コンピュータサイエンス入門第一

永藤 直行

東京工業大学 情報理工学院

3rd quarter

Part I

Prologue

Prologue

elementaryCS-
1st

Naoyuki
Nagatou

ガイダンス

科目の概要

評価基準

講義スケジュール

CS 講義概要

講義の目標

講義内容

- 1 ガイダンス
 - 科目の概要
 - 評価基準
 - 講義スケジュール

- 2 CS 講義概要
 - 講義の目標
 - 講義内容

Outline

elementaryCS-
1st

Naoyuki
Nagatou

ガイダンス

科目の概要

評価基準

講義スケジュール

CS 講義概要

講義の目標

講義内容

1

ガイダンス

- 科目の概要
- 評価基準
- 講義スケジュール

2

CS 講義概要

- 講義の目標
- 講義内容

科目の概要

elementaryCS-
1st

Naoyuki
Nagatou

ガイダンス
科目の概要
評価基準
講義スケジュール

CS 講義概要

講義の目標
講義内容

- 学期: 水曜日 3-4 限
- 場所: Zoom
- 担当教員: 永藤 直行 (ナガトウ ナオユキ)
- 連絡先: nagatou@presystems.xyz
- 質問時間: Zoom は立ち上げたままにしておくので講義終了後かメールで
- CS4b クラスのサイト: OCW-i か
<https://sites.google.com/a/presystems.xyz/sample/home/elementary-computer-science>
- 共通サイト: 教育用電子計算機システム のサイトから辿ってください
- その他ツールは必要に応じて URL を示します

評価基準

elementaryCS-
1st

Naoyuki
Nagatou

ガイダンス

科目の概要

評価基準

講義スケジュール

CS 講義概要

講義の目標

講義内容

- 講義は全 7 回, 期末試験は行いません
- 宿題: 3 回 $3 \times 5 = 15$ 点
- 課題: 4 回 $20 + 20 + 20 + 25 = 85$ 点
- 宿題・課題提出:
 - 講義時間中に課題を出します
 - 提出方法はその都度指定します
- 宿題と課題提出で出欠確認に変わります

講義日程

elementaryCS-
1st

Naoyuki
Nagatou

ガイダンス

科目の概要

評価基準

講義スケジュール

CS 講義概要

講義の目標

講義内容

回	題目	場所
1	ガイダンス, 環境構築, 計算の基礎	Zoom
2	プログラミング演習	Zoom
3	配列, 文字列	Zoom
4	プログラミング演習	Zoom
5	暗号入門	Zoom
6	プログラミング演習	Zoom
7	まとめ	Zoom

Outline

elementaryCS-
1st

Naoyuki
Nagatou

ガイダンス

科目の概要

評価基準

講義スケジュール

CS 講義概要

講義の目標

講義内容

1

ガイダンス

- 科目の概要
- 評価基準
- 講義スケジュール

2

CS 講義概要

- 講義の目標
- 講義内容

講義の目標

elementaryCS-
1st

Naoyuki
Nagatou

ガイダンス

科目の概要

評価基準

講義スケジュール

CS 講義概要

講義の目標

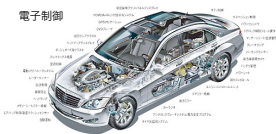
講義内容

- 本講義では、このコンピュータサイエンスの基本をなす考え方を、課題をやることを通して体得する
- 物理現象をシミュレートしたり
- 経済活動にともなう帳票類を管理したり
- 機器を制御したり
- コンピュータがいろいろな場面で利用されている



シミュレーション

電子制御



帳票管理



情報クラウド

INTERNET

コンピュータに載せるとは？

elementaryCS-
1st

Naoyuki
Nagatou

ガイダンス

科目の概要

評価基準

講義スケジュール

CS 講義概要

講義の目標

講義内容

- なぜコンピュータが利用できるのか
- どうやってコンピュータに載せるのか

コンピュータに載せる

やりたいことを計算をもちいて表現し、コンピュータに処理させること

目標

- CS 第一
 - 計算で表現するとは何か
 - コンピュータで処理するとは
- CS 第二 第二の講義概要
 - 計算の強力な道具 ⇒ 再帰
 - 載せ方の上手下手があること ⇒ アルゴリズムやデータ

講義内容

elementaryCS-
1st

Naoyuki
Nagatou

ガイダンス

科目の概要

評価基準

講義スケジュール

CS 講義概要

講義の目標

講義内容

- 以下の演習を通して実感しながら理解していく

演習内容

- 演習課題 1: 四則演算でアニメーション
 - 計算の基本要素を知る
- 演習課題 2: 循環小数
 - 配列とは
- 演習課題 3: 暗号解読に挑戦
 - プログラミングとは
 - 計算の組み立て方

Part II

計算の基本

計算の基本

elementaryCS-
1st

Naoyuki
Nagatou

はじめに

課題 1 の目標とテーマ
CS のこころ

データは数で
ある

Bit と Byte
自然数の n 進表記

計算 $= \pm 1$ と繰
り返し

計算とは
計算の基本要素

プログラムの
書き方

プログラミングのため
の仕掛け

宿題 1 を動かしてみる

開発環境構築

Python3 環境構築
動作確認

3

はじめに

- 課題 1 の目標とテーマ
- CS のこころ

4

データは数である

- Bit と Byte
- 自然数の n 進表記

5

計算 $= \pm 1$ と繰り返し

- 計算とは
- 計算の基本要素

6

プログラムの書き方

- プログラミングのための仕掛け
- 宿題 1 を動かしてみる

7

開発環境構築

- Python3 環境構築
- 動作確認

Outline

elementaryCS-
1st

Naoyuki
Nagatou

はじめに

課題 1 の目標とテーマ
CS のこころ

データは数で
ある

Bit と Byte
自然数の n 進表記

計算 $= \pm 1$ と繰
り返し

計算とは
計算の基本要素

プログラムの
書き方

プログラミングのため
の仕掛け

宿題 1 を動かしてみる

開発環境構築

Python3 環境構築
動作確認

3

はじめに

- 課題 1 の目標とテーマ
- CS のこころ

4

データは数である

- Bit と Byte
- 自然数の n 進表記

5

計算 $= \pm 1$ と繰り返し

- 計算とは
- 計算の基本要素

6

プログラムの書き方

- プログラミングのための仕掛け
- 宿題 1 を動かしてみる

7

開発環境構築

- Python3 環境構築
- 動作確認

CS のころ

elementaryCS-
1st

Naoyuki
Nagatou

はじめに

課題 1 の目標とテーマ
CS のころ

データは数で
ある

Bit と Byte
自然数の n 進表記

計算 $= \pm 1$ と繰
り返し

計算とは
計算の基本要素

プログラムの
書き方

プログラミングのため
の仕掛け

宿題 1 を動かしてみる

開発環境構築

Python3 環境構築
動作確認

- しつこいようですが、すべては計算
- コンピュータに載せるには
 - 対象をデータとして表すこと
 - 処理を基本演算の組み合わせで表すこと
- 処理とはコンピュータのなかの抽象的な世界に存在して
- データというもう一つの抽象的な存在を操作する
- この処理やデータをプログラミング言語の記号をもちいて注意深く構成したのがプログラム

Outline

elementaryCS-
1st

Naoyuki
Nagatou

はじめに

課題 1 の目標とテーマ
CS のこころ

データは数で
ある

Bit と Byte
自然数の n 進表記

計算 $= \pm 1$ と繰
り返し

計算とは
計算の基本要素

プログラムの
書き方

プログラミングのため
の仕掛け
宿題 1 を動かしてみる

開発環境構築

Python3 環境構築
動作確認

- 3 はじめに
 - 課題 1 の目標とテーマ
 - CS のこころ

- 4 データは数である
 - Bit と Byte
 - 自然数の n 進表記

- 5 計算 $= \pm 1$ と繰り返し
 - 計算とは
 - 計算の基本要素

- 6 プログラムの書き方
 - プログラミングのための仕掛け
 - 宿題 1 を動かしてみる

- 7 開発環境構築
 - Python3 環境構築
 - 動作確認

データは数である

elementaryCS-
1st

Naoyuki
Nagatou

はじめに

課題 1 の目標とテーマ
CS のこころ

データは数で
ある

Bit と Byte
自然数の n 進表記

計算 $= \pm 1$ と繰
り返し

計算とは
計算の基本要素

プログラムの
書き方

プログラミングのため
の仕掛け

宿題 1 を動かしてみる

開発環境構築

Python3 環境構築
動作確認

- データから見ていくことにして
- データはすべて 2 進列で表される
 - 自然数, 整数, 実数: 18, -3, 3.14 など
 - 文字: 文字コード: ASCII, Unicode など
 - 画像, 映像
 - 音
 - におい, 味, 触覚

情報とは

elementaryCS-
1st

Naoyuki
Nagatou

はじめに

課題 1 の目標とテーマ
CS のこころ

データは数で
ある

Bit と Byte
自然数の n 進表記

計算 $= \pm 1$ と繰
り返し

計算とは
計算の基本要素

プログラムの
書き方

プログラミングのため
の仕掛け

宿題 1 を動かしてみる

開発環境構築

Python3 環境構築
動作確認

- ここで情報とは何か考えてみます
- 情報とは"ある物事，事情についてのお知らせ"
- 情報の価値はどう決まるか？
 - 驚きをもって受け止められる情報は価値が高い？
 - 日常的な情報は価値が低い？

まずは情報量というもの

elementaryCS-
1st

Naoyuki
Nagatou

はじめに

課題 1 の目標とテーマ
CS のこころ

データは数で
ある

Bit と Byte
自然数の n 進表記

計算 $= \pm 1$ と繰
り返し

計算とは
計算の基本要素

プログラムの
書き方

プログラミングのため
の仕掛け

宿題 1 を動かしてみる

開発環境構築

Python3 環境構築
動作確認

- ① ある結果や情報を得る場合を考える
- ② 結果や情報を生じる事象が確率現象であると見なす
- ③ 確率 p の事象の情報量を $I(p)$ であらわす
- ④ $I(p)$ は単調減少関数
 - 頻繁に起こっていること (p が大きい) は情報量が少ない
 - 頻繁に起こらないこと (p が小さい) は情報量が多い
- ⑤ 連続関数である
 - 確率のわずかな変化で情報量が大きく変化するの是不自然

情報量の定義

ある事象 a の生起確率を p_a とするとその情報量 $I(p_a)$ は
 $\log_2 \frac{1}{p_a} = -\log_2 p_a$ であらわすことにする

ビットとは

elementaryCS-
1st

Naoyuki
Nagatou

はじめに

課題 1 の目標とテーマ
CS のこころ

データは数で
ある

Bit と Byte

自然数の n 進表記

計算 = ± 1 と繰
り返し

計算とは
計算の基本要素

プログラムの
書き方

プログラミングのため
の仕掛け

宿題 1 を動かしてみる

開発環境構築

Python3 環境構築
動作確認

1 bit とは

- ① 今, 2 つの事象を考える
- ② 同じ確率 $P = \frac{1}{2}$ で生起するとする
- ③ このとき, ひとつの事象 a の情報量 $I(a) = \log_2 \frac{1}{\frac{1}{2}} = 1$
- ④ これが 1 bit
- ⑤ 確率 $\frac{1}{2}$ で起こる事象を知った時の情報量が 1 bit(ビット)

- それでは確率 $\frac{1}{10}$ で起こる事象を知った時は 1 hartley(ハートレー)
- 確率 $\frac{1}{e}$ では 1 nat(ナット)

情報の記録

elementaryCS-
1st

Naoyuki
Nagatou

はじめに

課題 1 の目標とテーマ
CS のこころ

データは数で
ある

Bit と Byte
自然数の n 進表記

計算 $= \pm 1$ と繰
り返し

計算とは
計算の基本要素

プログラムの
書き方

プログラミングのため
の仕掛け

宿題 1 を動かしてみる

開発環境構築

Python3 環境構築
動作確認

- 情報はビットの列として記録
- 明確に区別された 2 つの状態で記録しています
 - 磁性体の向き, 電圧の高低, スイッチの開閉
- 計算機科学では 2 つの状態を便宜的に 0 と 1 として議論しています

ビットによる表現

elementaryCS-
1st

Naoyuki
Nagatou

はじめに

課題 1 の目標とテーマ
CS のこころ

データは数で
ある

Bit と Byte
自然数の n 進表記

計算 $= \pm 1$ と繰
り返し

計算とは
計算の基本要素

プログラムの
書き方

プログラミングのため
の仕掛け

宿題 1 を動かしてみる

開発環境構築

Python3 環境構築
動作確認

- 2 つの状態を取り得るデバイスを N 個並べてそれぞれ独立としたらどれだけの情報があらわせるか
- 答えは $\log_2 2^N = N$ となり N ビットの情報量となります
- N ビットでどれだけの事象を区別できるでしょうか
- 答えは 2 個の要素から N 個の重複順列 ${}_2\Pi_N = 2^N$ です

バイトとは

elementaryCS-
1st

Naoyuki
Nagatou

はじめに

課題 1 の目標とテーマ
CS のこころ

データは数で
ある

Bit と Byte
自然数の n 進表記

計算 = ± 1 と繰
り返し

計算とは
計算の基本要素

プログラムの
書き方

プログラミングのため
の仕掛け
宿題 1 を動かしてみる

開発環境構築

Python3 環境構築
動作確認

- 人間にとって意味をなす長さ N の小ブロック
- 現在のコンピュータでは 8 bit としています

数表記

elementaryCS-
1st

Naoyuki
Nagatou

はじめに

課題 1 の目標とテーマ
CS のこころ

データは数で
ある

Bit と Byte
自然数の n 進表記

計算 $= \pm 1$ と繰
り返し

計算とは
計算の基本要素

プログラムの
書き方

プログラミングのため
の仕掛け

宿題 1 を動かしてみる

開発環境構築

Python3 環境構築
動作確認

- ビットの列で表すことは先に述べました
- では自然数はどうあらわすでしょう
- 数表記は 10 進が唯一の方法ではありません
- n 進表記が可能です
- 数はどうコンピュータ内で表現されるかみていきます

n 進表記

elementaryCS-
1st

Naoyuki
Nagatou

はじめに

課題 1 の目標とテーマ
CS のこころ

データは数で
ある

Bit と Byte
自然数の n 進表記

計算 $= \pm 1$ と繰
り返し

計算とは
計算の基本要素

プログラムの
書き方

プログラミングのため
の仕掛け

宿題 1 を動かしてみる

開発環境構築

Python3 環境構築
動作確認

- 実は日常的に n 進法を利用しています
- 時間は 24 進法, 60 進法, 30 進法, 360 進法をもちいています
- たとえば 24 進法では 24 になったら位が一つ上がります
- コンピュータでは 2 進法をもちいて自然数を表します
- 2 進法ではやはり人間には分かりづらいので 8 進法や 16 進法であらわすことが多いです

n 進法の各桁

elementaryCS-
1st

Naoyuki
Nagatou

はじめに

課題 1 の目標とテーマ
CS のこころ

データは数で
ある

Bit と Byte
自然数の n 進表記

計算 = ± 1 と繰
り返し

計算とは
計算の基本要素

プログラムの
書き方

プログラミングのため
の仕掛け

宿題 1 を動かしてみる

開発環境構築

Python3 環境構築
動作確認

- 2 進法, 8 進法, 16 進法でも 10 進法と同じように位どりによってあらわします
- 良くご存知のように 10 進法では 1 桁を 0-9 のいずれかであらわしています
- 123 という自然数であれば $1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0$ といった具合です

2 進法の各桁

elementaryCS-
1st

Naoyuki
Nagatou

はじめに

課題 1 の目標とテーマ
CS のこころ

データは数で
ある

Bit と Byte
自然数の n 進表記

計算 $\equiv \pm 1$ と繰
り返し

計算とは
計算の基本要素

プログラムの
書き方

プログラミングのため
の仕掛け

宿題 1 を動かしてみる

開発環境構築

Python3 環境構築
動作確認

- 2 進法では各桁は 0 と 1 だけになります
- 10 進法の場合と同様に位どりします, ただし底が 2 になります
- $(010)_2$ という自然数であれば $0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$ と
いった具合です
- この例では $(2)_{10}$ は位が一つ上がって $(010)_2$ となっています

16 進法の各桁

elementaryCS-
1st

Naoyuki
Nagatou

はじめに

課題 1 の目標とテーマ
CS のこころ

データは数で
ある

Bit と Byte
自然数の n 進表記

計算 = ± 1 と繰
り返し

計算とは
計算の基本要素

プログラムの
書き方

プログラミングのため
の仕掛け

宿題 1 を動かしてみる

開発環境構築

Python3 環境構築
動作確認

- 2 進法では桁が多くなって見づらいので 16 進で表記します
- 16 進法では各桁:
 - 0,1,2,3,4,5,6,7,8,9 と 0-9 までは 10 進と同じ
 - 10,11,12,13,14,15 は A,B,C,D,E,F をもちいます
- 10 進法の場合と同様に位どりします, ただし底が 16 になります
- $(1F0)_{16}$ という自然数であれば $1 \times 16^2 + F \times 16^1 + 0 \times 16^0$ といった具合です

各数字の対応

elementaryCS-
1st

Naoyuki
Nagatou

はじめに

課題 1 の目標とテーマ
CS のこころ

データは数で
ある

Bit と Byte
自然数の n 進表記

計算 $\equiv \pm 1$ と繰
り返し

計算とは
計算の基本要素

プログラムの
書き方

プログラミングのため
の仕掛け

宿題 1 を動かしてみる

開発環境構築

Python3 環境構築
動作確認

10 進	8 進	16 進	2 進
0	0	0	0
1	1	1	1
2	2	2	10
3	3	3	11
4	4	4	100
5	5	5	101
6	6	6	110
7	7	7	111

10 進	8 進	16 進	2 進
8	10	8	1000
9	11	9	1001
10	12	A	1010
11	13	B	1011
12	14	C	1100
13	15	D	1101
14	16	E	1110
15	17	F	1111

n 進数の変換

elementaryCS-
1st

Naoyuki
Nagatou

はじめに

課題 1 の目標とテーマ
CS のこころ

データは数で
ある

Bit と Byte
自然数の n 進表記

計算 $= \pm 1$ と繰
り返し

計算とは
計算の基本要素

プログラムの
書き方

プログラミングのため
の仕掛け

宿題 1 を動かしてみる

開発環境構築

Python3 環境構築
動作確認

- m 進数から n 進数への変換
- 手始めに 10 進数から 2 進数への変換

Example (10 進 \Leftrightarrow 2 進)

$$\begin{array}{r} 2)19 \dots 1 \\ \hline 2)9 \dots 1 \\ \hline 2)4 \dots 0 \\ \hline 2)2 \dots 0 \\ \hline 2)1 \dots 1 \\ \hline 0 \end{array} \begin{array}{l} \text{Low} \\ \\ \\ \\ \text{High} \end{array}$$

$$\begin{array}{r} 1 \times 2^0 = 1 \\ \hline 1 \times 2^1 = 2 \\ \hline 0 \times 2^2 = 0 \\ \hline 0 \times 2^3 = 0 \\ \hline 1 \times 2^4 = 16 \\ \hline 19 \end{array}$$

Quiz: 10 進表記 \Leftrightarrow 2 進表記 の変換

elementaryCS-
1st

Naoyuki
Nagatou

はじめに

課題 1 の目標とテーマ
CS のこころ

データは数で
ある

Bit と Byte
自然数の n 進表記

計算 $\equiv \pm 1$ と繰
り返し

計算とは
計算の基本要素

プログラムの
書き方

プログラミングのため
の仕掛け

宿題 1 を動かしてみる

開発環境構築

Python3 環境構築
動作確認

Quiz

- ① $131_{10}, 112_{10}$ を 2 進数に変換してみてください
- ② 1 で得られた 2 進数を 10 進表記に戻してください
- ③ もとの 10 進数が得られれば正しく変換できています
- ④ この数字は東工大に割り当てられた IP アドレスになります

Outline

elementaryCS-
1st

Naoyuki
Nagatou

はじめに

課題 1 の目標とテーマ
CS のこころ

データは数で
ある

Bit と Byte
自然数の n 進表記

計算 $= \pm 1$ と繰
り返し

計算とは
計算の基本要素

プログラムの
書き方

プログラミングのため
の仕掛け

宿題 1 を動かしてみる

開発環境構築

Python3 環境構築
動作確認

3

はじめに

- 課題 1 の目標とテーマ
- CS のこころ

4

データは数である

- Bit と Byte
- 自然数の n 進表記

5

計算 $= \pm 1$ と繰り返し

- 計算とは
- 計算の基本要素

6

プログラムの書き方

- プログラミングのための仕掛け
- 宿題 1 を動かしてみる

7

開発環境構築

- Python3 環境構築
- 動作確認

計算とは

elementaryCS-
1st

Naoyuki
Nagatou

はじめに

課題 1 の目標とテーマ
CS のこころ

データは数で
ある

Bit と Byte
自然数の n 進表記

計算 $= \pm 1$ と繰
り返し

計算とは
計算の基本要素

プログラムの
書き方

プログラミングのため
の仕掛け

宿題 1 を動かしてみる

開発環境構築

Python3 環境構築
動作確認

- つぎは計算について
- 計算には入力と出力があって
- 入力と出力の関係をとくに関数とよぶ
- この場合、ある入力に対して出力は一意に決まる
- また関数には適当な名前をつけることができる
- 関数を合成してより複雑な計算を実現していく
- ここまではまだ入出力の関係としか云っていない

Example (最大公約数)

最大公約数 $\gcd(n, m)$ は n, m の公約数で最大ものと定義されるが、どう求めるか方法は書いていない

計算の方法

elementaryCS-
1st

Naoyuki
Nagatou

- 計算の方法について考える
- 計算機科学では関数といった時はこちらの意味
- 計算の方法のことをアルゴリズム (algorithm) といっている
- アルゴリズムとデータを特定の形式で書いたものがプログラム

Listing 1: 最大公約数

```

1 # Greatest common divisor
2 # Input: 自然数 x, y
3 # Output: gcd(x,y)
4 ###
5
6 def gcd(x,y):
7     ans=1
8     n=min(x,y)
9     for i in range(1,n):
10         if (x%i==0) and (y%i==0):
11             ans=i
12     return (ans)
13 print(gcd(x,y))

```

計算の基本要素

計算 = ± 1 と繰り返し

elementaryCS-
1st

Naoyuki
Nagatou

はじめに

課題 1 の目標とテーマ
CS のこころ

データは数で
ある

Bit と Byte
自然数の n 進表記

計算 = ± 1 と繰
り返し

計算とは
計算の基本要素

プログラムの
書き方

プログラミングのため
の仕掛け

宿題 1 を動かしてみる

開発環境構築

Python3 環境構築
動作確認

- 合成していくとして最も基本となるものは何か
- 結論から云ってしまえば、ある値に ± 1 する操作と繰り返しと条件分岐である

Listing 2: 加算

```
1 # add.py
2 # Input: 自然数 a, b
3 # Output: a + b
4 ###
5
6 a = int(input("? ")) # 入力された自然数を a に代入
7 b = int(input("? ")) # 入力された自然数を b に代入
8 wa = a                # a の値を wa に代入
9 while b > 0:           # b が 0 より大きい間は end までを繰り返す
10     wa = wa + 1        # wa + 1 の値を wa に代入
11     b = b - 1          # b - 1 の値を b に代入
12 print(wa)             # wa の値を出力
```

基本要素だけの乗算

elementaryCS-
1st

計算とは

計算の基本要素

Listing 3: 乗算

```

1 # mult_basically.py
2 # Input: 自然数 x, y
3 # Output: x * y
4 ###
5
6 x = int(input("x? "))
7 y = int(input("y? "))
8 seki = 0
9 while y > 0:
10     a = seki
11     b = x
12     wa = a
13     while b > 0:
14         wa = wa + 1
15         b = b - 1
16     seki = wa
17     y = y - 1
18 print(seki)

```

入力された自然数を x に代入
 # 入力された自然数を y に代入
 # seki を 0 で初期化
 # y が 0 より大きい間は end までを繰り返す

和のプログラム add.py を挿入

wa の値 (seki + x) を seki に代入
 # y - 1 の値を y に代入
 # seki の値を出力

[Back to composit-slide](#)

Back to while-slide

Outline

elementaryCS-
1st

Naoyuki
Nagatou

はじめに

課題 1 の目標とテーマ
CS のこころ

データは数で
ある

Bit と Byte
自然数の n 進表記

計算 $= \pm 1$ と繰
り返し

計算とは
計算の基本要素

プログラムの
書き方

プログラミングのため
の仕掛け
宿題 1 を動かしてみる

開発環境構築

Python3 環境構築
動作確認

3

はじめに

- 課題 1 の目標とテーマ
- CS のこころ

4

データは数である

- Bit と Byte
- 自然数の n 進表記

5

計算 $= \pm 1$ と繰り返し

- 計算とは
- 計算の基本要素

6

プログラムの書き方

- プログラミングのための仕掛け
- 宿題 1 を動かしてみる

7

開発環境構築

- Python3 環境構築
- 動作確認

プログラミングのための仕掛け

elementaryCS-
1st

Naoyuki
Nagatou

はじめに

課題 1 の目標とテーマ
CS のこころ

データは数で
ある

Bit と Byte
自然数の n 進表記

計算 $= \pm 1$ と繰
り返し

計算とは
計算の基本要素

プログラムの
書き方

プログラミングのため
の仕掛け

宿題 1 を動かしてみる

開発環境構築

Python3 環境構築
動作確認

- プログラミング言語にはプログラミングのための 3 つの仕掛けがある
 - 基本式: プログラミングに関わる最も単純なもの
 - 組合せ法: 単純なものからより複雑なものをつくる (構文として定義されている)
 - 抽象化法: 合成物に名前をつけ基本式と同様に扱う
- データについても同様

Python における基本式

elementaryCS-
1st

Naoyuki
Nagatou

はじめに

課題 1 の目標とテーマ
CS のここ

データは数で
ある

Bit と Byte
自然数の n 進表記

計算 = ± 1 と繰
り返し

計算とは
計算の基本要素

プログラムの
書き方

プログラミングのため
の仕掛け

宿題 1 を動かしてみる

開発環境構築

Python3 環境構築
動作確認

- 式の基本要素は 整数, 実数がある
 - 整数の例: 286, 386, 486
 - 講義では自然数だけあつかうので注意
- 基本要素と演算子 $+$, $-$, $*$, $//$, $\%$, $**$ などを組み合わせて式をつくる
 - 数の例: $286 + 386, 486$
 - 入れ子にできます: $286 + (386 + 486)$
- 式には名前をつけることができ, この名前のことを変数と呼びます
- その名前で参照することもできます
 - 例: $abc = 286 + 386$
 - 例: $efg = abc + 486$
- $=$ は論理記号ではなくて代入をあらわすので注意
- abc は $a \times b \times c$ ではなく変数名なので注意

Python における合成

elementaryCS-
1st

Naoyuki
Nagatou

はじめに

課題 1 の目標とテーマ
CS のこころ

データは数で
ある

Bit と Byte
自然数の n 進表記

計算 $= \pm 1$ と繰
り返し

計算とは
計算の基本要素

プログラムの
書き方

プログラミングのため
の仕掛け

宿題 1 を動かしてみる

開発環境構築

Python3 環境構築
動作確認

- 式を順番に並べる
- 実行は上から下へ、左から右に順番に実行される
- 前に実行された式の結果は変数を持ちいて参照
- 実行順序を変えたいときは **while** や **if** を使う
- [Jump to an example](#)

if 文

elementaryCS-
1st

Naoyuki
Nagatou

はじめに

課題 1 の目標とテーマ
CS のこころ

データは数で
ある

Bit と Byte
自然数の n 進表記

計算 $= \pm 1$ と繰
り返し

計算とは
計算の基本要素

プログラムの
書き方

プログラミングのため
の仕掛け

宿題 1 を動かしてみる

開発環境構築

Python3 環境構築
動作確認

- 条件によって実行順序を変更する
- [Jump to an example](#)
- 条件式は [Python ドキュメント 6.10, 6.11, 6.12 節](#) 参照

while 文

elementaryCS-
1st

Naoyuki
Nagatou

はじめに

課題 1 の目標とテーマ
CS のこころ

データは数で
ある

Bit と Byte
自然数の n 進表記

計算 $= \pm 1$ と繰
り返し

計算とは
計算の基本要素

プログラムの
書き方

プログラミングのため
の仕掛け

宿題 1 を動かしてみる

開発環境構築

Python3 環境構築
動作確認

- 実行列を繰り返し実行する

- [Jump to an example](#)

宿題 1 を動かしてみる

宿題 1 ± 1 だけで四則演算を作る

elementaryCS-1st

Naoyuki
Nagatou

はじめに

課題 1 の目標とテーマ
CS のこころ

データは数である

Bit と Byte
自然数の n 進表記

計算 ± 1 と繰り返し

計算とは
計算の基本要素

プログラムの書き方

プログラミングのための仕掛け

宿題 1 を動かしてみる

開発環境構築

Python3 環境構築
動作確認

- <https://sites.google.com/a/presystems.xyz/sample/home/elementary-computer-science> に `div-skeleton.py` と `sub-skeleton.py` があるはずなのでそれを完成させて実行する
- OCW-i から提出
- ファイル名に日本語文字は使用しないでください

Listing 4: sub.py

```
1 # sub.py
2 # Input: 自然数 a, b
3 # Output: a - b
4
5 a = int(input("a? "))
6 b = int(input("b? "))
7 sa = a
8 while :
9     sa =
10     b =
11 print(sa)
```

Listing 5: div.py

```
1 # div.py
2 # Input: 自然数 x, y
3 # Output: x ÷ y の商と余り
4
5 x = int(input("x? "))
6 y = int(input("y? "))
7 shou =
8 amari =
9 while :
10     shou =
11     amari = amari - y
12 print(shou)
13 print(amari)
```

Outline

elementaryCS-
1st

Naoyuki
Nagatou

はじめに

課題 1 の目標とテーマ
CS のこころ

データは数で
ある

Bit と Byte
自然数の n 進表記

計算 $= \pm 1$ と繰
り返し

計算とは
計算の基本要素

プログラムの
書き方

プログラミングのための
仕掛け

宿題 1 を動かしてみる

開発環境構築

Python3 環境構築
動作確認

- 3 はじめに
 - 課題 1 の目標とテーマ
 - CS のこころ

- 4 データは数である
 - Bit と Byte
 - 自然数の n 進表記

- 5 計算 $= \pm 1$ と繰り返し
 - 計算とは
 - 計算の基本要素

- 6 プログラムの書き方
 - プログラミングのための仕掛け
 - 宿題 1 を動かしてみる

- 7 開発環境構築
 - Python3 環境構築
 - 動作確認

環境構築

Python3

elementaryCS-
1st

Naoyuki
Nagatou

はじめに

課題 1 の目標とテーマ
CS のこころ

データは数で
ある

Bit と Byte
自然数の n 進表記

計算 $= \pm 1$ と繰
り返し

計算とは
計算の基本要素

プログラムの
書き方

プログラミングのため
の仕掛け

宿題 1 を動かしてみる

開発環境構築

Python3 環境構築
動作確認

- Python を利用するためには各自の PC 上に環境を構築する必要があります
- すでに python 開発環境を持っている人は以下の作業は不要です
- 各状況に合わせて環境を作っていきます
 - Cloud で利用したい方は <https://www.pythonanywhere.com/> で beginner account を作成（無料でインストール不要です）
 - Windows, Mac OSX, Linux で自分の PC に環境を作りたい方は <https://www.python.jp/> を参照

Pythonanywhere のアカウント開設

elementaryCS-1st

Naoyuki
Nagatou

はじめに

課題 1 の目標とテーマ
CS のこころ

データは数で
ある

Bit と Byte
自然数の n 進表記

計算 $= \pm 1$ と繰
り返し

計算とは
計算の基本要素

プログラムの
書き方

プログラミングのための
仕掛け

宿題 1 を動かしてみる

開発環境構築

Python3 環境構築
動作確認

- “Start running Python on line in less than a minute” をクリック



Host, run, and code Python in the cloud!

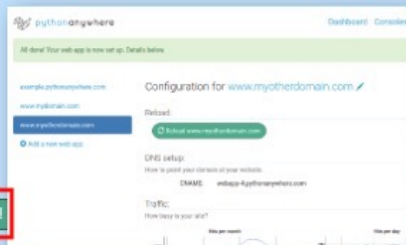
Get started for free. Our basic plan gives you access to machines with a **full Python environment** already installed. You can develop and host your website or any other code directly from your browser without having to install software or manage your own server.

Need more power? Upgraded plans start at \$5/month.

Start running Python online in less than a minute!

[Watch our one-minute video ▶](#)

Not convinced? [Read what our users are saying!](#)



Pythonanywhere のアカウント作成

elementaryCS-
1st

Naoyuki
Nagatou

はじめに

課題 1 の目標とテーマ
CS のこころ

データは数で
ある

Bit と Byte
自然数の n 進表記

計算 $= \pm 1$ と繰
り返し

計算とは
計算の基本要素

プログラムの
書き方

プログラミングのため
の仕掛け

宿題 1 を動かしてみる

開発環境構築

Python3 環境構築
動作確認

- “Create a Beginner account” をクリック
- 無料のアカウントを作成します



Plans and pricing

Beginner: Free!

A limited account with one web app at `your-username.pythonanywhere.com`, restricted outbound Internet access from your apps, low CPU/bandwidth, no IPython/Jupyter notebook support.

It works and it's a great way to get started!

Create a Beginner account

Pythonanywhere のアカウント登録

elementaryCS-
1st

Naoyuki
Nagatou

Python3 環境構築

- “Username”を入力（任意）
- “Email”を入力（m ドメイン以外でもかまいません）
- “Password”を入力
- “I agree...”にチェック
- “Register”をクリック



Create your account

Username:

Email:

Password:

Password (again):

チェック→

I agree to the [Terms and Conditions](#) and the [Privacy and Cookies Policy](#), and confirm that I am at least 13 years old.

[Register](#)

We promise not to spam or pass your details on to anyone else.

登録後の画面

- 登録アドレスにメールが届くので確認
- 以下のような画面が見えれば OK

登録後の画面

The screenshot shows the PythonAnywhere Dashboard for user 'nagatou'. At the top, there are navigation links: Dashboard, Consoles, Files, Web, Tasks, and Databases. The main heading is 'Dashboard' with a 'Welcome, nagatou' message on the right. Below this, there are two status bars: 'CPU Usage: 2% used - 2.53s of 100s. Resets in 17 hours, 38 minutes' and 'File storage: 0% full - 700.0 KB of your 512.0 MB quota'. An 'Upgrade Account' button is visible on the right. The dashboard is divided into four main sections: 'Recent Consoles', 'Recent Files', 'Recent Notebooks', and 'All Web apps'. The 'Recent Consoles' section shows 'You have no recent consoles.' with a 'View all' button. The 'Recent Files' section lists several files: '/home/nagatou/test.py', '/home/nagatou/MatplotlibTraining/fig2-...', '/home/nagatou/FiguresTraining/fig2-c...', '/home/nagatou/Mathplotlib-Training/fig...', and '/home/nagatou/.bashrc'. Below the list are buttons for '+ Open another file' and 'Browse files'. The 'Recent Notebooks' section shows a message: 'Your account does not support Jupyter Notebooks. Upgrade your account to get access!' with an 'Upgrade your account' link. The 'All Web apps' section shows 'You don't have any web apps.' with an 'Open Web app' button. At the bottom of the dashboard, there is a 'New console:' section with buttons for '\$ Bash', '>>> Python', and 'More...'. The footer of the slide contains navigation icons.

elementaryCS-
1st

Naoyuki
Nagatou

はじめに

課題 1 の目標とテーマ
CS のここ

データは数で
ある

Bit と Byte
自然数の n 進表記

計算 $= \pm 1$ と繰
り返し

計算とは
計算の基本要素

プログラムの
書き方

プログラミングのための
仕掛け
宿題 1 を動かしてみる

開発環境構築

Python3 環境構築
動作確認

自身の PC 上にインストールひと

elementaryCS-
1st

Naoyuki
Nagatou

はじめに

課題 1 の目標とテーマ
CS のこころ

データは数で
ある

Bit と Byte
自然数の n 進表記

計算 $= \pm 1$ と繰
り返し

計算とは
計算の基本要素

プログラムの
書き方

プログラミングのため
の仕掛け

宿題 1 を動かしてみる

開発環境構築

Python3 環境構築
動作確認

- Idle を起動してください

— Mac, Linux のひと —

- ターミナルを起動して `idle3` と入力

```
> idle3
```

— Windows のひと —

- スタートメニューから IDLE を起動

開発統合環境

Idle と Pythonanywhere

elementaryCS-1st

Naoyuki
Nagatou

はじめに

課題 1 の目標とテーマ
CS のところ

データは数である

Bit と Byte
自然数の n 進表記

計算 $= \pm 1$ と繰り返し

計算とは
計算の基本要素

プログラムの書き方

プログラミングのための仕掛け

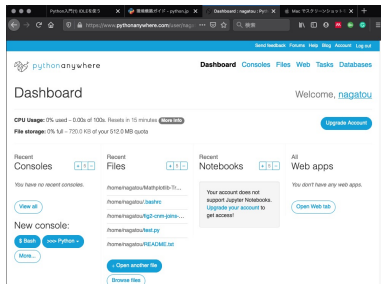
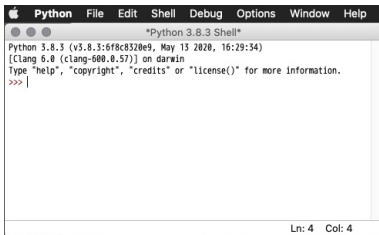
宿題 1 を動かしてみる

開発環境構築

Python3 環境構築

動作確認

- Mac, Windows, Linux の人もこれで準備ができました
- それぞれ以下のような画面が見えるはずです
- 以後、いずれかの開発統合環境を利用していきます
- Pythonanywhere や IDLE は編集、実行が統合された環境を提供しています
- IDLE の使い方は [Python 公式](#) を参照
- http://www.isc.meiji.ac.jp/~mizutani/python/intro1_python.html も参考になるかも



開発環境テスト用コード

elementaryCS-
1st

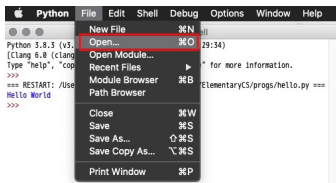
Naoyuki
Nagatou

● <https://sites.google.com/a/presystems.xyz/sample/home/elementary-computer-science> から

hello.py をダウンロード

— IDLE 利用のひと —

● File->Open で hello.py を開く



— Pythonanywhere 利用のひと —

- 右上 “Files” をクリック
- “upload” をクリックして hello.py をアップロード
- hello.py をクリックすると中が見えます



はじめに

課題 1 の目標とテーマ
CS のここ

データは数で
ある

Bit と Byte
自然数の n 進表記

計算 ± 1 と繰
り返し

計算とは
計算の基本要素

プログラムの
書き方

プログラミングのた
めの仕掛け

宿題 1 を動かしてみる

開発環境構築

Python3 環境構築
動作確認

プログラムの実行

elementaryCS-1st

Naoyuki
Nagatou

はじめに

課題 1 の目標とテーマ
CS のこころ

データは数である

Bit と Byte

自然数の n 進表記

計算 $= \pm 1$ と繰り返し

計算とは

計算の基本要素

プログラムの書き方

プログラミングのための仕掛け

宿題 1 を動かしてみる

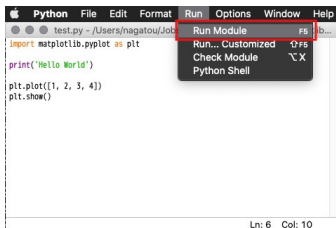
開発環境構築

Python3 環境構築

動作確認

IDLE 利用のひと

- Run->Run Module をクリック
- “Hello World” が表示されれば正常



Pythonanywhere 利用のひと

- Run this file をクリック
- 下半分の黒い画面に “Hello World” と表示されれば正常
- 下半分の黒い画面で exit() と入力してください



Part III

CS 第 1—課題 1

CS 第 1 — 課題 1

elementaryCS-
1st

Naoyuki
Nagatou

課題 1 演習ガ
イド

課題 1 の説明

8

課題 1 演習ガイド

- 課題 1 の説明

Outline

elementaryCS-
1st

Naoyuki
Nagatou

課題 1 演習ガ
イド

課題 1 の説明

8

課題 1 演習ガイド

- 課題 1 の説明

課題 1 の説明

四則演算でアニメーション

elementaryCS-
1st

Naoyuki
Nagatou

課題 1 演習ガ
イド

課題 1 の説明

- 課題: 四則演算でアニメーションを作成してください
 - 動きがあること
 - 計算だけで動かすこと
- 提出物:
 - 作成したアニメーションプログラムのソースコード (anime.py)
 - 作成したアニメーションプログラムの使い方の説明
 - 作成したアニメーションプログラムの計算の仕組みの説明
 - 工夫した点も書くこと
- 採点者はソースコードを読まないと仮定して説明すること
- Python 言語の説明は不要
- 提出は OCW-i から
- 期限は来週のこの時間まで

ひつじさんを動かしてみる

elementaryCS-
1st

課題1の説明

- 大きな数字を定義して
- 各桁をひとつの画素とみなす
- 各桁は 0-9 でこの違いで絵にする
- 14 個の自然数
- 先頭は 1 にしないとずれる
- 動かすときは桁をシフトさせて (ここでは 10 で割っている) 動かす

Listing 6: sheep.py (declaration)

[illegible]

ひつじさんを動かしてみる

elementaryCS-
1st

Naoyuki
Nagatou

課題 1 演習ガ
イド

課題 1 の説明

- 動かすときは桁をシフトさせて (ここでは 10 で割っている) 動かす
- `a0` を足しているのは絵がずれないようにするため

Listing 7: sheep.py (shift)

```
18 # shift #
19 a0 = d0
20
21 a1 = (a1 - a0) // 10 + a0
22 a2 = (a2 - a0) // 10 + a0
23 a3 = (a3 - a0) // 10 + a0
24 a4 = (a4 - a0) // 10 + a0
25 a5 = (a5 - a0) // 10 + a0
26 a6 = (a6 - a0) // 10 + a0
27 a7 = (a7 - a0) // 10 + a0
28 a8 = (a8 - a0) // 10 + a0
29 a9 = (a9 - a0) // 10 + a0
30 a10 = (a10 - a0) // 10 + a0
31 a11 = (a11 - a0) // 10 + a0
32 a12 = (a12 - a0) // 10 + a0
33 a13 = (a13 - a0) // 10 + a0
34 a14 = (a14 - a0) // 10 + a0
35 a15 = (a15 - a0) // 10 + a0
36 a16 = (a16 - a0) // 10 + a0
37 a17 = (a17 - a0) // 10 + a0
38 a18 = (a18 - a0) // 10 + a0
39 a19 = (a19 - a0) // 10 + a0
40 a20 = (a20 - a0) // 10 + a0
```

Part IV

配列

配列

elementaryCS-
1st

Naoyuki
Nagatou

課題 2 演習ガ
イド

配列

文字列

課題 2 予告

9

課題 2 演習ガイド

- 配列
- 文字列
- 課題 2 予告

Outline

elementaryCS-
1st

Naoyuki
Nagatou

課題 2 演習ガ
イド

配列

文字列

課題 2 予告

9

課題 2 演習ガイド

- 配列
- 文字列
- 課題 2 予告

配列

- ◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ↺ 🔍 ↻

配列の参照と代入

elementaryCS-
1st

Naoyuki
Nagatou

課題 2 演習ガ
イド

配列

文字列

課題 2 予告

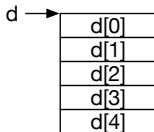
- 連続したメモリ領域を確保
- 先頭を 0 としてインデックスで参照 (ここは板書します)

Listing 10: 代入

```
1 d = [0]*9
2 d[0]=10000000000000000000000000000000
3 d[1]=10000000000110000011000000000000
4 d[2]=10000000000110000011000000000000
5 d[3]=10000000000000000000000000000000
6 d[4]=10000001100000000000000110000000
7 d[5]=10000000110000000000000110000000
8 d[6]=10000000011000000000000110000000
9 d[7]=10000000000111111110000000000000
10 d[8]=10000000000000000000000000000000
```

Listing 11: 参照

```
1 for i in range(9):
2     print(d[i])
```



配列の例

elementaryCS-
1st

Naoyuki
Nagatou

課題 2 演習ガ
イド
配列
文字列
課題 2 予告

- 総和を求める
- Listing 12: 整数の配列を宣言する例
- Listing 13: 入力した整数を配列にする例

Listing 12: 総和

```
1 a=[2,4,6,8,10,12]
2 s=0
3 for k in range(len(a)):
4     s=s+a[k]
5     k=k+1
6 print(s)
```

Listing 13: 入力した数の総和

```
1 a = list(map(int,input("numbers? ").split()))
2 s=0
3 for k in range(len(a)):
4     s=s+a[k]
5     k=k+1
6 print(s)
```

最大値を求める

宿題 2

elementaryCS-
1st

Naoyuki
Nagatou

課題 2 演習ガ
イド

配列

文字列

課題 2 予告

- <https://sites.google.com/a/presystems.xyz/sample/home/elementary-computer-science> の

max-skeleton.py を完成させる

- 最大値 = $\max(a_1, a_2, \dots, a_n)$
- `for j in range(0, n)` は `j` を 0 から `n-1` まで繰り返す
という意味

Listing 14: max.py

```
1 # max.py
2 # 入力: 整数の列
3 # 出力: 最大値
4 array = list(map(int, input("numbers? ").split()))
5 if not array: # array が空の場合
6     raise ValueError("...") # 入力エラー
7 # 以下が計算部分
8 max_value = array[0] # array[0] を一時的に
   最大値に
9 max_index = 0
10 for i in range(0, len(array)):
11     if: # より大きな数を探す
12         max_value =
13         max_index =
14 print(max_value, max_index)
```

出力例

```
> python3 max.py
numbers? -3 8 19 -4
19 2
```

文字データの表現

elementaryCS-
1st

Naoyuki
Nagatou

課題 2 演習ガ
イド

配列

文字列

課題 2 予告

- コンピュータは数値だけでなく文字も処理することができる
- 文字はコード化されて処理される
- 文字列は文字の配列

文字コード

elementaryCS-
1st

Naoyuki
Nagatou

課題 2 演習ガ
イド

配列

文字列

課題 2 予告

- コードとは、文字や記号をコンピュータで扱うための記号です
- コンピュータ内では適当な正整数が文字や記号に割り振られています
 - 整数は2進数で表されているので0と1の列にコード化されます
- コードは任意に決めることもできます
- しかし、各コンピュータで違っては不都合が生じます
- 異なるコンピュータでは全く違った文字になってしまうかも知れません
 - 文字コードが違っているとうまく表示できません

コードの違いの例

elementaryCS-
1st

Naoyuki
Nagatou

課題 2 演習ガ
イド

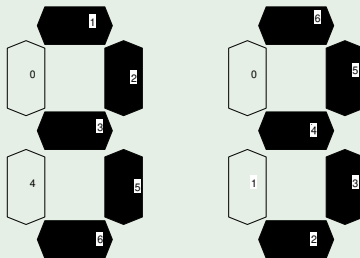
配列

文字列

課題 2 予告

- 3 という文字のコードの例です
- 下の図は電光掲示板の例です
- 左の図では $110\ 1110_{(2)}$ とコードを割り当てています
- 右の図では $111\ 1100_{(2)}$ とコードを割り当てています
- このように違ったコードを対応づけることもできます

Example (電光掲示板の例)



ASCII コード

elementaryCS-
1st

Naoyuki
Nagatou

課題 2 演習ガ
イド

配列

文字列

課題 2 予告

- 共通のコード体系として **ASCII** コードが策定されました
- これは英語のアルファベットと数字と記号にコードを割り当てています
 - ASCII Code Chart

日本語漢字のコード体系

elementaryCS-
1st

Naoyuki
Nagatou

課題 2 演習ガ
イド

配列

文字列

課題 2 予告

- 現在は日本語など漢字圏の文字もコードが割り当てられています
- JIS, Shift-JIS, EUC, Unicode があります
 - これらは異なるコード体系です
 - 同じ文字でも異なるコードが割り当てられています
- 漢字圏で Unicode 以前に用いていたコードの規格
 - 日本: JIS X 0208-1990, JIS X 0212-1990(第一水準, 第二水準, 補助漢字)
 - 中国: GB 2312-80, GB 12345-90...
 - 台湾: CNS 111643-1986
 - 韓国: KS C 5601-1987, KS C 5657-1991
- Python3 は Unicode を利用しています

Python での文字列

elementaryCS-
1st

Naoyuki
Nagatou

課題 2 演習ガ
イド

配列

文字列

課題 2 予告

- 文字列は各文字の配列として扱える
- `str` という名前の配列の各要素に一文字入っている
- 0 番目から順番にインデックスで参照できる

Listing 15: stringPrint.py

```
1 # stringPrint.py
2 # 文字列処理の練習プログラム
3 # 入力: 文字列
4 # 出力: 文字列の文字を 1 行 1 文字で出す
5 import os
6
7 os.system('clear')
8 str = (input("strings? ")).encode("ascii")
9 print(str)
10 for k in range(0, len(str)):
11     print(chr(str[k]), hex(str[k]))
```

出力例

```
> python3 stringPrint.py
strings? Ice%cream
l 0x49
c 0x63
e 0x65
% 0x25
c 0x63
r 0x72
e 0x65
a 0x61
m 0x6d
```

文字列の表現

elementaryCS-
1st

Naoyuki
Nagatou

課題 2 演習ガ
イド

配列

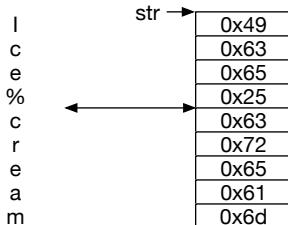
文字列

課題 2 予告

- 文字列は文字の配列
- 各文字はその符号 (文字コード) で表されて
- 各要素に各文字コードを格納

出力例

```
> python3 stringPrint.py  
strings? Ice%cream  
l 0x49  
c 0x63  
e 0x65  
% 0x25  
c 0x63  
r 0x72  
e 0x65  
a 0x61  
m 0x6d
```



英小文字だけを画面に出力

宿題 3

elementaryCS-
1st

Naoyuki
Nagatou

課題 2 演習ガ
イド

配列

文字列

課題 2 予告

● <https://sites.google.com/a/presystems.xyz/sample/home/elementary-computer-science> の

abcPrint-skeleton.py を完成させる

Listing 16: abcPrint.py

```
1 # abcPrint.py
2 # 文字列処理の練習プログラム, 小文字だけ出力
3 # 入力: 文字列
4 # 出力: 文字列の文字で小文字のみ出力する
5 ss = (input("strings? ")).encode("ascii")
6 for k, code in enumerate(ss):
7     if :                # 小文字ならば
8         print(chr(ss[k])) # 文字を表示する
```

出力例

```
> python3 abcPrint.py
strings? lce%cream
c
e
c
r
e
a
m
```

課題 2 予告

elementaryCS-
1st

Naoyuki
Nagatou

課題 2 演習ガ
イド

配列

文字列

課題 2 予告

- 来週の予定です

- [Jump to Quiz 2](#)

Part V

CS 第 1—課題 2

CS 第 1 — 課題 2

elementaryCS-
1st

Naoyuki
Nagatou

宿題 2, 3

課題 2 演習ガ
イド

課題 2

課題 S 演習ガ
イド

課題 S

10 宿題 2, 3

11 課題 2 演習ガイド
● 課題 2

12 課題 S 演習ガイド
● 課題 S

Outline

elementaryCS-
1st

Naoyuki
Nagatou

宿題 2, 3

課題 2 演習ガ
イド

課題 2

課題 S 演習ガ
イド

課題 S

10 宿題 2, 3

11 課題 2 演習ガイド
● 課題 2

12 課題 S 演習ガイド
● 課題 S

Outline

elementaryCS-
1st

Naoyuki
Nagatou

宿題 2, 3

課題 2 演習ガ
イド

課題 2

課題 S 演習ガ
イド

課題 S

10 宿題 2, 3

11 課題 2 演習ガイド
● 課題 2

12 課題 S 演習ガイド
● 課題 S

課題 2 テーマ

elementaryCS-
1st

Naoyuki
Nagatou

宿題 2, 3

課題 2 演習ガ
イド
課題 2

課題 S 演習ガ
イド
課題 S

- 前回までにみたように配列は複数のデータを統一的に扱う方法を提供した
- それ以外の例もこの課題で見ていくことにします

やってほしいこと

- 今、自然数の組で表わされている有理数を小数表記に変換するプログラムを作ろうとしています
- 配列を使って循環小数になっても停止するようにプログラム `junkan.py` を改良してください
- ただし、分子は 1 に固定する

- まずは `junkan.py` を実行してみてください
- あまりは 0 から $d-1$ の有限の範囲なので, 10 倍して割るを繰り返しているとどこかで同じあまりが出てくるはず

Listing 17: junkan.py

```
1 # junkan.py
2 # 配列の使い方の練習 (循環小数を循環するまで求める)
3 # 入力: d
4 # 出力: 1/d の各桁を循環するまで求める
5 d = int(input("1/d d(>=2)? "))
6 print("1/", d, " を求めます")
7 x = 1
8 print("0.", end=""),
9 while (True):
10     x = x * 10
11     q = x // d
12     x = x % d
13     print(q, end="")
14     if x == 0:
15         break
16 print("")
```

Outline

elementaryCS-
1st

Naoyuki
Nagatou

宿題 2, 3

課題 2 演習ガ
イド

課題 2

課題 S 演習ガ
イド

課題 S

10 宿題 2, 3

11 課題 2 演習ガイド
● 課題 2

12 課題 S 演習ガイド
● 課題 S

課題 S

elementaryCS-
1st

Naoyuki
Nagatou

宿題 2, 3

課題 2 演習ガ
イド

課題 2

課題 S 演習ガ
イド

課題 S

- 自然数上の演算を整数に拡張
- <https://sites.google.com/a/presystems.xyz/sample/home/elementary-computer-science> からまずは `integers-skeleton.py` をダウンロード
- まだ、お話していない部分も含まれるのですぐに取り掛からなくていいです
- ソースコード中のコメントを参照して (1)-(4) が虫食いになっているので完成させる
- 提出は出来たところまででいいです
- 出来なかったところはコメントにしてください

Part VI

関数とサブルーチン

関数とサブルーチン

elementaryCS-
1st

Naoyuki
Nagatou

関数, メソッド,
サブルーチン

関数宣言
計算の抽象化

関数を使って
世の中の事象
を抽象化

暗号方式のおはなし
暗号通信のプログラム

13 関数, メソッド, サブルーチン

- 関数宣言
- 計算の抽象化

14 関数を使って世の中の事象を抽象化

- 暗号方式のおはなし
- 暗号通信のプログラム

Outline

elementaryCS-
1st

Naoyuki
Nagatou

関数，メソッド，サブルーチン

関数宣言
計算の抽象化

関数を使って
世の中の事象
を抽象化

暗号方式のおはなし
暗号通信のプログラム

- 13 関数，メソッド，サブルーチン
 - 関数宣言
 - 計算の抽象化

- 14 関数を使って世の中の事象を抽象化
 - 暗号方式のおはなし
 - 暗号通信のプログラム

関数, メソッド, サブルーチン

elementaryCS-
1st

Naoyuki
Nagatou

関数, メソッド,
サブルーチン

関数宣言
計算の抽象化

関数を使って
世の中の事象
を抽象化

暗号方式のおはなし
暗号通信のプログラム

- 抽象化の方法について触れてきた
 - 変数: 計算の結果に名前をつけるということ
 - 配列: データの集まりを名前をつけて抽象化
- 今回は計算を合成する抽象化について見てみます
- 合成したものに名前をつけて単一の手続きとして抽象化する方法

Python における合成手続き

elementaryCS-
1st

Naoyuki
Nagatou

関数, メソッド,
サブルーチン

関数宣言

計算の抽象化

関数を使って
世の中の事象
を抽象化

暗号方式のおはなし
暗号通信のプログラム

- Python では `def` というキーワードをつかって定義します
- `def` のあとに関数名と仮引数を書いて、最後にコロンの (忘れずに)
- 仮引数は関数を呼び出したときに実引数にバインドされる
- 引数はその関数内だけで有効

Listing 18: add.py

```
1 a = int(input("a? "))
2 b = int(input("b? "))
3 wa = a
4 while b > 0:
5     wa = wa + 1
6     b = b - 1
7 print(wa)
```

Listing 19: 関数定義

```
1 def add(x,y):
2     wa = x
3     while y > 0:
4         wa = wa + 1
5         y = y - 1
6     return(wa)
```

Listing 20: 関数適用

```
7 def mult(x,y):
8     seki = 0
9     while y > 0:
10         b = x
11         seki = add(seki,b)
12         y = y - 1
13     return(seki)
```

ブラックボックス抽象としての関数

elementaryCS-
1st

Naoyuki
Nagatou

関数, メソッド,
サブルーチン

関数宣言
計算の抽象化

関数を使って
世の中の事象
を抽象化

暗号方式のおはなし
暗号通信のプログラム

- “プログラムを部分に分ける”
- どう分けるかが重要になる
- 他のプログラムの部品として使え, まとまった仕事ができるようにわかる
- 部品としての関数はどう計算するかには関心をもたず, 計算結果にだけ関心を持てば良い

Listing 21: 集合演算

```
1 ### Set operations
2 def union(seta, setb, result):
3     global Size
4     for i in range(Size):
5         result[i] = seta[i] or setb[i]
6 def intersection(seta, setb, result):
7     global Size
8     for i in range(Size):
9         result[i] = seta[i] and setb[i]
```

Listing 22: 集合演算

```
10 def complement(seta, result):
11     global Size
12     for i in range(Size):
13         result[i] = not seta[i]
14 def difference(seta, setb, result):
15     global Size
16     tmp = [-1] * Size
17     for i in range(Size):
18         complement(setb, tmp)
19     intersection(seta, tmp, result)
```

仮引数と実引数

elementaryCS-
1st

Naoyuki
Nagatou

関数, メソッド,
サブルーチン

関数宣言
計算の抽象化

関数を使って
世の中の事象
を抽象化

暗号方式のおはなし
暗号通信のプログラム

- 関数は仮引数というものをもつ
- 複数の関数が同じ名前の仮引数を持っていたとしても良い
- 仮引数は関数の本体で有効である
- 関数を呼び出したときの値に束縛 (bind) されて、関数の本体では呼び出し時の値に置き換えられる
- 呼び出し時の値を実引数という
- 一般に変数は有効範囲 (scope) が決まっている
- 仮引数は関数本体が有効範囲である

Outline

elementaryCS-
1st

Naoyuki
Nagatou

関数, メソッド,
サブルーチン

関数宣言
計算の抽象化

関数を使って
世の中の事象
を抽象化

暗号方式のおはなし
暗号通信のプログラム

13 関数, メソッド, サブルーチン

- 関数宣言
- 計算の抽象化

14 関数を使って世の中の事象を抽象化

- 暗号方式のおはなし
- 暗号通信のプログラム

暗号通信

elementaryCS-
1st

Naoyuki
Nagatou

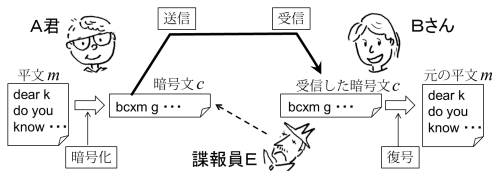
関数, メソッド,
サブルーチン

関数宣言
計算の抽象化

関数を使って
世の中の事象
を抽象化

暗号方式のおはなし
暗号通信のプログラム

- 暗号システムを例に関数としての抽象化を見ていきます



暗号方式 (Cryptography)

elementaryCS-
1st

Naoyuki
Nagatou

関数, メソッド,
サブルーチン

関数宣言
計算の抽象化

関数を使って
世の中の事象
を抽象化

暗号方式のおはなし
暗号通信のプログラム

- シーザ暗号 (Caesar cipher): ローマ皇帝シーザが使った方式
- ビジュネル暗号 (Vigenère cipher): Vigenère が作った方式
- エニグマ (enigma): 大戦中にドイツ軍が使った方式
- DES (Data Encryption Standard), RSA (Rivest, Shamir and Adleman): 現在広く利用されている方式

シーザ暗号

elementaryCS-
1st

Naoyuki
Nagatou

関数, メソッド,
サブルーチン

関数宣言
計算の抽象化

関数を使って
世の中の事象
を抽象化

暗号方式のおはなし
暗号通信のプログラム

- 文字を k 字先にシフトして暗号文を作る
- $k = 3$ とすると下図のようになる
- z, y, x は a, b, c になる
- これからこれを関数として表していく

a	b	c		z
↓	↓	↓	...	↓
d	e	f		c

暗号システムに必要な要素

elementaryCS-
1st

Naoyuki
Nagatou

関数, メソッド,
サブルーチン

関数宣言
計算の抽象化

関数を使って
世の中の事象
を抽象化

暗号方式のおはなし
暗号通信のプログラム

- M : 文の集合
- C : 暗号文の集合
- \mathcal{K} : 鍵の集合
- $\mathcal{E}: M \rightarrow C$: 暗号関数の集合
- $\mathcal{D}: C \rightarrow M$: 復号関数の集合

Example (Caesar Cipher)

- アルファベットは 0 から 25 に順番に対応付けられていると仮定する
- M : アルファベットの文字の列
- C : アルファベットの文字の列
- $\mathcal{K}: \{i \mid 0 \leq i \leq 25 \text{ であるような整数 } i\}$
- $\mathcal{E}: \{E_k \mid k \in \mathcal{K} \text{ and } \forall m(= m_1, m_2, \dots) \in M. E_k(m) = (m_i + k) \bmod 26\}$
- $\mathcal{D}: \{D_k \mid k \in \mathcal{K} \text{ and } \forall c(= c_1, c_2, \dots) \in C. D_k(k, c) = (26 + c_i - k) \bmod 26\}$

いくつかの用語

elementaryCS-
1st

Naoyuki
Nagatou

関数, メソッド, サブルーチン

関数宣言
計算の抽象化

関数を使って
世の中の事象
を抽象化

暗号方式のおはなし
暗号通信のプログラム

- $E_k \in \mathcal{E}$ を $m \in \mathcal{M}$ に適用することを暗号化 (encipher)
- $D_k \in \mathcal{D}$ を $c \in \mathcal{C}$ に適用することを復号 (decipher)
- $k \in \mathcal{K}$ を 鍵 (key)

平文 (plaintext)

Hello World

Hello World

暗号文 (ciphertext)

Hhoor Wruog

Hhoor Wruog

暗号化
→

復号
←

シーザ暗号を関数であらわす

elementaryCS-
1st

Naoyuki
Nagatou

関数, メソッド,
サブルーチン

関数宣言
計算の抽象化

関数を使って
世の中の事象
を抽象化

暗号方式のおはなし
暗号通信のプログラム

- 各文字を 0 から 25 で置き換える
 - $a \rightarrow 0, z \rightarrow 25$
- 暗号化関数 $enc(3, m) = c$: 平文 $m = m_1, m_2, \dots, m_n$, 暗号文 $c = c_1, c_2, \dots, c_n$ として $f(m_i) = (m_i + 3) \bmod 26$ と表せる
- 復号関数 $dec(3, c) = m$: 平文 $m = m_1, m_2, \dots, m_n$, 暗号文 $c = c_1, c_2, \dots, c_n$ として $f^{-1}(c_i) = ((26 + c_i) - 3) \bmod 26$ と表せる

Example (Hello のシーザ暗号)

- 暗号化は 関数 enc として $enc(3, "Hello") = "Hhoor"$ と表せる
- 復号は 関数 dec として $dec(3, "Hhoor") = "Hello"$ と表せる

プログラムにしてみる

宿題 4

elementaryCS-
1st

Naoyuki
Nagatou

関数, メソッド,
サブルーチン

関数宣言
計算の抽象化

関数を使って
世の中の事象
を抽象化
暗号方式のおはなし
暗号通信のプログラム

- 関係として仕様をあらわしたので、実際にプログラムにしてみます
- 具体的な計算として手続きを示した関数を定義します
- 暗号システムという複雑なものを分割

Listing 23: ango.py

```
1 # ango.py
2 # 暗号化サブルーチンの定義と利用
3 # 入力: 文字列
4 # 出力: 暗号化した文字列
5
6 ### Global variables
7 K = 3 # 暗号鍵の設定
8
9 # 平文を暗号化するサブルーチン
10 # enc(秘密鍵 k, 平文 m) = 暗号文 c
11 def enc(k, m):
12     ALPHABET = range(ord('a'), ord('z')+1) # 英字小文字アルファベット
13     plain = list(m.encode("ascii"))        # 文字列 -> 文字コードの配列
14     cipher = plain.copy()                  # 暗号文格納用配列
15     for i, code in enumerate(plain):
16         ###
17         # 宿題 4
18         ###
19     return(bytes(cipher).decode("ascii"))
```

宿題 4 (ango.py) のヒント

elementaryCS-
1st

Naoyuki
Nagatou

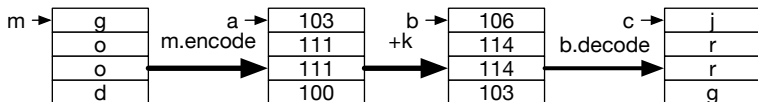
関数, メソッド,
サブルーチン

関数宣言
計算の抽象化

関数を使って
世の中の事象
を抽象化

暗号方式のおぼなし
暗号通信のプログラム

- **ASCII Code Chart** を思い出してください
- 小文字 **a** は **97 (0x61)** が割り当てられています
- アルファベットを **0 から 25 までの数字** に対応づける
 - **a** の文字コード **97** を引くと文字を **0 から 25** に対応付けることができる
- **m.encode("ascii")** で **1 バイトの文字コードの列** に変換
- **list()** で配列を作成
- 鍵 **k** 分だけ各整数にたす
- **25** をこえるときは **0** にもどって計算
- **bytes(cipher).decode("ascii")** で文字の列に変換



Python における文字列の操作

elementaryCS-
1st

Naoyuki
Nagatou

- `str.encode("ascii")` で1バイトの列に変換
- `list()` で配列を作成
- `code.py` を実行すると動作が見られます

Listing 24: code.py

```

1 # code.py
2 # 文字列処理の復習用
3 # 入力: 文字列
4 # 出力: 文字列の文字で小文字のみ、文字と各種情報を出力する
5 ALPHABET = range(ord('a'), ord('z')+1) # 英字文字アルファベット
6 bun = input("Enter a string:") # 入力文字列から改行除法
7 cc = list(bun.encode("ascii")) # 文字列 → 文字コードの配列
8 for i, moji in enumerate(bun): # mojiはbunのi文字目を得る (i は
    0 から始まる)
9     code = cc[i] # その文字のコードを得る
10    offset = code - ALPHABET[0] # 文字 a との差分
11    if code in ALPHABET: # 小文字アルファベットなら
12        print(moji, ": ", code, ", ", hex(code), ", ", offset) # 差分まで表示する
13    else: # そうでない時は
14        print(moji, ": ", code, ", ", hex(code)) # 差分は表示しない

```

復号関数

宿題 5

elementaryCS-
1st

Naoyuki
Nagatou

関数, メソッド,
サブルーチン

関数宣言
計算の抽象化

関数を使って
世の中の事象
を抽象化

暗号方式のおはなし
暗号通信のプログラム

Listing 25: hukugo.py

```
1 # hukugo.py
2 # 復号サブルーチンの定義と利用
3 # 入力: 暗号文の文字列
4 # 出力: 復号した平文
5 ### Global variables
6 K = 3 # 暗号鍵の設定
7
8 # 平文を暗号化するサブルーチン
9 # dec(秘密鍵 k, 暗号文 c) = 平文 m
10 def dec(k, c):
11     ALPHABET = range(ord('a'), ord('z')+1) # 英字小文字アルファベット
12     cipher = list(c.encode("ascii"))      # 文字列 -> 文字コードの配列
13     plain = cipher.copy()                 # 平文格納用配列
14     for i,code in enumerate(cipher):
15         ###
16         # 宿題 5
17         ###
18     return(bytes(plain).decode("ascii"))
```

宿題 4, 5

elementaryCS-
1st

Naoyuki
Nagatou

関数, メソッド,
サブルーチン

関数宣言
計算の抽象化

関数を使って
世の中の事象
を抽象化

暗号方式のおはなし
暗号通信のプログラム

- <https://sites.google.com/a/presystems.xyz/sample/home/elementary-computer-science> から
ango-skeleton.py と hukugo-skeleton.py をダウンロード
- 同様にテスト用サンプルデータ plaintext.txt と
ciphertext.txt をダウンロード (ソースコードと同じディレクトリに保存してください)
- ファイル名は暗号化プログラム: ango.py, 復号プログラム: hukugo.py としてください
- 提出は OCW-i から

Part VII

CS 第 1—課題 3

CS 第 1 — 課題 3

elementaryCS-
1st

Naoyuki
Nagatou

課題 3 演習ガ
イド

暗号解読
(Cryptoanalysis)

課題 3 のシチュエー
ション

課題 3

15

課題 3 演習ガイド

- 暗号解読 (Cryptoanalysis)
- 課題 3 のシチュエーション
- 課題 3

Outline

elementaryCS-
1st

Naoyuki
Nagatou

課題 3 演習ガイド

暗号解読
(Cryptoanalysis)

課題 3 のシチュエーション

課題 3

15

課題 3 演習ガイド

- 暗号解読 (Cryptoanalysis)
- 課題 3 のシチュエーション
- 課題 3

暗号解説

elementaryCS-
1st

Naoyuki
Nagatou

課題 3 演習ガ
イド

暗号解説
(Cryptanalysis)

課題 3 のシチュエー
ション

課題 3

- 暗号をやぶろうとすることを暗号解説 (cryptoanalysis) と呼ぶ
- 暗号アルゴリズム (\mathcal{E} と \mathcal{D}) は知っているが、鍵を知らないという前提
- Ciphertext only attack: 暗号文だけを得ることができるとして解説 (e.g. 踊る人形)
- Known plaintext attack: 平文とその暗号文を得ることができるとして解説 (e.g. エニグマで "異常なし")
- Chosen plaintext attack: 特定の平文を送り、埋め込んだ暗号文を得ることができるとして解説 (e.g. 日本軍のミッドウェイ島攻撃)

課題 3 のシチュエーション

elementaryCS-
1st

Naoyuki
Nagatou

課題 3 演習ガ
イド

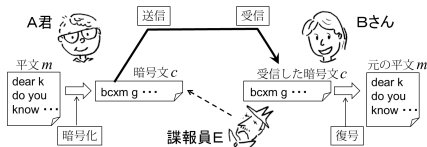
暗号解説

(Cryptanalysis)

課題 3 のシチュエー
ション

課題 3

- Alice と Bob が通信しているとして、そこに Eve (eavesdropper) がいるとします
- 通信の内容を盗み見られないように暗号化して通信します
- 暗号化の手順は以下の通り
 - ① 送りたい文 (平文 m) を暗号化 (encryption) して暗号文 c を作成
 - ② 暗号文 c を送信
 - ③ 暗号文 c を受信
 - ④ 暗号文 c を復号 (decryption) して平文 m を得る
- Eve はこの通信の内容を盗み見て暗号文だけが入手可能で暗号解読を試みる



課題 3

elementaryCS-
1st

Naoyuki
Nagatou

課題 3 演習ガ
イド

暗号解読
(Cryptanalysis)

課題 3 のシチュエー
ション

課題 3

- 暗号解読に挑戦
- <https://sites.google.com/a/presystems.xyz/sample/home/elementary-computer-science> に置いてある `kaidoku-skeleton.py` を参考に暗号解読プログラムを `kaidoku.py` を作成してください
- 課題 3 で作成してほしいプログラム
- 提出は OCW-i からソースコードを提出

暗号解読のヒント

elementaryCS-
1st

課題 3

- 意味をなす一般的な文章では文字の出現頻度には偏りがあります
- たとえば英語では母音 e が最も出現頻度が高い
- シーザ暗号はこの出現頻度は暗号化しても偏りは変わりません
- この特徴を利用して何文字移動しているかを予測することができます
- 各文字 26 個の頻度を計算するために出現回数を要素とする配列をつくる

英語の場合

一番多く現れる文字が e のはず！

qxuvbŋ qjm kŋnw bnjcnm oxa bxvŋ qxdab rw bruŋwŋl frq qrb
uxwp, qqrw kjlt ldaenm xena j lqnrvlju enbbnw rw fqlq qŋ
fjb kaŋfrwp j yjacrldujauh vjuxmxaxdb yaxmdlc. qrb qŋjm
fjb bdwt dyxw qrb kaŋjbc, jwm qŋ uxxtŋm oaxv yh vxrw xō ...

n が19回出現で最多

暗号解読のヒント—つづき

elementaryCS-
1st

Naoyuki
Nagatou

課題 3 演習ガ
イド

暗号解読

(Cryptanalysis)

課題 3 のシチュエー
ション

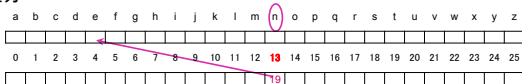
課題 3

- 13 番目の文字 n が最多なので 4 番目の文字 e にシフト
- $13 - 4 = 9$ なので 9 文字シフトしていると推測できる

qxuvnb qjm knnw bnjcnm oxa bxvn qxdab rw brunwln frcq qrb
uxwp, cqrw kjlt ldae nm xena j lqnvrlju enbbnu rw fqrlq qn
fjb kanfrwp j yjacrldujah vjuxmxaxdb yaxmdlc. qrb qnjm
fjb bdwt dyxw qrb kanjbc, jwm qn uxxt nm oaxv vh yxrcw xo ...

差分

n が19回出現で最多



暗号解読のヒント—より高度な方法

elementaryCS-
1st

Naoyuki
Nagatou

課題 3 演習ガ
イド

暗号解読

(Cryptanalysis)

課題 3 のシチュエー
ション

課題 3

- フォーマルには各文字の出現頻度と暗号文での出現頻度の相関 $\phi(i)$ をとります
- $\phi(i) = \sum_{0 \leq c \leq 25} f(c)p(c-i)$, ここで $f(c) = \frac{n_c}{l_{ct}}$ は暗号文での文字 c の出現頻度, $p(c-i)$ は一般の出現頻度とする
- <https://sites.google.com/a/presystems.xyz/sample/home/elementary-computer-science> に置いてある 1-gram.txt が出現頻度のファイルです
- 相関係数 $\phi(i)$ が
 - 1 に近いほど: $f(c)$ が大きくなれば $p(c-i)$ も大きくなり相関が強くなる
 - 0 近傍: $f(c)$ と $p(c-i)$ はあまり相関がない