(3)「コンピュータとプログラミング」で学ぶこと

- 要するに
  - コンピュータの構造と論理演算
  - 計算誤差
  - センサやアクチュエータとの接続
  - プログラムの基本要素
  - 応用プログラムを書くための道具
  - 探索や並べ替えの手順
  - プログラムによるシミュレーション

学習14, 15, 16, 17

学習14 応用的プログラム

(1)リスト

(2)乱数

(3)関数

(4)WebAPI

リスト

- リストとは
  - 複数の値を一つの名前（変数名）と通し番号（添字）によって管理する仕組み
    - 例: `a=[56,3,62,17,87]　　a[0]:56, a[1]:3,...`
  - リストの中にあるひとつひとつの値を要素という
  - 要素に付けられた0から始まる番号を添字という
- リストを用いたプログラムの例
  - リストの一部を取り出し演算するプログラム
  - リスト内の全要素を足し合わせるプログラム

リストの一部を取り出し演算するプログラム

```
a = [56,3,62,17,87,22,36,83,21,12] # リストa の定義
goukei = 0
goukei = a[3] + a[7]
print(goukei)
```

```
a[3] : 17
a[7] : 83
goukei :100
```

リスト内の全要素を足し合わせるプログラム

```
a = [56,3,62,17,87,22,36,83,21,12]
goukei = 0
for i in range(0,10,1):
    goukei = goukei + a[i]
print(goukei)
```

```
goukei : 399
```

リスト内の全要素を足し合わせるプログラム

- こちらの方がPythonらしい書き方

```
a = [56,3,62,17,87,22,36,83,21,12]
goukei = 0
for i in a:
    goukei += i
print(goukei)
```

リストから要素を
ひとつづつ取り出す

```
goukei : 399
```

補足：辞書

- 辞書とは
  - 複数の値を一つの名前（変数名）と文字列（キー）によって管理する仕組み
    - 例: age = {'松賀':18, '北山':24}　age['松賀']:18, age['北山']:24
- 辞書を使う理由
  - 複数の値をリストのような順序ではなく、値の意味を表すキーと対応づけて管理したい
  - 特定のキーと対応づけられた値を高速に検索したい
    - 例：数万人のデータがあるときに、キー「'松賀'」の年齢を検索

乱数

- 乱数とは
  - ある一定の範囲内において，すべての数が等確率で現れるような数
  - コンピュータで生成する乱数は計算によって値を生成しているので、擬似乱数とよばれる
  - 乱数を用いることで実行するたびに処理内容が変わるようなことが実現できるので、シミュレーションなどに応用できる
- 乱数を用いたプログラムの例
  - 0から9までの乱数を発生させて数当てゲーム
  - 0から9までの乱数を1から10までに変更するプログラム

0から9までの乱数を発生させて数当てゲーム

```
import random #random モジュールを読み込む
a = 5
r = random.randrange(10) #0 ～ 9 までの整数をランダムに発生
if a == r:
    print("当たり")
elif a > r:
    print("aの方が大きい")
elif a < r:
    print("aの方が小さい")
```

0から9までの乱数を1から10までに変更するプログラム

```
import random
r = random.randrange(10) + 1
print(r)
```

関数

- 関数とは
  - 引数を受け取り、何らかの処理を行って、戻り値を返すもの
  - 大規模なプログラムを開発する際に、機能毎に切り分けて小さな関数として開発を行う
  - 作成した関数は何度でも呼び出すことができ、再利用が可能になる。
- 関数で分割したプログラムの例
  - 主たる流れと、リストの値の合計を求める処理に分けてコーディングを行う

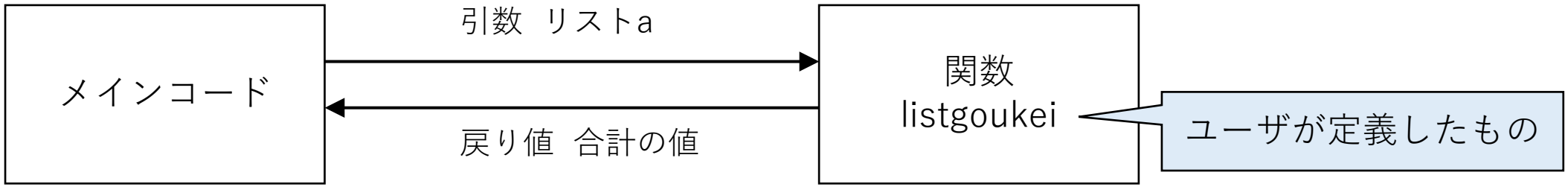
関数定義の例

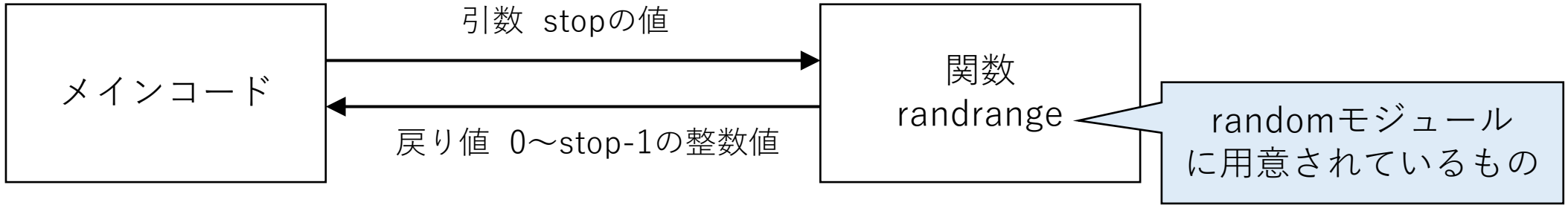
```
def listgoukei(a): # 合計を求める関数listgoukei を作成
    goukei = 0
    for i in a:
        goukei += i
    return goukei

a = [56,3,62,17,87,22,36,83,21,12]
goukei = listgoukei(a) # 作った関数listgoukei を呼び出し
print(goukei)
```

```
goukei : 399
```

関数の実行イメージ


メインコード
引数　リストa
戻り値　合計の値
関数
listgoukei
ユーザが定義したもの


メインコード
引数　stopの値
戻り値　0～stop-1の整数値
関数
randrange
randomモジュール
に用意されているもの

WebAPI

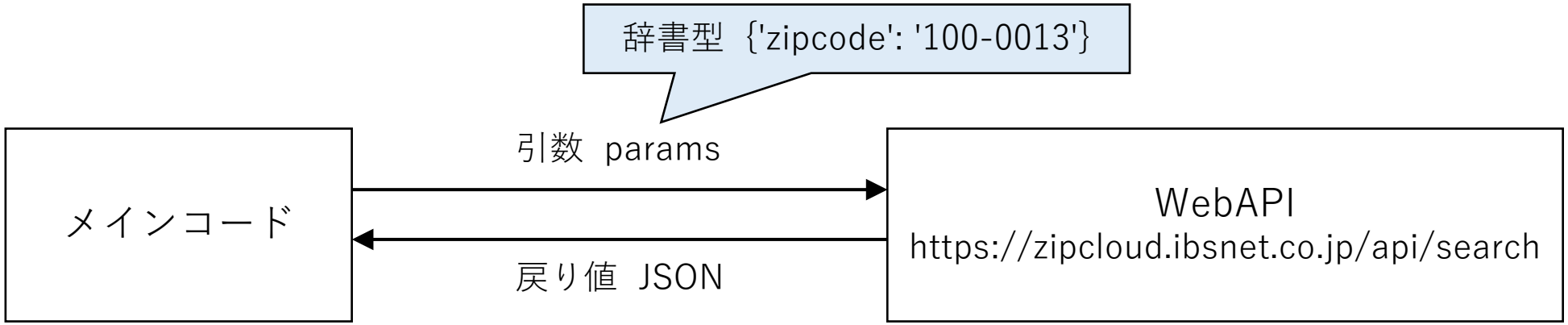
- API (Application Programming Interface) とは
  - 共通して使われる複雑な処理を関数に近い形でまとめて提供するもの
- WebAPIとは
  - インターネット上のサービスやビッグデータへのAPI
  - Webの通信方式を利用したアクセス機能を提供している
- WebAPIを利用したプログラムの例
  - 郵便番号を引数としてそれに対応する住所を得る例

WebAPIの使用例

```
import requests
import json
url = 'https://zipcloud.ibsnet.co.jp/api/search' #使用するWebAPIのURL
param = {'zipcode': '606-8001'} #WebAPIの引数を辞書型で指定
res = requests.get(url, params=param) #WebAPIの呼び出し
response = json.loads(res.text) #JSONで書かれた戻り値を辞書型に変換
address = response['results'][0]
print(address['address1'] + address['address2'] + address['address3'])
```

```
京都府京都市左京区山端柳ケ坪町
```

WebAPIの実行イメージ



https://zipcloud.ibsnet.co.jp/api/search?zipcode=100-0013


```
{
        "message": null,
        "results": [
                {
                        "address1": "東京都",
                        "address2": "千代田区",
                        "address3": "霞が関",
                        "kana1": "ﾄｳｷｮｳﾄ",
                        "kana2": "ﾁﾖﾀﾞｸ",
                        "kana3": "ｶｽﾐｶﾞｾｷ",
                        "prefcode": "13",
                        "zipcode": "1000013"
                }
        ],
        "status": 200
}
```


JSON: JavaScript Object Notation

キーと値をコロンで区切り、カンマで並べる

値はJSONを要素とするリストでもよい

WebAPIの活用

- 内閣官房 新型コロナウィルス感染症対策 各種データ
  - https://corona.go.jp/dashboard/
- 政府統計の総合窓口 e-Stat（第4章）
  - https://www.e-stat.go.jp/

学習15 アルゴリズムの比較

(1)探索アルゴリズム 線形探索と二分探索

(2)ソートアルゴリズム 選択ソートとクイックソート

(3)選択ソートとクイックソートの比較

探索アルゴリズム 線形探索と二分探索

- 探索
  - リストなどの中から必要なデータを探し出すこと
- 線形探索 https://yutaka-watanobe.github.io/star-aida/1.0/algorithms/linear_search/anim.html
  - リストを先頭から順に比較しながら探索値に一致するデータを探し出す探索方法
- 二分探索 https://yutaka-watanobe.github.io/star-aida/1.0/algorithms/binary_search/anim.html
  - リストの中から探索範囲を半分ずつ狭めながら目的のデータを探し出す探索方法
- 線形探索と二分探索の比較
  - 線形探索はデータ数が多くなると時間がかかるが、二分探索は線形探索と比較するとデータ数が多くなってもそれほど時間は増加しない
  - 二分探索は事前にデータを並べ替えておく処理が必要になる

ソートアルゴリズム 選択ソートとクイックソート

- ソート
  - リストなどの中を昇順, 降順に並べ替えること
- 選択ソート https://yutaka-watanobe.github.io/star-aida/1.0/algorithms/selection_sort/anim.html
  - リスト内のデータから最小値を探索し, 最小値から順に取り出すことで並べ替えを実現する
- クイックソート https://yutaka-watanobe.github.io/star-aida/1.0/algorithms/quick_sort/anim.html
  - リスト内の一つのデータを軸として, 大小2つに分割した後, 分割したデータに対して同じ処理を再度行うことにより並べ替えを実現する
- 選択ソートとクイックソートの比較
  - 選択ソートは最小値を探す処理を、ほぼデータ数回行う
  - クイックソートは2分割を繰り返すので、データ数が多くなっても必要な時間はそれほど増加しないが、特殊な場合は選択ソートより時間がかかる場合がある

学習16 確定モデルと確率モデル

(1)モデル化とシミュレーション

(2)確定モデルのシミュレーション

(3)確率モデルのシミュレーション

モデル化とシミュレーション

- モデルとは
  - 事物や現象の本質的な形状や法則性を抽象化して，より単純化したもの
  - 事物や現象のモデルを作ることをモデル化とよぶ
  - 表現形式によるモデルの分類
    - 縮尺モデル、数式モデル、図的モデル
  - 対象の特性によるモデルの分類
    - 静的モデル、動的モデル（確定モデル、確率モデル）
- シミュレーションとは
  - モデルを使って実際にどのような現象が起こるのかを予測すること

確定モデルのシミュレーション

- 確定モデル
  - 不規則な現象を含まず、方程式などで表せるモデル
- 確定モデルの例
  - 複利法による預金金額の時間的変化

確率モデルのシミュレーション

- 確率モデル
  - 不規則な現象を含んだモデル
- 確率モデルの例
  - サイコロの目の出方
- モンテカルロ法
  - 対象のモデルに乱数を大量に生成して入力し，近似解を得るもの
  - 例）円周率の計算

学習17 自然現象のモデル化とシミュレーション

(1)物体の放物運動のモデル化

(2)物体の放物運動のプログラムによるシミュレーション

(3)生命体の増加シミュレーション

(4)ランダムウォークのシミュレーション

物体の放物運動

- 物体の放物運動のモデル化
  - 初速度と投げ上げる角度によって，物体がどのような軌跡を描いて飛んでいくのか，遠くに飛ばすためにはどのような角度で投げるとよいのかという問いの答えを導く
  - 水平方向と鉛直方向でそれぞれ運動を表す式を立てる
- プログラムによるシミュレーション
  - 作成した数式モデルをプログラムで記述
  - 条件を変えながらシミュレーションを行う

生命体の増加シミュレーション

- 生物の個体数の変化をシミュレートする
- 環境に生息する個体数の上限：環境収容数
- 単位時間あたりに個体が増加する数：個体数×増加率
- 単位時間あたりに個体が減少する数：個体数×減少率
- 減少率＝（現在の個体数／環境収容力）× 増加率
- 個体数が制約条件の中で成長する曲線をロジスティック曲線といい，環境収容力に収束する

ランダムウォークのシミュレーション

- ランダムウォーク
  - ブラウン運動や株価の変動など，不確実な現象のシミュレーションとして用いられるモデル
  - 原点をスタートし、x軸方向、y軸方向にランダムな距離だけ移動することを繰り返す

参考文献・資料

- 渡部有隆、ニコライ・ミレンコフ:アルゴリズム ビジュアル大事典 図解でよくわかるアルゴリズムとデータ構造, マイナビ出版, 2020.
  - サポートページ: https://yutaka-watanobe.github.io/star-aida/books/