NPCA 部誌 2025

灘校パソコン研究部
npca

79th Nada School Festival
weave

目次

あいさつ

皆さんこんにちは。灘校パソコン研究部部長の Kohenyan と申します。本日は第 79 回灘校文化祭 “weave” にご来場いただきありがとうございます。灘校パソコン研究部では、競技プログラミング[*1]や機械学習、Unity[*2]などによるゲーム作成、CTF[*3]やセキュリティ、サーバー構築など、情報系の様々な分野に日々取り組んでいます。一口にパソコンと言えども多様な「好き」を持った部員たちの集大成である展示と部誌を、ぜひ楽しんで行ってください！

この冊子について

部誌のコードやファイルは、灘校パソコン研究部の公式サイトにて公開しています。少しでもご興味を持っていただけましたら、公式サイト内の「作品」ページをご覧いただくか、「NPCA 作品」で検索してください。

執筆者の表記について

執筆者名として、本名ではなくハンドルネームが記載されています。これは、部内では本名を呼ぶことが滅多になく、ほとんどの場合部内ではハンドルネームで互いを呼び合うという慣習によるものです。また、灘校の文化に従い、学年ではなく何回生かが記されています。2025 年 5 月現在、各回生は次の学年に相当します。

- 79 回生：高校 2 年生
- 80 回生：高校 1 年生
- 81 回生：中学 3 年生

[*1] 与えられた課題を解決するプログラムをいかに素早く正確に記述するかを競うプログラミングのコンテストの総称

[*2] ゲームを作るためのアプリケーション (ゲームエンジン) の一種

[*3] Capture The Flag（旗取りゲーム）の略、情報セキュリティの技術を競う競技

第 1 章

電動キックボードのポート配置を最適化してみた

79 回生　Kohenyan

1.1　はじめに

こんにちは！パソコン研究部の部長をしている、高 2 の Kohenyan と申します。部長になってみて、思った以上にやることが多くて大変だなと感じる毎日です。

先に説明しておくと、「情報科学の達人」というプロジェクトは、高校生や高専生が最先端の研究者と一緒に学びながら、自分たちで研究テーマに挑戦するというプログラムです。

今回はこの「情報科学の達人」という教育プロジェクトに参加して、その中で取り組んだ研究内容を記事にまとめました。このプロジェクトでは、東京大学の河瀬先生（特任准教授）がメンターとしてサポートしてくださいました。河瀬先生には、研究を進めるうえでたくさんアドバイスをいただき、本当に感謝しています。改めてお礼を申し上げます。

1.2　電動キックボードの研究をやろうと思ったきっかけは？

電動キックボードって少し前に結構話題になりましたよね。

それで、キックボードのポートってどれぐらいあるんだろう？ということで、学校の近く、三宮駅周辺のポートを調べてみました。

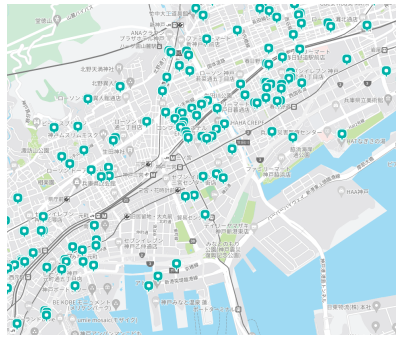


図 1.1: ポート配置の図

上の図を見て、ポートの配置偏ってるのでは？と思い、実際の需要とどれぐらいずれているか、そして再配置を行うことでどのように改善するか調べることにしました。

1.3 前提知識

電動キックボードについて： 電動キックボードはポートとポートの間を移動できる交通手段です。今回は LUUP という電動キックボードについて調べました。

線形計画問題とは：線形計画問題は、線形な目的関数を最大化または最小化し、線形な制約条件を満たす解を求める問題です。変数は連続値（実数）を取ることができ、特定の範囲内で (目的関数が最大化 or 最小化するような) 最適な解を探します。

簡単な例) 目的関数: $-2x + 3y$ ←最小化

制約条件: $x + y \leq 5,\ x, y \geq 0$

このような問題を、ソルバー PuLP というものを使えば最小化される (x, y) を求めることができます。実際は、この x や y の変数が数十個、数百個もある問題を解きます。

1.4 実験の手法

神戸の三宮駅、神戸駅周辺をグリッド状に分割することで、プログラム上にリアルの情報を表現しました。また、ポート配置を最適化する問題を、のちのページのような線形計画問題として定式化し、ソルバー PuLP によって求解しました。

ポートを利用する人口は、特定の時間帯における人の分布を示す「人口流動マップ」（NTT ドコモ提供）を活用して推計しました。具体的には、平日 13 時の $500m$ グリッドごとの人口データに対し、電動キックボードを利用する割合（$= 0.01$ 程度の定数）を乗じることで、各グリッドから駅へと移動する利用者数、つまり需要を算出しました。

(問題設定:2 つの駅があり、各利用者は最寄りの駅を利用します。キックボードを利用すると移動速度が徒歩の 2 倍となり、利用者全体の平均移動時間が最小となるように利用されるものとします。)

線形計画問題の定式化

注釈: パラメータ、関数、変数、目的関数、制約条件をのせますが、少し難易度高めなので、分からない場合は実験結果まで飛ばしてください。

パラメータ

- M: グリッド数
- pop_i: i 番目のグリッドの人口 $(0 \leq i < M)$
- $pos_{i,j}$: i 番目のグリッドの座標 ($i = M$ は駅 A, $i = M + 1$ は駅 B, $j = 0$ が x 座標, $j = 1$ が y 座標)
- $portprev_i$: i 番目のグリッドの現在のポート数 $(0 \leq i < M)$

関数

距離関数：

$$d(i,j) = \sqrt{(pos_{i,0} - pos_{j,0})^2 + (pos_{i,1} - pos_{j,1})^2}$$

変数

- $port_i$: i 番目のグリッドに設置するポート数 $(0 \leq i < M)$
- $v_{i,j}$: i 番目のグリッドが利用する移動手段の人数 $(0 \leq i < M,\ 0 \leq j \leq M,\ j = M$ の場合は徒歩移動)

目的関数

$$\min \left\{ \begin{array}{l} \displaystyle\sum_{i=0}^{M} \sum_{j=0}^{M-1} \Big(\underbrace{d(i,j) \times 2}_{\text{徒歩でグリッドからポート}} + \underbrace{\min\big(d(j,M), d(j,M+1)\big)}_{\text{キックボードでポートから最寄駅}} \Big) \times v_{i,j} \\ + \displaystyle\sum_{i=0}^{M-1} \Big(\underbrace{\min\big(d(i,M), d(i,M+1)\big) \times 2}_{\text{徒歩でグリッドから最寄駅}} \Big) \times v_{i,M} \end{array} \right\}$$

制約条件

1. ポート容量制約：
$$\sum_{j=0}^{M-1} v_{j,i} \leq port_i \quad (0 \leq i < M)$$
2. 人口分配制約：
$$\sum_{j=0}^{M} v_{i,j} = pop_i \quad (0 \leq i < M)$$
3. ポート増設上限 (X はポートを追加する数)：
$$\sum_{i=0}^{M-1} (port_i - portprev_i) \leq X$$
4. ポート非減少制約 (再配置実験の時は除く)：
$$port_i \geq portprev_i \quad (0 \leq i < M)$$
5. 非負制約：
$$v_{i,j} \geq 0,\ port_i \geq 0 \quad (0 \leq i < M, 0 \leq j \leq M)$$

1.5 行う実験

まず、実際のポートの配置と需要がどれぐらいマッチしているか調べるために、ポートの総数を変えずに (実際のポートの配置をすべて破棄した状態で) 再配置する実験を行いました。次に、ポートを追加したらど

れぐらい改善するのか確認するために、実際のポートの配置を維持したままポート数を増やしていく実験を行いました。

1.6 実験結果

1. ポートの総数を変えずに再配置した場合の改善効果: 平均移動時間の値が、再配置前では 6.9067 であったのに対して、再配置後は 4.5964 となり、再配置により 34% の改善が確認できました。
2. ポートを追加した場合の改善効果: 下図のように、ポートの追加に応じて平均移動時間が改善しました。

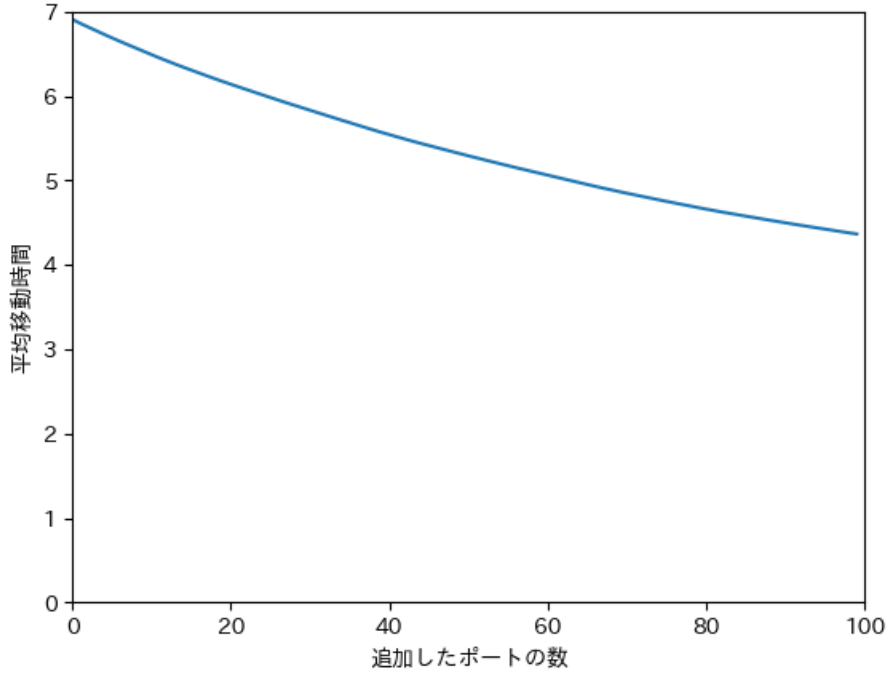


図 1.2: ポート追加による移動時間の変化

実験結果から分かること: しかし、元々のポートは 114 個であり、約 90 個の追加によって再配置と同等の改善効果が得られたことが分かります。従って、単純にポートを増設するよりも、適切に再配置する方が効率的な改善策であることが分かりました。

1.7 まとめ

実験結果に書かれている通り、ポートを増設するよりも適切に再配置する方が効率的であるというのは、言い換えると、既存のポートが需要とかなり乖離された状態で配置されている、ということですね。今回は、ポート数が 100 個程度、駅も二つだけの神戸、三ノ宮駅周辺で実験しましたが、ポート数ももっと多く、駅も密集している東京 23 区全体で実験してみたり、ポート配置による設置費用と、設置すると需要によってどれだけ利益が増えるか、も実験してみたいと思いました。

1.8 参考資料

- `https://note.com/shirotabistudy/n/n7fc4ba351eea` 「Python pulp メモ」

- https://luup.sc/port-map/ 「LUUP ポート一覧」
- https://mobakumap.jp/#34.688958,135.187581,14z 「人口流動統計 - モバイル空間統計」

第 2 章

ポケポケの戦略をプログラミングで導く

79 回生 かつどん

2.1　はじめに

こんにちは。79 回生のかつどんです。好きなポケモンはラティアスです。僕の灘校文化祭も残り 2 回となってしまいました。去年はなんと準備期間から文化祭が終わるまで熱を出して寝ていました。最後の文化祭でそうならないことを祈るばかりです。

さて皆さん、Pokémon Trading Card Game Pocket　通称「ポケポケ」　プレイされているでしょうか？ポケポケはポケモンカードゲームを原作としたスマホ向けのカードゲームアプリで、手軽にカードを集めたり対戦したりでき、去年の 10 月にリリースされて以来大人気のゲームになっています。この記事ではそんなポケポケの「対戦」について深く考えてみます。

皆さんポケポケをプレイしていてこんなことを思ったことはありませんか？

- 進化ポケモンを立てたいけどなかなか手札に来ない。
- 相手のカスミで運だけで負けた。
- 手札のカードどれから使ったら一番勝ちに近づくの？

やはりカードゲームなので当然確率が絡みますが、なかなか釈然としないことがあるかもしれません。今回は「進化ポケモンが立つ確率」に焦点を当てて確率をプログラミングで求め、戦略を考えてみます。この記事を皆さんが読んでいるころにはランクマッチが開幕していると思うので、これを参考にプログラミングを駆使して爆勝ちしてみてください。

2.2　ポケポケをプレイしていない人へ

ここではポケポケの対戦機能について軽く説明します。既にプレイされている方は飛ばして大丈夫です。

ポケポケでは 20 枚のカードを使って 1 対 1 で対戦します。カードには大きく分けて「ポケモン」と「トレーナーズ」の 2 種類のカードがあります。対戦の最初に先攻・後攻を決めて山札を 5 枚引きます。それぞれの番のはじめに山札からカードを 1 枚引き、手札のカードを使って戦います。

ポケモンのカードは、実際にワザを使って戦うカードです。

ポケモンは「バトル場」と「ベンチ」という 2 種類の場所に出すことができ、自分のターンにバトル場にいるポケモンがワザを使って相手のポケモンに攻撃することができます。ワザを使うと自分の番が終わりま

す。ワザなどでダメージを受けて HP が 0 になるとそのポケモンはきぜつします。相手の番にバトル場のポケモンが相手に倒されたときはベンチからポケモンを出します。ポケモンは進化していないポケモンであるたねポケモンとその進化先のポケモンをデッキに入れておくことで進化させることもできます。ポケモンの中にも「ポケモン ex」という特別なポケモンがいて、相手のポケモン ex を倒すと 2 ポイント、そうでないポケモンを倒すと 1 ポイント獲得でき、3 ポイント取るとバトルに勝利となります。

トレーナーズのカードは戦うのではなく、自分のターン中に使用して山札を引いたりポケモンの HP を回復したり対戦をサポートしてくれるカードです。

トレーナーズのカードには「グッズ」「サポート」「どうぐ」の 3 種類があります。グッズは自分のターンに何回も使用できるカードです。サポートは自分のターンに 1 回しか使えません。どうぐは他とは違い、自分の場のポケモンに付けて効果を発揮します。

正直プレイしてみないとわからない部分が多いと思うのでこの記事を読んで興味を持ったら実際にプレイしてみてください。

2.3　今回使うデッキや手法など

今回は執筆時点で環境デッキの一角である「ダークライ ex+ ジバコイルデッキ」を用いて「最速で 2 進化ポケモンのジバコイルが立つ確率」を考えていきます。その番場に出したばかりのたねポケモンやその番進化させたばかりのポケモンを進化させることはできないため、2 進化が立つ最も早いターンは 3 ターン目です。前半では 1 つめのデッキ、後半では 2 つめのデッキについて考えます。どちらも海外大会の上位入賞デッキで強さはお墨付きです。

採用されているカードについて軽く説明します。

- コイル・レアコイル・ジバコイル　今回計算するテーマのポケモンです。自分で自分にエネルギーを付けられて優秀なアタッカーです。コイル→レアコイル→ジバコイルと進化します。
- ダークライ・ガルーラ　どちらもジバコイルと相性が良いたねポケモンです。
- モンスターボール　山札からランダムにたねポケモンを 1 枚手札に加えるグッズです。
- 博士の研究　山札を 2 枚引けるサポートです。
- ポケモン通信　2 つめのデッキにのみ採用されています。手札のポケモンを 1 枚選んで山札のランダムなポケモンといれかえるグッズです。

- スピーダー　1 つめのデッキにのみ採用されています。使用したターンポケモンのにげるために必要なエネルギーが 1 減るグッズです。
- リーフ　使用したターンポケモンのにげるために必要なエネルギーが 2 減るサポートです。
- ナツメ・アカギ　どちらも相手のベンチのポケモンをバトル場に引っ張り出すサポートです。
- ゴツゴツメット　持たせたポケモンがワザのダメージを受けたとき相手に 20 ダメージ与えるどうぐです。
- 大きなマント　持たせたポケモンの HP が 20 増えるどうぐです。

スピーダー以降のカードは今回調べる 2 進化が立つ確率には関係がないカードですが、どれも強力なカードです。

ところでみなさん、高校で確率を勉強するときどうやって求めていたでしょうか？全事象のうち○○が起こる場合が何通りあるから何分の何、という風に求めていたりしたと思うのですが、2 進化が立つ確率なんていうのは全事象が多すぎて到底正確に求めることはできません。そこで、コンピュータの力を借りて「実際に何万回と試してみて何回成功したか」で確率を求めていきます。プログラミングの世界ではモンテカルロ法とか言われています。2,3 回試してみただけでは正確な確率は求められませんが、プログラミングを用いて大量に試行することでかなり正確に求めることができます。

2.4　実際にプログラムを書いて計算してみよう！

最初は 1 つめのポケモン通信が入ってないデッキで 3 ターン目にジバコイルが立つ確率を計算していきます。ちなみに今回、レッドカードなどのカードで相手からこちらの手札に干渉してくるカードについては考えないものとします。そんなもの考えてられないので。最初は順番を特に考えずにカードを使っていきます。具体的には全て博士の研究→モンスターボールの順番でプレイしています。後でカードを使う順番をよく考えると確率がどのように変わるかお見せしようと思います。

今回は僕が普段使い慣れている C++ を使って計算していきます。紙面上ではスペースを使いすぎてしまうためコードは紙版では割愛します。web 版では実際に使ったコードを載せているのでそちらも是非お読みください。ちなみに実験のために人に見せることとか考えずに適当に書き殴ってたら 400 行とかになってました。ひどすぎる。web に載せるコードはそういうきったねえコードじゃなくてきれいにする予定です。

```
56      if(hand['G'] >= 1){
57        for(int i = 0; i < 2; i++){
58          hand[deck[0].first]++;
59          deck.erase(deck.begin());
60        }
61        hand['G']--;
62      }
63      while(hand['F'] > 0){
64        bool flag = true;
65        for(auto x : deck){
66          if(x.first == 'A' || x.first == 'D' || x.first == 'E'){
67            flag = false;
68            break;
69          }
70        }
71        if(flag) break;
72        while(deck[0].first != 'A' && deck[0].first != 'D' && deck[0].first != 'E'){
73          random_device get_rand_dev;
74          mt19937 get_rand_mt(get_rand_dev());
75          shuffle(deck.begin(), deck.end(), get_rand_mt);
76        }
77        hand[deck[0].first]++;
78        deck.erase(deck.begin());
79        hand['F']--;
```

こんな感じです。ちなみにこれは博士の研究とモンスターボールを使用するときのコード。

実際にカードを引いたりポケモンを持ってきたりのシミュレーションを 100000 回行うコードを書いて実行してみると...36505 回成功、だいたい 36.5 %の確率で 3 ターン目にジバコイルが立つことがわかりました！だいたい 3 回に 1 回といったところです。みなさんがプレイする時の感覚とは合っているでしょうか？僕的には少し感覚より低いかも？という感じです。というわけでここからはポケモン通信を入れたり順番を変えたりして確率がどれくらいあがるか調べていきましょう！

2.5　ポケモン通信を入れて計算してみる

次はポケモン通信を入れたデッキで計算してみます。順番は深く考えずに博士→モンスターボール→ポケモン通信でプレイします。ちなみにポケモン通信は手札に 2 枚来たポケモンがあるときにだけ使用しています。さっきと同じように 100000 回シミュレーションしてみると...？

結果は 43004 回成功、約 43 %で 3 ターン目にジバコイルが立つという結果になりました。なんとカードを 1 枚変えるだけで 6.5 %も確率が上がるのです。ポケモン通信というカードの強さがお分かりいただけたでしょうか？普段なんとなくデッキに入れているカードでもこうやって実際に大量に実験してみるとその強さが数値としてわかるのです。(ちなみにポケモン通信の代わりにデッキから抜けたスピーダーも強力なので 2 進化が立つ確率が上がったからといって一概にデッキが強くなったとは言えないことに注意してください (笑))

2.6　順番を考えて計算してみる

最後はカードを使う順番を変えてみます。こんな順番にします。

コイルがいるとき

以下の 2 点において変えています。

まずは博士の研究とモンスターボールの使用順です。1 ターン目にまだコイルを持っていないときは博士→ボール、それ以外の場合はボール→博士→（博士でボールを引いたら）ボール　という順番にします。なぜこうするのかというと、コイルが欲しい場合はコイルを引く確率を少しでも上げたいため、先に 2 枚引いてそこで他のたねポケモンを引けたら後でコイルを持ってこれる確率を上がるので先に博士を使う方がよく、逆にコイルを引いている場合は 2 進化が立つ確率を上げるためにたねポケモンはいらないため先にモンスターボールを使用してたねポケモンを山札から抜くことで欲しいカードを引ける確率を上げた方がいい、ということです。

2 つ目はポケモン通信についてです。ここまでは手札にあったら番の終わりに必ず使用していましたが、そのターン欲しいポケモン（1 ターン目であればコイル、2 ターン目ならレアコイル、3 ターン目ならジバコイル）を引いていないときに限り番の終わりに使用するように変えます。まだ引けていない進化先を引く確率を上げるためにはターンが過ぎてたくさん山札を引いてからポケモン通信を使った方が良いからです。

この 2 点について変更しまた 100000 回シミュレーションすると... 結果は 44112 回成功、約 44.1 %で 3 ターン目にジバコイルが立つという結果になりました。先ほどのような大きな変化はありませんがそれでも確率は上昇しました。ポケポケは運要素の強いゲームではありますが、こうしたデッキ構築やカードの使用順によって勝つ確率を上げることができるのです。

2.7　おわりに

いかがでしたでしょうか？プログラミングの便利さとポケポケの奥深さを理解していただけたなら幸いです。

ポケポケには他にもいろいろ確率を計算してみたいことがあると思うので、そんなときにこの記事を参考にシミュレーションして、どんな風にデッキを組めば安定性が高くて強いデッキが組めるか、どうプレイすれば勝てるようになるか、いろんなデッキについて計算してみてください。もちろん、ポケポケ以外のことでも計算してみてくださいね。ちなみに僕は紙のポケモンカードもやっているのですが、60 枚のデッキでこれをしようとして挫折しました (笑)。

ここまでお読みいただきありがとうございました。引き続き灘校文化祭「weave」をお楽しみください！

第 3 章

日本語で書かれる PG 言語　なでしこ

81 回生 fujimon

3.1　はじめに

はじめまして、fujimon と申します。今回は日本語で書かれるプログラミング言語を見つけたのでどんなものか、そして構文がどんなものかなどを見ていこうと思います。

3.2　なでしことは

「なでしこ」は山本 峰章さんが日本語で誰でも気軽にプログラムできるようにしたいという思いでつくった言語です。特徴としては日本語で書かれている、が大きなものだと思います。ほかの言語（c++,Python,javascript など）は大体英語で書かれます。プログラミング言語は世界中で使われるので英語で書かれるほうが良いです。ちなみにほかの日本語で書かれるプログラミング言語は「ひまわり」や「ドリトル」、「プロデル」など 7 種類以上あるみたいです。

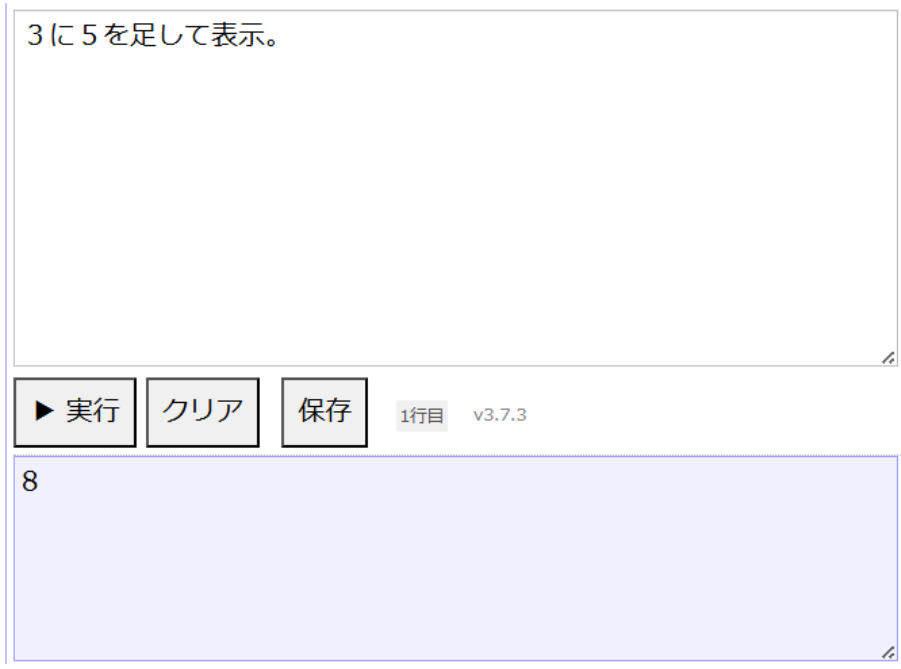
3.3　「なでしこ」の構文など

それでは「なでしこ」の構文などを見ていきましょう。「なでしこ」の構文は公式 HP に載っていました。

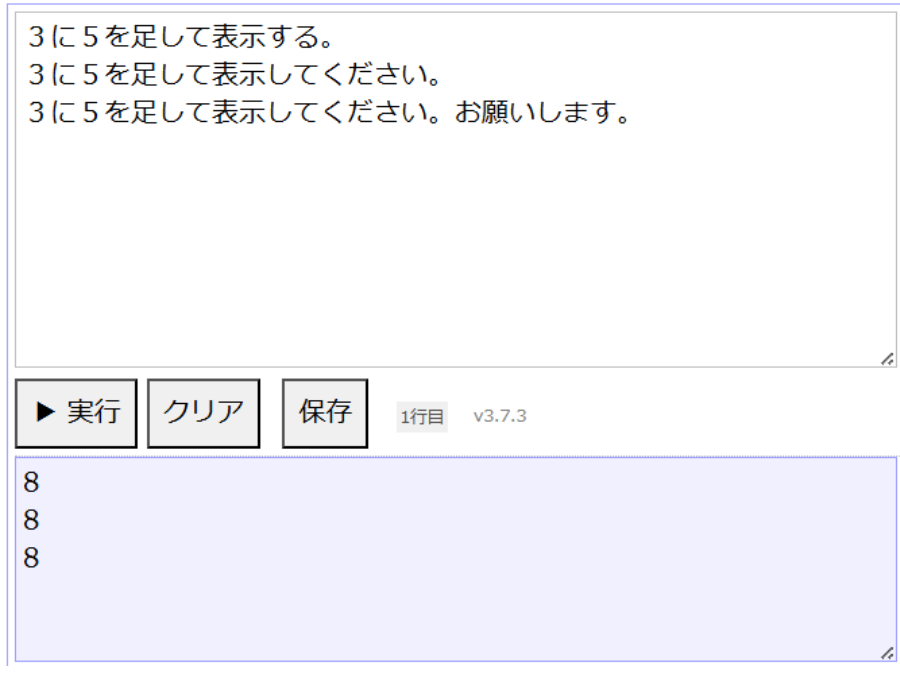
→→→　`https://nadesi.com/`

まず、出力は　〜〜を表示。　とするとできるみたいです。

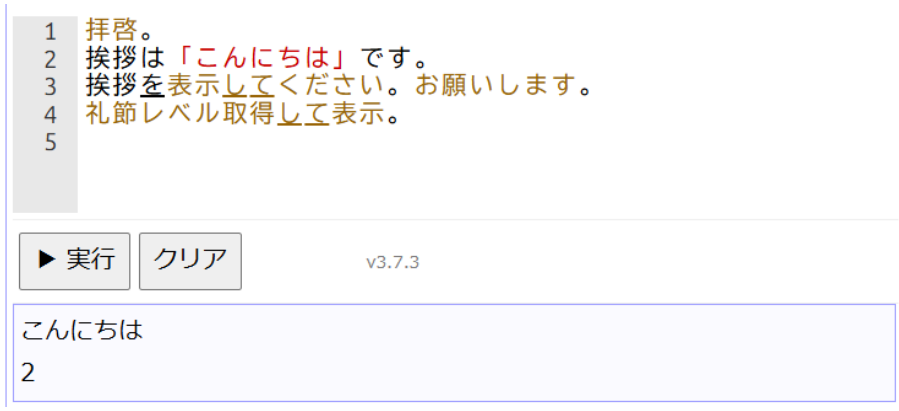
次に四則演算の仕方を見てみましょう。四則演算は例えば３＋５なら　３に５を足して表示。



　とするとできるみたいです。
また、敬語の機能も実装されており、さっきのコードを敬語化して
３に５を足して表示する。
３に５を足して表示してください。
３に５を足して表示してください。お願いします。



　のようにしても３＋５を実行できます。
さらに拝啓、敬具とかいて手紙のようにすることもできるみたいです。ちなみに「敬具」には礼節ポイントを初期化（０にすること）して、「ください」、「お願いします」などの数をカウントし、礼節ポイントを表示させることもできるみたいです。~~謎使用ですね。~~



上では１行目で礼節ポイントを０にし、３行目に「ください」、「お願いします」をつかっているので礼節ポイントが２になっています。

3.4 問題を解いてみる

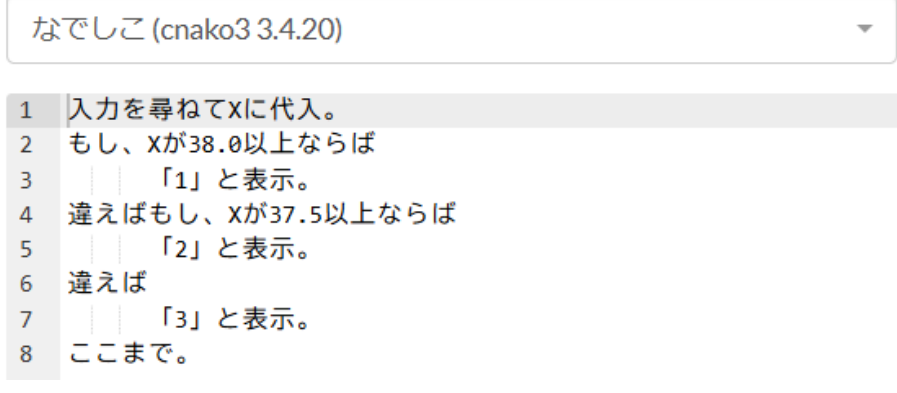
それでは問題を解いてみましょう。
今回は AtCoderBeginnerContest397 の A 問題を解きます。
`https://atcoder.jp/contests/abc397/tasks/abc397_a`

問題文
高橋君は自分の体温を計測し、 X ℃ であることがわかりました。
体温は次の 3 種類に分類されます。

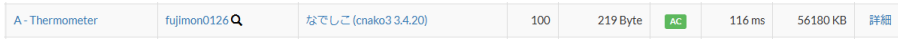
- 38.0 ℃ 以上のとき「高熱」
- 37.5 ℃ 以上 38.0 ℃未満のとき「発熱」
- 37.5 ℃未満のとき「平熱」

高橋君の体温は何に分類されるでしょうか？「出力」の項にしたがって整数で出力してください。
※高熱の時１，発熱の時２、平熱の時３を出力

これはただ単に X に応じて場合分けするだけですね。



解説をしておくと、1行目ではXという変数をつくり、ここに与えられた数字を代入しています。2, 3行目ではXが38.0以上の時の場合分けです。4, 5行目ではXが37.5以上38.0未満の時、5, 6行目ではXが37.5未満の時の場合分けです。ちなみに最後の行の「ここまで」はよくわかりません。ほかの人も書いていてなかったらエラーがおきました。まあ普通にコードの最後を意味しているのでしょう。
提出してACだと正解、WAだと不正解、CEだとなにかエラーが起きています。



ということで正解でした。

3.5 さいごに

初めて部誌を書くものでクオリティとか低いかもですがご了承ください。
来年はバイナリコードとプログラミング言語について書きたいなぁと思います。

第 4 章

MythicMobs でオリジナルモブを作ろう

81 回生 ぱんだ

4.1 はじめに

皆さん、こんにちは。81 回生のぱんだです。今回は、Minecraft でオリジナルのモブ（生物）を作ります。よかったら最後まで読んでいってください。

前提条件

Minecraft の正規版を持っている/使える

Visual Studio Code(文中では VSCode と表記します) を使用できる

Java が PC にインストールされていて、バージョンが 17 または 21 である

環境

Windows 11

Minecraft 1.20.4

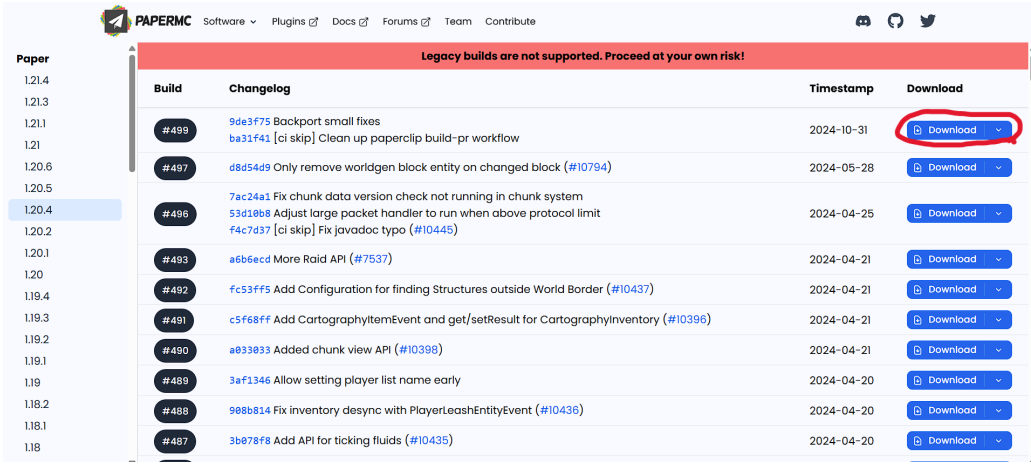
Java 21

PaperMC 1.20.4-499

4.2 サーバーの作成

※ここで立てるサーバーは自分および自分が繋いでいるネットワークに接続している人のみ入ることが出来るものになります。外部の人とのプレイはできません。

まず、`https://papermc.io/downloads/all` にアクセスします。左側には Minecraft の今までのバージョンが羅列されているので、1.20.4 を選択して#499 と書いてある行の Download を押しファイルをダウンロードしてください。



次に、任意の場所（自分がわかりやすい所が良いです　デスクトップ等）に「paper1.20.4」など分かりやすい名前のフォルダを作成し、そのフォルダの中にダウンロードした paper-1.20.4-499.jar を入れてください。

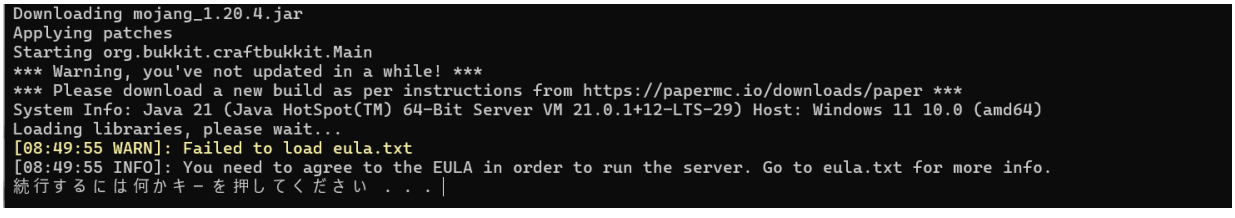
そして、フォルダ内に新規テキストファイル（名前は後で変えるので何でもいいです）を作成し、そこに以下のコードを記述してください。

```
@ECHO OFF
java -Xms512M -Xmx4096M -jar paper-1.20.4-499.jar nogui
PAUSE
```

これを保存した後に、ファイル名と拡張子を「起動.bat」に変更してください。

起動.bat を起動するとコマンドプロンプトが起動すると思われるので、少し待ちます。

待つと、以下のような画面になるはずです。



任意のキーを押してこのウィンドウを閉じ、先ほどのフォルダ内に追加された eula.txt を開いてください。この EULA というものは、End(er)-User License Agreement の略であり、ざっくり言うと「Minecraft やサーバー等のプログラムを使って利益を得ないでね」ということが記されています。(`https://www.minecraft.net/ja-jp/terms/r3` MINECRAFT 利用規約および エンド ユーザー使用許諾契約書)

これを読んで理解し、同意してから次の手順に進んでください。

eula.txt 内のおそらく 4 行目に`eula=false`と書いてあるので、ここを`eula=true`と書き換えてください。

eula.txt を保存しもう一度起動.jar を起動します。しばらく待つと一番下に Timing Reset と表示されると思います。

表示されない場合は、Java の ver を確認する・起動.bat を確認する・eula.txt を確認するなどを試してみてください。

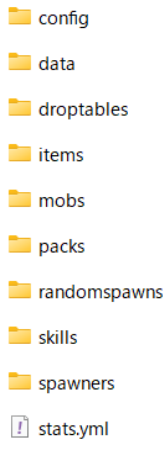
このサーバーに入りたい場合は、Minecraft を起動し、マルチプレイ→ダイレクト接続を押しアドレス欄に`localhost`と入力して接続すると入ることができます。

プラグインを導入するため、一度さっきのコマンドプロンプトで `stop` と実行してサーバーを閉じてください。これでサーバーの準備は完了です。

4.3 プラグインの導入

では、サーバーに MythicMobs を導入しましょう。

`https://mythiccraft.io/index.php?pages/official-mythicmobs-download/` にアクセスし、一番左の「Free version」をダウンロードします。次にサーバーの plugins フォルダにダウンロードした.jar ファイルを入れます。（執筆時 (2025/3/15) での最新 ver は 5.8.1 なので、実行する ver によっては少し変わる場合があります。）ここで一度サーバーを起動して、plugins フォルダ内に「MythicMobs」というフォルダができたことを確認してください。これで MythicMobs の導入は完了です。



MythicMobs フォルダ内はこのようになっているはず

4.4 オリジナルモブのコードを書く

MythicMobs フォルダ内の mobs フォルダを開き、test.yml というファイルを新しく作成します。このファイルを VSCode で開いてコードを書いていきます。（簡易テンプレートが最後にあるので是非使ってください）

ここでは以下の例を使用してコードの解説をしていきます。

コードの例

```
test:
Type: zombie
Display: '&d テスター君'
Health: 100
Armor: 0
Damage: 10
BossBar:
Enabled: true
Title: '&d テスター君'
Range: 10
Color: ffffff
```

```
Style: SOLID
Options:
AlwaysShowName: true
Despawn: false
NoDamageTicks: 5
Equipment:
- diamond_helmet HEAD
- elytra CHEST
Skills:
- skill{s=testskill} @target ~onAttack
Drops:
- testdrops
Killmessages:
- '<target.name> はテスト中に事故に遭った'
```

一行ずつ説明していきます。

`test:` Mob の id を自由に決められます。これを使ってゲーム内で Mob を召喚できます。この場合 ID は `test` になります。

`Type: zombie` Minecraft 内でのバニラ Mob の id を記述してください。この場合ではゾンビ（zombie）を選択しています。

`Display: '&d テスター君'` 表示される名前の設定ができます。 '' 内に名前を記述してください。&d はカラーコード短縮形です。

(`https://x.gd/BgbU5` 詳しくはこちらを参照してください Minecraft Wiki-装飾コード)

`Health:100` 体力の設定です。 この場合は 100 に設定しています。

`Armor:0` その Mob が受けたダメージをどれくらい軽減するかの設定をします。

実際に受けたダメージ-Armor 値=受けたダメージとなります。今回は 0 なので 10 ダメージ受けた場合 10-0=10 でそのままの 10 ダメージを実際に食らうことになります。

`Damage:10` その Mob が他に与えるダメージの設定です。攻撃力のことです。

`BossBar:` ボスバー（画面上部に表示される線の形のゲージのこと）の表示設定です。残りの体力を可視化できるので便利です。

`Enabled: true` true/false で有効/無効の切り替えができます。

`Title: '&d テスター君'` ボスバーの表示名を自由に決められます。名前と同じにするとわかりやすいです。

`Range: 10` ボスバーをどの範囲に表示するかを決められます。Mob からの半径で指定します。

`Color: ffffff` ボスバーの色を決められます。カラーコードで記述してください。

`Style: SOLID` ボスバーのスタイルを決められます。大体 SOLID でいいです。

ボスバーが有効、表示名が「テスター君」となっていることが以下の画像のように確認できます。色もしっかり ffffff=白になっています。

`Options:` 各種オプションの設定です。

`AlwaysShowName: true` 常に名前を表示するオプションが設定できます。true だと Display で設定した名前が Mob の上に常に表示されるようになります。false だと Mob にカーソルを合わせた時のみ名前が Mob の上に表示されるようになります。

`Despawn: false` デスポーン=時間経過で Mob が消える機能の on/off を true/false で切り替えられます。

`NoDamageTicks: 5` 無敵時間（一度ダメージを受けてから次にダメージを受けられるようになるまでに時間のこと）の設定ができます。1 秒=20tick（1tick=0.05 秒）です。

`Equipment:` 装備の設定をすることができます。

`- diamond_helmet HEAD` HEAD,CHEST,LEGS,FEET,HAND,OFFHAND のどれかを指定することで、指定した場所にアイテムを持たせる/装備させられます HEAD: 頭 CHEST: 胴 LEGS: 脚 FEET: 足 HAND: 利き手 OFFHAND: 利き手と逆の手 です。利き手のデフォルトは右手です。

`- elytra CHEST`

この場合「ダイヤモンドのヘルメット」を「頭」に、「エリトラ」を「胴」に装備するように設定しているので、以下の画像のような見た目になります。

また、MythicMobs で作ったアイテムも装備させることが出来ます。（アイテムの作り方はスペースがないので割愛します）3D のテクスチャをつけたアイテムを頭に装備させてみました。（以下の画像）

`Skills:`

`- skill{s=testskill} @target` ~onAttack　testskill という名前のスキルを~onAttack(攻撃したとき) に@target(ターゲット) に向かって使う　という意味です。

Testskill には「攻撃した相手のチャット欄に preparing summon. . . と表示する」というスキルを設定しているため、以下の画像のようになります。(攻撃頻度が高いため、一秒に二回ほどの割合でこれがチャットに流れます。)

(攻撃されたときのチャット欄)

`Drops:`

`- testdrops`　別でドロップテーブルという Mob が何を報酬としてドロップするかを決めるものを作り、それを指定しています。

`Killmessages:  - '<target.name> はテスト中に事故に遭った'`　Mob がプレイヤーをキルした時のメッセージの設定です。<target.name>にはキルされたプレイヤーの名前が入ります。

この場合、名前を`tuna334`だとすると、「tuna334 はテスト中に事故に遭った」と表示されます。

4.5　終わりに

このほかにも様々なオプションが存在しています。オプションを適切に設定することで自分だけのオリジナルモブが作れます。試しに作ってみた Mob(コードの例とは別のものです) との戦闘動画が下にありますので、よろしければご覧ください。

`https://youtu.be/Iot5pmpkppY`

参考にしたサイト

アジ鯖 Adminresources-MythicMobs

`https://adminresources.azisaba.net/plugins/MythicMobs/`

Mythic Mobs Wiki-home

`https://git.mythiccraft.io/mythiccraft/MythicMobs/-/wikis/home`

簡易テンプレート

ほぼ必須のものしか載せていませんので、参考にしたサイトなど見て自由に作ってください。

```
[]
Type: []
Display: '[]'
Health: []
Armor: []
Damage: []
```

第 5 章

Python を使った自作 SNS の開発

81 回生 mikan0131

5.1　はじめに

　こんにちは！ 81 回生（今中 3）の mikan0131 です。僕は競プロを中心に活動していますが、最近 Web アプリケーションに興味を持ったので、 その一つの成果として、Python で TweetApp という SNS を作りました。最後まで読んでくれると嬉しいです。

5.2　そもそも Web アプリケーションってなに？

　僕の作ったアプリを紹介する前に、Web アプリケーションがただの Web サイトと何が違うのか説明します。
HP などのウェブサイトは、静的な Web サイトと呼ばれており、閲覧者が Web サーバーに対して送ったリクエスト（例えば「https://npca.jp/のページが見たい！」など）に対して一定のレスポンス（Web サイトを表示するための情報）しか送りません。
一方、Web アプリケーションなどの動的なウェブサイトは、閲覧者が送ったリクエストに対して、Web サーバー内のデータベースから、情報を参照して、一人ひとりに違うレスポンスを返しています。（図 1 ）

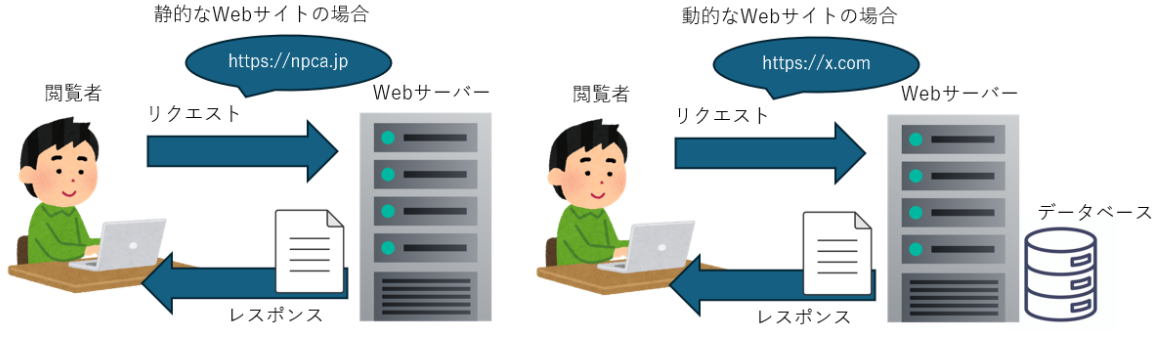


図 5.1: 静的な Web サイトと動的な Web サイトとの違い

このように、静的な Web サイトと動的な Web サイトではかなり違いがあります。 今回僕が開発したのは、そのなかでも動的な Web サイトです。題名にもある通り、Python というプログラミング言語を使って、SNS のようなものを作りました。では、今から具体的な開発について紹介します。

5.3 事前準備

5.3.1 ファイル構造

今回は、Python のモジュール（関数がいっぱい入った道具箱のようなもの）の一つである Flask を使って Web サイトを作成します。Flask では、アプリをすべて管理する「app.py」というファイルと、Web サイトの見た目を格納する「templates」というディレクトリに HTML ファイルを入れてアプリを構築します。

さらに、CSS・JavaScript・画像などのファイルは「static」というディレクトリの中に格納しておきます。
よって、ファイル構成としては図 2 のようになります。
この構成で今回は Web サイトを作ります。

図 5.2: ファイル構成

5.3.2 データベース

今回の Web サイト作成にあたって、もう一つ重要なものがあります。それは、データベースです！
つくるサイトは動的な Web サイトなので、データベースが必要だということは「2. そもそも Web アプリケーションって何？」で説明しました。ということで、データベースを用意します。今回は MySQL というデータベースを Linux 系の Ubuntu で動かすことにします。さらに、データベースを簡単に操作するため、phpMyAdmin というツールを用意しました。これで、コマンドを使うことなくデータベースを視覚的に操作できます。
これで事前準備はひとまず終了です。では早速、コードを書いていきましょう！

5.4 見た目を作る

まずは、app.py で基本的なページを表示する方法を説明します。まず app.py をソースコード 1 のように記述し、templates のなかに index.html を作成し、ソースコード 2 のように記述します。このコードでは、ソースコード 1 の app.py が templates ディレクトリの index.html を呼び出して、url が「/index」となるときに表示します。ちなみに、このコードを書いてから Web アプリケーションを起動して index にアクセスすると、図 3 のようになっています。さらに、CSS も付け加え、HTML もさらに複雑にしていくと、図 4 のようになります。

ソースコード 5.1: app.py

```
1 from Flask import flask, render_template
2 app = Flask(__name__)
3
4 @app.route('/index')
5 def index():
6     return render_template('index.html')
```

ソースコード 5.2: index.html

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset = 'utf-8'>
        <title>Flask-App</title>
    </head>
    <body>
        <h1>Hello World!</h1>
    </body>
</html>
```

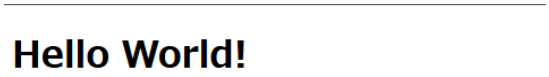


図 5.3: 結果

図 5.4: TweetApp の最初のページ

このようにして、HTML, CSS を書いていき、ユーザーのログイン・新規登録・ユーザー検索・プロファイルページなど、様々なページを作成します。これで、Web サイトの見た目を作り、表示することができました！

それでは、次にシステムの根幹となる app.py を記述して、TweetApp の機能の一つであるログイン機能を作っていきます。

5.5 ログイン機能を作る

5.5.1 データベースに接続する

「3. 事前準備」でデータベースを作ったことを覚えているでしょうか？

この機能を実装するには、ユーザーの名前やそのパスワードなどの情報を管理するデータベースが欠かせません。そこで、app.py で MySQL を使えるように設定します。app.py にソースコード 3 のように記述します。

ソースコード 5.3: app.py

```
from flask_sqlalchemy import SQLAlchemy
user = 'root'
host = 'localhost'
database = 'tweet_app'
password = '****'

db_uri = f'mysql+pymysql://{user}:{password}@{host}/{database}?charset=utf8'
app.config['SQLALCHEMY_DATABASE_URI'] = db_uri
```

```
db = SQLAlchemy(app)


class User(db.Model):
  __tablename__ = 'user'
  id = db.Column(db.Integer, primary_key = True, autoincrement = True)
  name = db.Column(db.Text())
  mail_address = db.Column(db.Text())
  password_hash_md5 = db.Column(db.Text())
  groups = db.Column(db.JSON())
  addresses = db.Column(db.JSON())
  request = db.Column(db.JSON())
```

このソースコードでは、SQLAlchemy というツールを使って、MySQL のデータベースにアクセスし、MySQL のデータを Python において使いやすい形に変換してくれます。SQL と Python を使う Web アプリケーションには欠かせません。

5.5.2 ユーザー情報の格納の仕方

そういうことで、じゃあもうログインの判定ってユーザーネームとパスワードの一致をそのまま判定すればいい！と思ったのですが、よく考えると、データベースの中にユーザーの情報をそのまま入れてしまうと、管理者にユーザーネーム・パスワードもろとも筒抜けになってしまいます。
それはちょっと良くないなと思ったので、パスワードだけですが、ユーザー登録するときに暗号化し、ログインするときも入力情報を同じ方法で暗号化して、暗号同士で照合することにしました。ちなみに、暗号化の方法は Hash と呼ばれるもののひとつで MD5 といいます。Hash 暗号は変換すると元に戻すことが非常に困難で元の情報がばれることがないので、様々な情報の管理に活用されています。ログインする場所（/login）のソースコードは次のようになっています。

ソースコード 5.4: app.py

```
# * パスワードの一致をチェック
def check_login(username, password_hash):
  # * そもそもログインしたいアカウントが存在するか
  here = User.query.filter(User.name == username).count()
  if (here == False):
    return False
  # * ログインしたいユーザーのデータを取り出し、パスワードのHash の一致をチェック
  user_data = User.query.filter(User.name == username)
  if (user_data[0].password_hash_md5 == password_hash):
    return True
  else:
    return False


```

```
15 # * ログインフォーム
16 @app.route('/login', methods = ['POST', 'GET'])
17 def login_form():
18   if request.method == 'POST':
19     # * 前のページから送られてきたデータを抽出
20     user_name = request.form['user_name']
21     password = request.form['password']
22     # * 入力したパスワードのハッシュを計算
23     enc_password = bytes(password, encoding = 'utf-8')
24     password_hash = hashlib.md5(enc_password).hexdigest()
25     # * ログインのパスワードを確認し、一致するならホーム表示、一致しないならもう一回ログイ
         ンさせる
26     if (check_login(user_name, password_hash)):
27       # * セッション(ブラウザがローカルで持ってる情報(暗号化済み))を編集
28       session['username'] = user_name
29       session['login'] = 'ok'
30       return redirect(url_for('index'))
31     else:
32       session['login'] = 'falied'
33       return redirect(url_for('login_form'))
34   login = True
35   if (session['login'] == 'falied'):
36     login = False
37   print(login)
38   return render_template('login.html', login = login)
```

これで、ログイン機能は実装完了です。
ほかの機能も、同じ要領で、MySQL のデータベースにつなげる→ app.py でデータベースを必要に応じて編集という流れでほとんど実装できます。

5.6 一人一人に違うページをレスポンスする

「2. そもそも Web アプリケーションってなに？」で、動的な Web サイトでは閲覧者によって違うレスポンスを返すと説明しました。しかし、それはどうやってやるのでしょうか？今から、その方法ついて説明します。
ソースコード 1 の app.py で、render template という関数を使っていました。これがカギになってきます。この関数は、第一引数に HTML ファイルの名前、第二引数以降に変数＝値を入れることで、第一引数の HTML ファイルで第二引数以降に定義した変数を使うことができます。例えば、TweetApp の index.html では、ソースコード 5 のようになっています。

ソースコード 5.5: index.html

```
1 <!DOCTYPE html>
2 <html lang ="ja">
3     <head>...</head>
```

```
4     <body>
5         <header>
6             <img id = 'icon' class = 'header-logo' src = '/static/favicon.ico'
                  width = 60 height = 60>
7             <p class = 'header-logo'><a href = '/'>TweetApp</a></p>
8             <i class="search-icon␣fa-solid␣fa-magnifying-glass"></i>
9             <form class = 'search' action = '/search' method = 'get'>
10                <input class = 'search-input' name = 'word'>
11            </form>
12            <ul id = 'header-holder' class = 'header-holder'>
13                {% if session['login'] == 'ok'%}
14                    <li id = 'username'>{{ session['username'] }}</li>
15                    <li class = 'list'><a href = '/profile/{{ session['username']
                          }}'>プロファイル
                          </a></li>
16                    <li class = 'list'><a href = '/logout'>ログアウト</a></li>
17                {% endif %}
18                {% if session['login'] != 'ok' %}
19                    <li id = 'username'>? ? ? ?</li>
20                    <li class = 'list'><a href = '/login'>ログイン</a></li>
21                    <li class = 'list'><a href = '/register'>新規登録</a></li>
22                {% endif %}
23            </ul>
24        </header>
25    </body>
26    <div class = 'copy-container'>...</div>
27    <div class = 'about'>...</div>
28 </html>
```

このコードで、{%%} や {{}} などの見慣れない括弧があるのがわかると思います。ここで Python + Flask の力が発揮されます。実は、{%%} 内では Python の条件分岐などが使えて、さらに {{}} 内では render template で渡された変数や、セッションの値（session[] の部分）などを表示することができるのです。

この機能によって、ブラウザ内に保存されたセッションや、app.py から処理されて送られてきた値を使うことができます。このようにして、MySQL のデータや、閲覧者によって違うセッションの値を Web ページに反映できるというわけなんです。

この技術を応用して、MySQL に投稿を保存してサイトで表示したり、個人チャットを作ったりしました。

5.7 終わりに

この紙版の部誌では Python で Web サイトを作ることの概念と、僕の開発した「TweetApp」の体験版のようになってしまいましたが、Web 版ではアプリの機能一つ一つにもっと踏み込むような内容になっています。

ぜひ、この紙版の部誌を読んでくれた人は、Web 版の部誌も読んでくれると嬉しいです。ちなみに、この「TweetApp」のソースコードは以下の URL・QR コードのサイトにすべて載っているので、興味のある人は見てみてください。

https://github.com/mikan0131/TweetApp

第 6 章

サイクリングの周回ルートを探索する

79 回生 number_cat

6.1　はじめに

こんにちは、79 回生の number_cat です。いつのまにか高 2 になってしまいました。

僕はときどきサイクリングをしているのですが、サイクリングルートを自動で作成できたら便利だと思ったので、自分で作ってみることにしました。

今回作成するルートは、特定の目的地を設定するのではなく、一定距離を走行して出発地点に戻る「周回ルート」です。特に、なるべく良いルートを生成することを目指します。

6.2　周回ルートの良さを評価する

良いルートとはどのようなものでしょうか。まず、以下の 2 つの基準を満たしていることが望ましいと考えます。

1. **指定距離に近い**（例：30km のルートを作る場合、なるべく 30km に近いこと）
2. **道の重複が少ない**（同じ道を何度も通るのは面白みに欠けるため、できるだけ異なる道を通ること）

さらに、今回は次の基準も考慮します。

3. **路上条件が快適である**（なるべく坂が少なく、広い道を通ること）

快適な路上条件の定義には議論の余地がありますが、本研究では、坂の少なさや道路の広さを指標とします。そして、この基準がどの程度満たされているかを評価し、単純な手法と比較してどれだけ快適なルートを選択できているかを検証します。

これらの基準を数値化するため、それぞれ「距離ペナルティ」「重複ペナルティ」「非快適度」を算出し、3 つの合計を「総ルートコスト」と定義します。この値が小さいほど、より良いルートであると判断します。

6.3　実装環境

実際にルートを作成するプログラムを実装しました。本研究では神戸市内を対象としています。

道路データは OpenStreetMap から取得し、勾配を求めるための標高データは国土地理院のデータを使用しました。

データの取得には Anaconda 環境で OSMnx と GeoPandas を用いました。具体的には、OpenStreetMap の道路データを OSMnx で取得し、標高データとして 国土地理院の基盤地図情報数値標高モデル（10m メッシュデータ） を使用しました。この標高データは 基盤地図情報ビューア を用いて Shape 形式に変換し、それを GeoPandas に読み込んで処理しました。

取得したデータを一度テキストファイルに出力し、そのあと C++ で読み込んで経路探索アルゴリズムを実行しました。

6.4 手法

周回ルートは最大 40km 程度なので、出発地点から半径 20km 以内の道路を考えれば十分です。この範囲の交差点（ノード）数と道路（エッジ）数は約 10 万ほどあり、効率的なアルゴリズムが必要になります。計算量は $O(E \log V)$（E は道路数、V は交差点数）に抑えました。

また、簡単のため、出発地点は交差点の一つとします。

6.4.1 ルートの作成方法

まず、2 つの経由点を選び、出発地点とそれぞれを結ぶことで周回ルートを作成します。各点間のルートは、坂のきつさや道の広さを考慮した「非快適度」に道の長さを重みづけしたものをコストとして扱い、その最小コストルートをダイクストラ法で求めます。ただし、経由点の選び方によって距離が大きく変わるため、さまざまな候補を試し、最も総ルートコストが小さいものを採用します。

6.4.2 計算量の削減

ここで問題になるのは計算量です。単純に複数個（ n 個）の経由点の組み合わせについて最小コストルートを求めると、そのたびに $O(E \log V)$ 時間かかるので、全体で $O(nE \log V)$ になってしまいます。

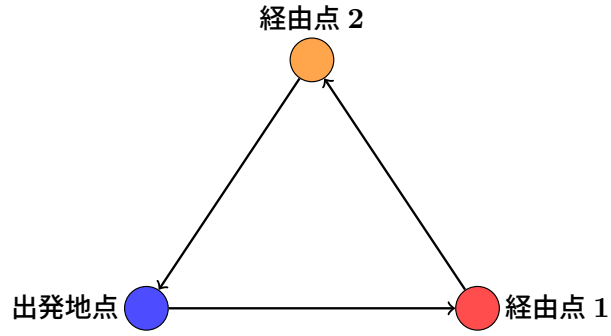
そこで、1 つ目の経由点（経由点 1）を固定し、2 つ目の経由点（経由点 2）を動かすことにします。こうすると、ダイクストラ法を 2 回使うだけで、

- 経由点 1 $\rightarrow$ 各交差点 の最小コストルート
- 各交差点 $\rightarrow$ 出発地点 の最小コストルート

がそれぞれ求まり、経由点 1 を固定したときに各交差点を経由点 2 とする周回ルートを $O(E \log V)$ で算出できます。最終的に、試したルートの中で総ルートコストが最小のものを出力します。

補足：経由点 1 は、出発地点からのルート長が 30〜40% で、非快適度の平均が低い点とします。極端な形状のルートの生成を防ぐためです。

実装の詳細は https://github.com/numbercat314/cycling_route_search に載せる予定です。



6.5 経路例

図 6.1

図 6.2

6.6 検証

路上条件を考慮できているのか確認するために、距離と重複のみを考慮したナイーブな手法と総ルートコストを比較します。

神戸市内を対象として、ランダムに 100 の出発地点を選びます。さらに、5 種類の距離を指定して、総ルートコストの平均値を算出しました。結果を以下の表に示します。

$X_{[\mathrm{km}]}$	今回	ナイーブ	比率
3	0.348	0.453	0.77
5	0.305	0.393	0.78
10	0.195	0.292	0.67
20	0.149	0.256	0.58
40	0.145	0.240	0.60

表 6.1: 提案手法と比較手法の結果

今回の手法では、ナイーブな手法と比べて総ルートコストを 20〜40% 削減できています。つまり、路上条件をしっかり考慮できていることが分かります。

6.7 おわりに

本研究では、サイクリングの周回ルートを、路上条件を考慮しながら探索できるプログラムを実装しました。その結果、ナイーブな手法よりも良いルートを生成できることが確認できました。

今回は「なるべく坂が少なく、広い道を通る」ことを快適な路上条件としましたが、「なるべく坂が多く、狭い道を通る」といった設定に変更しても、同じアルゴリズムでルートを作成できます。ただし、「少し坂がほしい」といったルート全体のバランスを考慮しないと最適化できない条件には対応できません。

こうしたルート全体のバランスを考慮した最適化を行う手法として、焼きなまし法などの確率的最適化手法が考えられます。焼きなまし法を用いることで、局所最適解に陥ることを防ぎつつ、総ルートコストがより小さい解を探索できるように思います。どうして思いつかなかったのでしょうか……。焼きなまし法のための初期解として利用することくらいはできそうですが。

また、現状はシミュレーションのみなので、実際にこのルートを走ってみて、体感的な評価を検証するのも面白そうです。

以上、サイクリングの周回ルート探索でした！

6.8 参考文献

c60evaporator. ”【Python で GIS】GeoPandas まとめ”. Qiita. 2021-07-04.
https://qiita.com/c60evaporator/items/ac6a6d66a20520f129e6

NAVITIME_Tech. ”自転車向け「周遊ルート機能」の開発秘話”. note. 2024-02-28.
https://note.com/navitime_tech/n/n9b060bc0dd82

ritogk. ”地図からこだわりの道を抽出する方法”. Zenn. 2023-12-25.
https://zenn.dev/homing/articles/f9a314841c737d

Captain-K. ”OpenStreetMap と Python で地域道路データを解析: OSMnx と GeoPandas の使い方”. Hangout Laboratory.
https://hanlabo.co.jp/memorandum/3110

moshi. ”【Python】OSMnx で車両走行道路の最短経路探索＆可視化をしてみる”. Qiita. 2021-04-12.
https://qiita.com/moshi/items/e383491664d028cd2166

WisteriaHill. ”OSMnx を使って道路ネットワークを取得してみる”. FRONT. 2018-11-02.
https://wisteriahill.sakura.ne.jp/CMS/WordPress/2018/11/02/osmnx-graph-network

kntoshiya. ”NetworkX で地理院ベクトルタイルの道路データを使ってネットワーク解析”. Qiita. 2023-12-16.
https://qiita.com/kntoshiya/items/77a8964fd36ef57e4425