Pythonで試して解く
情報Iプログラミング
試作・過去問題集

プログラムは多くの分野で多様な仕事をしています。同様に入試においても、いろいろな問題を出題することができます。そのため、数学や理科に比べて過去問題と類似した問題が出題されることは少ないと予想されます。

したがって、過去問題を学習することは、その答えを覚えるのではなく、入試に必要なプログラムの本質的な考え方やテクニックを習得することが目的です。

ドラフト版 2024年9月

太田 剛

学習を開始する前に読んでください

この教材について

この教材は2025年より始まる大学入学共通テストの「情報I」のプログラミング分野の学習を支援するために作成されました。2024年時点ではまだ「情報I」の試験は始まっていませんが、大学入試センターや多くの情報系の一般入試を行う私立大学では情報Iに対応した試作問題を公開しています。また共通テストでは「情報関連基礎」科目として過去にプログラムミングを含む試験を実施しています。

この教材では、これらの試作や過去問題から情報Iの試験の出題に類似した問題を抽出して、Pyhtonで実行しながら回答できるようにしたものです(多くの問題は、日本語で記述されるプログラミングで表現される。配列などの具体的な記述が行われていない。大きなプログラムを複数に分けて出題されるなど、実際のプログラミング環境で実行するには適切ではないため)

学習の進め方について

この教材では大きく[プログラミング問題]と[プログラム例]から構成されていて、下の表に示す14個の問題から構成されています。

[プログラミング問題]で示された問題説明と実行例を見て、プログラムの穴埋め問題の[????]に入るプログラムを考えみてください([????]はほぼ元になった問題の穴埋め箇所に対応しています)。なお、元の問題に比べて説明は必要最低限であることと、元の問題では回答は選択枝から選ぶようになっているので、この教材は、やや難易度が高いかもしれません。ただし、Pythonのプログラム環境で実際に試行錯誤しながら回答していく学習を想定しています。(元の問題では、手順を追って丁寧に説明していることも多いので必要に応じて、後述するサイトから問題を入手してみてください。)

	タイトル	元の試作問題/過去問題
1	nまでの素数の個数を求める	南山大学試作問題2024年7月
2	連続した数になる確率	東北学院大学試作問題2024年
3	支払いとお釣りの硬貨の枚数を最小にする	入試センター試作問題2023年10月
4	生徒名簿の読み込みと集計	日本大学試作問題2024年
5	魔法陣の確認	共通テスト情報関係基礎2024年問3-2
6	あみだくじ	共通テスト情報関係基礎2022年
7	すごろくゲーム	共通テスト情報関係基礎2021年
8	6つの数字の次に大きな数	電気通信大試作問題2023年11月
9	平面でのロボットの移動	共通テスト情報関係基礎追加2021年
10	データの圧縮	京都産業大学試作問題2024年
11	ゲームプレイヤーの管理	共通テスト情報関係基礎追加2024年
12	選挙のドント表	入試センター試作問題2021年3月
13	パスワードの総当たり攻撃	共通テスト情報関係基礎追加2022年
14	講師配置	共通テスト情報関係基礎追加2023年

学習を開始する前に読んでください

Pythonでの学習についての注意点

●ある程度Pythonでプログラムが開発できることが学習の開始条件です。特に二次元配列、while文、関数定義、論理式などについての知識も必要なので、これらについて知識がない場合は適宜学習しながら進めてください。
●前述したように多くの試作・過去問題は日本語のプログラム言語で記述していたり、使用する配列の定義が明確でないことも多いです。そのためPythonで動作するようにしたため次の点に注意してください。
- 試作・過去問題では配列の添え字が1から始まるものがありますが、Pythonの配列では添え字が0から始まるため、問題自体を一部変更したり、定義した添え字が0の値にダミーのデータ(0, 9, .)を設定している場合があります。
-試作・過去問題では二次元配列の個々のデータの記述を例えは、D[a, b]で表現していますが、Pythonでの表記でD[a][b]にしています。
-問題によっては配列を初期化するものがありますが、特に2次元配列ではPythonの特性から次のように初期化しているところもあります。

```
Yonda = [[0 for i in range(yoko+1)] for j in range(tate+1)]
```

●ループ処理において、試作・過去問題では次のように指定していますが、

i を1からxまで1ずつ増やしながら

pyhtonではrange()を使用して次のように指定しています。

```
for i in range(1, x+1, 1):
```

元の試作・過去問題の参照について

次の二つのサイトからアクセスしてみてください。
・キミのミライ発見(河合塾)学部・学科/入試方式別掲載問題一覧
試作問題と大学入試センターの過去問題がアクセスできます。
https://www.wakuwaku-catch-mondai.net/question/
・大学入試センターの過去問題
過去3年分の過去問題がアクセスできます(追加と示した部分もあり)
https://www.dnc.ac.jp/kyotsu/kakomondai/

図表を含む引用について

・本教材では、前表で示した試験問題を参考に作成しいます。
・大学入試センターの過去問題から図表を引用しています。

プログラミング問題

1. nまでの素数の個数を求める

南山大学試作問題2024年7月改

問題説明

nまでの素数を個数cを求める。

手順/考え方

1. 予めnまでの数値が素数があるか判断するための一次元配列Pを用意する。配列の添え字は素数かどうか判断する数値と対応している。
2. プログラム実行後、Pには素数の場合は1、そうでない時には0を格納する。なお、配列が0から始まること、1は素数でないため、予めP[0]とP[1]に0を代入しておく。その他の要素にはすべて1を代入しておく。
3. 素数かどうかの判断: 2からn-1の数値において、各数値の倍数は素数でないため、対応する要素に0を設定する。
4. 配列Pの中で1の数を数えると素数の個数がわかる。

プログラムの穴埋め問題

```
#南山大学試作問題2024年7月
n = 10     #素数か判断する最大の数
P = [1]*(n+1)
P[0] = 0
P[1] = 0
for i in range(2, n, 1):
    for j in range(i+1, n+1, 1):
        if [????]:
            P[j] = 0
c = 0
k = 1
while(k<n):
    if [????]:
        [????]
    [????]

print("P=",P)
print("素数の数=",c)
```

実行例

```
P= [0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0]
素数の数= 4
```

2. 連続した数になる確率

東北学院大学試作問題2024年改

問題説明

1から13までの番号が書かれた13枚のカードがある。これらのカードからランダムに2枚のカードを選ぶとき，選ばれた2枚のカードに書かれた番号が連続した数値となる確率p を計算するプログラムについて考える。

手順/考え方

・13枚のカードから2枚をランダムに選ぶ組み合わせの数を求める。

・その中で、連続した番号になる組み合わせの数を計算する。この場合、連続した数の小さい方の数をiとすると、もう一つの数はi+1となる。

プログラムの穴埋め問題

```
#東北学院大学試作問題2024年
x = 0 #総数
y = 0 #連続した組み合わせ数
for i in [????]:
    for j in  [????]:
        x = x + 1
        if [????]:
            [????]
p = [????]
print("総数=",x)
print("連続した組み合わせ数=",y)
print("p=",p)
```

実行例

```
総数= 78
連続した組み合わせ数= 12
p= 0.15384615384615385
```

3. 支払いとお釣りの硬貨の枚数を最小にする　入試センター試作問題2023年10月改

問題説明

　価格に対して支払いとお釣りの硬貨の枚数を最小にするようなプログラムを考える。

手順/考え方

・使用する硬貨は、1円、5円、10円、50円、100円とする
・まず、与えられた金額に対して、硬貨の枚数が最小となるような関数を考える。この関数では、大きい金額の硬貨から何枚必要か考える、これは金額をその硬貨金額で割った商がその枚数になる。そして、その余りで次の大きい硬貨が何枚かという処理を繰り返す。
・釣銭の最大は99円と想定してプログラムを作成する。そして価格と釣銭を足したものが支払い金額になる。
・この時、釣銭の方を1～99円に変化させた時、元の価格に対して、支払いと釣銭の枚数が最小になる場合を求める。例えば定価46円のものを購入する時、支払いが51円で50円1枚.1円1枚、釣銭が5円1枚の時が合計の硬貨の枚数が最小になる。

プログラムの穴埋め問題

```
#入試センター試作問題2023年10月
def min_maisu(k):
    Kouka = [0, 1, 5, 10, 50, 100]
    maisu = 0
    nokori = k
    for i in [????]:
        maisu = [????] + [????]
        nokori =  nokori % Kouka[i]
    return maisu

kakaku = 46  #想定した価格
kekka_maisu = 100 #初期値として最大枚数を100とする
for [????] in range(100): #釣銭は最大99円と想定
    shiharai = kakaku + turi
    t_maisu = [????]
    if [????] < kekka_maisu:
        [????] = [????]
print("最小交換枚数=",kekka_maisu)
```

実行例

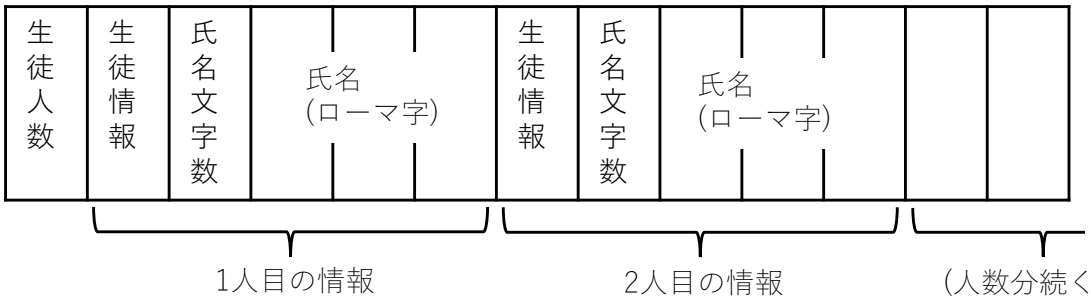
```
最小交換枚数=  3
```

4. 生徒名簿の読み込みと集計

日本大学試作問題2024年改

問題説明

生徒名簿から情報を読み込み東京都の生徒の人数を数えるプログラムを考える。



データは上図のような要素から構成されるファイルに格納されている。

生徒情報の構成は8ビットで構成されていて、居住地の情報は下位2ビットで表していて、00:東京都, 01:神奈川県, 10:それ以外が格納されている。

プログラムではreadData()という関数を使用し、これは関数から1要素ずつ読み込み戻り値とするものである。readData()で読み込んだ後は、もう一度readData()を使用すると次の要素を読み込む。

手順/考え方

・はじめにreadData()で生徒人数をnに読み込みます。

プログラムの穴埋め問題

```
# 日本大学 試作問題2024年
data_pos = 0
Org_Data = [4, 0b00100100, 3, "A","B","C", 0b01001001, 2,
"E","E",0b01001010, 3, "O","P","Q",0b01100100, 2, "X","Y"]
#問題外開始
def readData():
    global data_pos
    t_data = Org_Data[data_pos]
    data_pos = data_pos + 1
    return t_data
#問題外終了
sum_tokyo = 0
i = 0
n = readData()
while([????]):
    b=readData()
    if (b & 3) == 0:
    # if (b & 0b11) == 0:
        sum_tokyo = [????]
    b=readData()
    [????]
        c = [????]
    i = i + 1
print("東京の生徒数=",sum_tokyo)
```

実行例

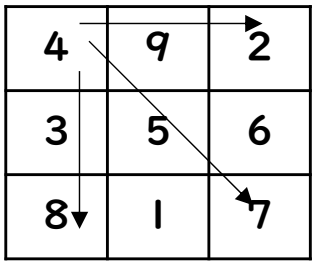
```
東京の生徒数= 2
```

5. 魔法陣の確認

共通テスト「情報関係基礎」2024年問3-2改

問題説明

正しい魔法陣か確認するプログラムを考える



手順/考え方

・　魔法陣はn X nの数値の表であり、2つの斜めの方向の合計、すべての列の合計、すべての行の合計が等しいものである。また、表の中の数に重複はないものである。

・すべての列の合計が等しいかの確認: 各行の合計をwaに順次いれていく。hantei_waには初めの行のwaを代入しておき、2行目以降はこのhantei_waと各行のwaを比較して、もし等しくなければbatuに1を入れておき、最後に判断する。

・数に重複がないかの確認: 一次元配列の`Kakunin`を用意しておき、ここに数があったかどうか記録しておく。

実行例

```
[4, 9, 2] <- 正しい魔法陣で実行
[3, 5, 7]
[8, 1, 6]
各行の和は一致しました
数の重複はありません

[4, 9, 7] <- 誤った魔法陣で実行
[3, 5, 7]
[8, 1, 6]
各行の和は一致しません!
数の重複がありました

[4, 9, 2] <- 誤った魔法陣で実行
[3, 5, 6]
[8, 1, 7]
各行の和は一致しません!
数の重複はありません
```

続く

5. 魔法陣の確認(続き)　　共通テスト「情報関係基礎」2024年問3-2改

プログラムの穴埋め問題

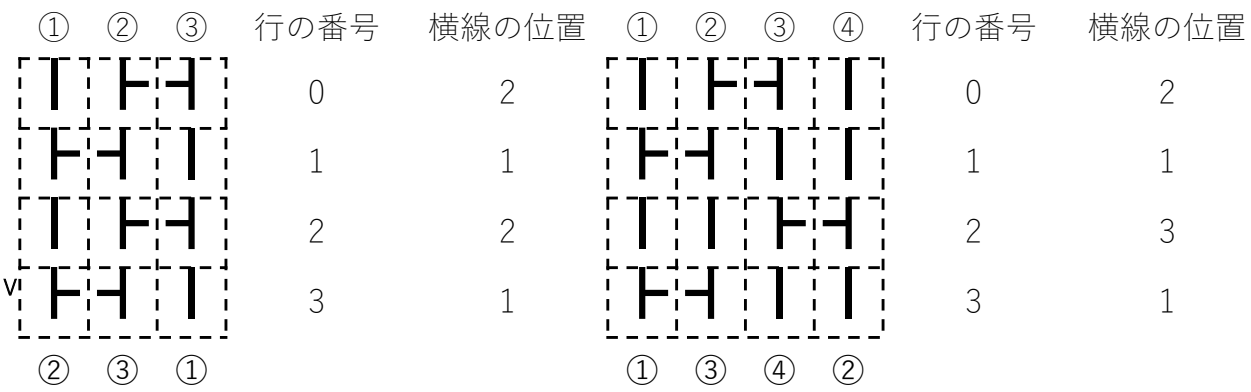
```
#大学入学共通テスト情報関係基礎2024問題3-2
Mahojin = [[4, 9, 2],
           [3, 5, 7],
           [8, 1, 6]]
for i in range(3):
    print(Mahojin[i])

#各行の和の一致
n = 3
hatei_wa = 0
batu = 0
for i in range(3):
    wa = 0
    for j in range(3):
        wa = [????]
    if [????] :
        hatei_wa = wa
    [????]
        batu = 1
if batu == 1:
    print("各行の和は一致しません!")
else:
    print("各行の和は一致しました")
#重複無しの確認
n = 3
Kakunin = [0] * (n * n)
f_batu = 0
for gyou in range(3):
    for retu in range(3):
        Kakunin[[????]] = 1
for i in range(n*n):
    if [????] == 0:
        [????]
if f_batu == 1:
    print("数の重複がありました")
else:
    print("数の重複はありません")
```

6. あみだくじ

共通テスト情報関係基礎2022年改

問題説明

あみだくじを表示し、結果を求めるプログラムを考える。



コンピュータであみだくじを考えるため、まず、「┃」「┣」「┤」の3つの文字を組み合わせて作成する。そして、縦(tate)と横(yoko)のマス目の集まりで考える。そしてプログラムを簡単にするため、各行に一個だけ横線があるとした。そして横線の位置をxとすると、xはx～x+1番目の縦線に横線があることを示す。プログラムでは、この横線の位置をYokosenの配列に格納し、左図ではYokosen=[2, 1, 2, 1]となる。

課題1: あみだくじを表示する関数を作る。

課題2: あみだくじの結果を表示する。

手順/考え方(課題2について)

・コマの状態(①②③の並び)をKoma配列に格納する。左図で初期状態ではKoma=[1, 2, 3]となる。

・コマの状態の変化は0行目から1行づつ考えていく。0行目のYakosen[0]は2なので、この時、Koma[2-1]と一つ前のKoma[2]の内容を入れ替える処理を行う。(Komaの添え字は0から始まるため)

実行例

```
[1, 2, 3]
 ┃ ┣
 ┣ ┃
 ┃ ┣
 ┣ ┃
[2, 3, 1]
```

6. あみだくじ(続き)

共通テスト情報関係基礎2022年改

プログラムの穴埋め問題

#大学入学共通テスト情報関係基礎2022

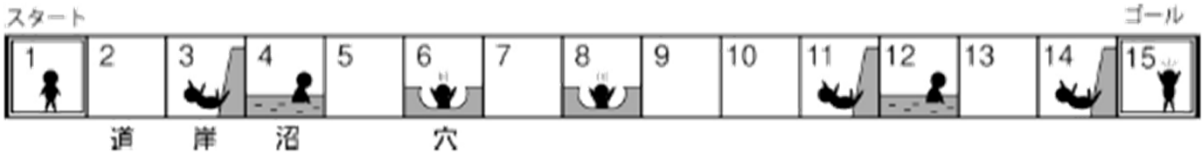
```
# 課題1:あみだくじを表示する
def dip_amida(tate, Yokosen, yoko):
    for y in range(tate):
        x = 1
        while( x <= yoko):
            if [????]:
                print("┣", end = "")
                [????]
            else:
                print("┃", end = "")
            x = x + 1
        print("")
    return()

tate = 4
yoko = 3
Koma = [1, 2, 3]
Yokosen = [2, 1, 2, 1]
print(Koma)
dip_amida(tate, Yokosen, yoko)

#課題2:あみだくじの結果を求める
for y in range(len(Yokosen)):
    t = Koma[Yokosen[y]]
    [????]
    [????]
print( Koma)
```

7. すごろくゲーム

共通テスト情報関係基礎2021年改

問題説明

サイコロを使った直線的なすごろくゲームで遊ぶことをシミュレーションするプログラムを考える



スタートと道: 出た目の数だけ前進する。
崖: 出た目の数だけ後退する。
穴: 出た目が4以上ならば、出た目の数だけ前進する。3以下ならば、そこにとどまる。
沼: 出た目の数の半分だけ前進する。小数点以下は切り捨てる

手順/考え方

・　プログラムでは次の配列に情報を格納する。
　　Masu[k]: マスkの効果値
　　Saikoro[i][r]:ラウンドrにおけるプレイヤーiの出た目
　　Koma[i][r]: ラウンドrにおけるプレイヤーiの更新後のコマの位置
・効果値は、マスの種類ごとに道:1, 崖:-1, 沼:0.5, 穴:0を設定し、崖の場合には出た目によって効果値をラウンドごとに変更している。
・何人でも遊べるように、プレーヤーの人数をninzuに格納しておく、またowariにはゲームの状態を格納し、どのプレーヤーもゴールしていない場合は0を、誰かがゴールした場合は1を設定する。
・数値の小数点以下を切り捨てて、整数にするためにint()を使用している。
・サイコロの出目はrandom.randint(1,6)で1～6までのランダムな整数を発生させている。
・SaikoroとKomaは配列の要素の数が分からないので、例えばSaikoro[i].append()のように配列に新しい要素を追加していっている。

実行例

サイコロの目 <- 実行するたびに結果が違います。
1 番さん [0, 5, 5, 6, 5, 4]
2 番さん [0, 6, 4, 2, 4, 5]
コマの位置
1 番さん [1, 6, 11, 5, 10, 14]
2 番さん [1, 7, 11, 9, 13, 15]

サイコロの目
1 番さん [0, 3, 5, 2, 4, 5]
2 番さん [0, 3, 5, 5, 6, 2]
コマの位置
1 番さん [1, 4, 6, 6, 10, 15]
2 番さん [1, 4, 6, 11, 5, 7]

続く

7. すごろくゲーム(続き)

共通テスト情報関係基礎2021年改

プログラムの穴埋め問題

```
#大学入学共通テスト情報関係基礎2021-A
import random
Masu = [0, 1, 1, -1, 0.5 , 1, 0, 1, 0, 1, 1, -1, 0.5, 1 -1, 2]

ninzu = 2
owari = 0
r = 0

Koma =  [[0] for i in range(ninzu+1)]
Saikoro = [[0] for i in range(ninzu+1)]
for i in range(1, ninzu+1, 1):
    Koma[i][r] = 1
while(owari == 0):
    r = r + 1
    for i in range(1, ninzu+1, 1):
        Saikoro[i].append(random.randint(1,6))
        k = Koma[i] [????]
        bairitu = Masu[k]
        if bairitu == 0 and [????] >= 4:
            bairitu = [????]
        idou = int(Saikoro[i][r] * [????])
        Koma[i].append(k + [????])
        if Koma[i][r] < 1:
            Koma[i][r] = 1
        if Koma[i][r] >=15:
            Koma[i][r] = 15
            owari =1
print("サイコロの目")
for i in range(1, ninzu+1, 1):
    print(i,"番さん",Saikoro[i])
print("コマの位置")
for i in range(1, ninzu+1, 1):
    print(i,"番さん", Koma[i])
```

8. 6つの数字の次に大きな数

電気通信大試作問題2023年11月改

問題説明

1～6までの数字を使って6桁の数を作る。数nが示された時に、nの次に大きな数を求めるプログラムを考える。なお、 1～6は必ず1回使用し、重複はないものとする。

手順/考え方

・例えば、342651の次の大きな数を考える場合、2は3/4より小さいので、上位の桁34xxxxの範囲が次の大きな数で、2651の中で次に大きな数を見つければ良い。

・2651では、2の次に大きな5xxxというの次に大きな数になり、のこった2/6/1でできる最も小さい数は126になる。これから345126が次に大きな数になる。

・プログラムでは、Narabiという配列に各桁ごとの数値を格納しておく。

・プログラム作成にあたってureverse(my_list, i, j)という関数を使用する。この関数は、my_listの配列のi番目からj番目の内容を逆順に入れ替えるものである。

例: Narabi = [1,3,5,2,4,6]
ureverse(Narabi, 2, 5)
実行後のNarabiの内容は、[1, 3, 6, 4, 2, 5]

プログラムの穴埋め問題

```
# 電気通信大試作問題2023年11月
#問題外開始
def ureverse(my_list, i, j):
  # iからjまでの要素をスライスで切り出し、逆順にして元のリストに代入
  my_list[i:j+1] = my_list[j:i-1:-1]
#問題外終了

Narabi = [1,2,3,5,4,6]   # 自由に設定してください。
print("元の数の並び=", Narabi)
ketasu = len(Narabi)
i = ketasu -2
while(i>=0 and [????]):
    i= i -1
if([????]):
    j = ketasu - 1
    while( i< j and [????]):
        j = j -1
    Narabi[i], Narabi[j] = [????]
    ureverse([????])
    print("次に大きな数の並び=",Narabi)
else:
    print("最も大きな数の並びです")
```

実行例

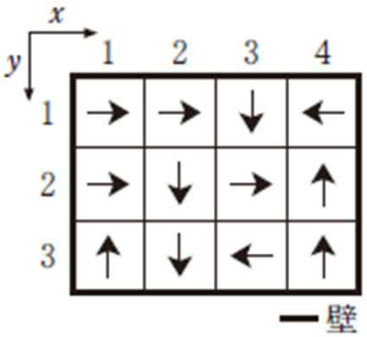
```
元の数の並び= [1, 2, 3, 4, 5, 6]
次に大きな数の並び= [1, 2, 3, 4, 6, 5]

元の数の並び= [5, 6, 4, 3, 2, 1]
次に大きな数の並び= [6, 1, 2, 3, 4, 5]
```

9. 平面でのロボットの移動

共通テスト情報関係基礎追加2021年改

問題説明

平面でのロボットの移動をシミュレーションするプログラムを考える。



同じ大きさの正方形のタイルが敷き詰められた部屋の中でロボットが移動する。各タイルには、次に移動する方向が矢印でしめされている。
この部屋の任意のタイトルにロボットを置いたときの移動をシミュレーションする。なお、壁にぶつかるとロボットは止まる。また、同じタイルに戻ると同じ動作を繰り返し続けるため、ロボットを止めることとする。

手順/考え方

・　部屋の構造とタイルの向きはTairuという二次元配列で定義する。x方向の数はyoki, y方向の数はtateで表す。また、配列の中で矢印の向きを、上:U, 下:D, 右:R, 左:Lで定義する。このときx, y位置のタイルの矢印の向きはTairu[y][x]になる。

・　同じタイルに戻ったか判断するため、Tairuと同じ構造の二次元配列Ypndaを用意する。Youndaは0で初期化し、ロボットが移動したところを1に変更する。そして、ロボットが移動した後、そのYoundaの内容が1であれば、すでにそのタイルは通過していると判断できる。

実行例

```
['.', '.', '.', '.', '.']
['.', 'R', 'R', 'D', 'L']
['.', 'R', 'D', 'R', 'U']
['.', 'U', 'D', 'L', 'U']
x=1   <- 入力
y=1   <- 入力
現在位置(x,y) 1 1
現在位置(x,y) 2 1
現在位置(x,y) 3 1
現在位置(x,y) 3 2
現在位置(x,y) 4 2
現在位置(x,y) 4 1
現在位置(x,y) 3 1
動き続ける
```

```
['.', '.', '.', '.', '.']
['.', 'R', 'R', 'D', 'L']
['.', 'R', 'D', 'R', 'U']
['.', 'U', 'D', 'L', 'U']
x=2   <- 入力
y=3   <- 入力
現在位置(x,y) 2 3
現在位置(x,y) 2 4
壁にぶつかる
```

9. 平面でのロボットの移動

共通テスト情報関係基礎追加2021年改

プログラムの穴埋め問題

```
#大学入学共通テスト情報関係基礎追加2021
Tairu = [[ ".", ".", ".",".",".",],
         [ ".", "R", "R","D","L"],
         [ ".", "R", "D","R","U"],
         [ ".", "U", "D","L","U"]]
tate = 3
yoko = 4
Yonda = [[0 for i in range(yoko+1)] for j in range(tate+1)]
for i in range(tate+1):
    print(Tairu[i])

x = int(input("x="))
y = int(input("y="))

print("現在位置(x,y)",x,y)

owari = 0
while( owari == 0):
    if (x >=1 and x <= yoko) and (y>=1 and y<= tate):
        t = Tairu[y][x]
        [????] = 1
    if t =="U":
        [????]
    elif t =="D":
         [????]
    elif t =="L":
         [????]
    elif t =="R":
        [????]
    print("現在位置(x,y)",x,y)
    if [????]:
        print("壁にぶつかる")
        owari = 1
    elif [????] == 1:
        print("動き続ける")
        owari = 1
```

10. データの圧縮

京都産業大学試作問題2024年改

問題説明

センサーから入手したデータを圧縮するプログラムを考える。
センサーからのデータは100以下の数値であり、0のデータが多い。そのため0のデータが連続する場合に圧縮することにした。0のデータが一つの場合は、そのままデータを利用する。0がn個連続していた場合は(257-n)を計算し、nを圧縮データとして使用する。なおnは2以上157以下でなければならない。

手順/考え方

・センサーからの読み込みではeofData(), getData(), そして圧縮データの保存(プログラムでは表示)のためにputData()という関数を使用する。 getData()は新しいデータを1個づつ取り出すものである。eofData()はデータがあれば0、データが終了した場合は-1を戻り値として返す。
・0又は複数連続したデータを保存するためputZeros(n)関数を作成する:nは連続した0の個数。圧縮したoを表すデータは156以下でなければならないため、156より大きい場合は、101を保存しnの値を156減らすという処理を繰り返す。

実行例

圧縮前データ= [66, 0, 7, 7, 7, 0, 0, 0, 29]
圧縮後データ= 66 0 7 7 7 254 29

圧縮前データ= [66, 0, 7, 0, 0, 7, 7, 0, 0, 0, 29]
圧縮後データ= 66 0 7 255 7 7 254 29

続く

10. データの圧縮(続き)

京都産業大学試作問題2024年改

プログラムの穴埋め問題

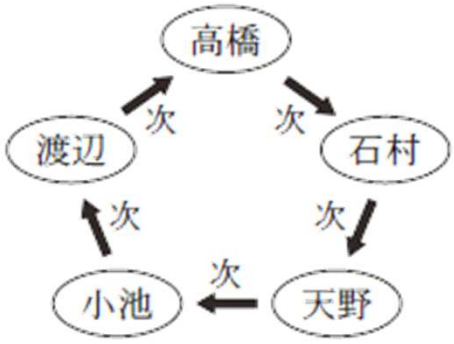
```
# 京都産業大学試作問題2024年-B
data_pos = 0
Org_Data = [66, 0, 7, 0, 0, 7, 7, 0, 0, 0, 29]
print("圧縮前データ=",Org_Data)
#問題外開始
def eofData():
    global data_pos
    global Org_Data
    ret = 0
    if data_pos == len(Org_Data):
        ret = -1
    return ret
def getData():
    global data_pos
    global Org_Data
    t_data = Org_Data[data_pos]
    data_pos = data_pos + 1
    return t_data
def putData(data):
    print(data, end = " ")
#問題外終了

def putZeros(n):
    count = n
    while([????]):
        putData(101)
        count = count - 156
    if count > 0:
        if count==1:
            data = [????]
        else:
            data = [????]
        putData(data)
    return

print(  "圧縮後データ= ", end = "")
length = 0
while(eofData() == 0):
    data = getData()
    if data == 0:
        [????]
    else:
        if length >0:
            putZeros(length)
            [????]
        putData(data)
if length > 0:
    putZeros(length)
```

11. ゲームプレイヤーの管理

共通テスト情報関係基礎追加2024年改

問題説明

複数で実施するプレーヤーと手番を管理するプログラムを考える。

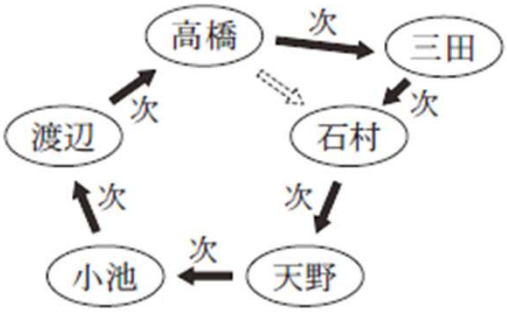


プレイヤーの手番の並び

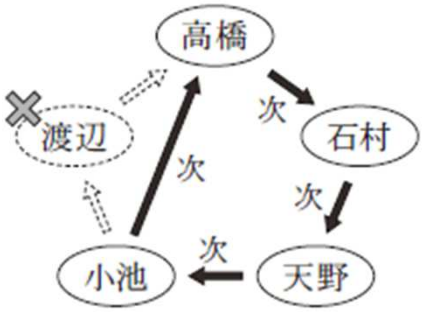
添え字	名前	次
0	小池	3
1	天野	0
2	石村	1
3	渡辺	4
4	高橋	2

プレーヤーは右のような表で示されるデータの配列で管理される。上図のようなゲームの手番の場合は、名前の次のデータは表で示したように、次の人の配列の添え字が格納される。

課題1: 配列をもとに手番順に名前を表示する。
課題2: 下左図のように、新しいプレーヤーを追加する。
課題3: 下右図のように一人のプレーヤーを削除する。
課題2と3では、図で示したような手順になるるように配列の中のデータも変更する。



プレイヤーを追加する例



プレイヤーが抜ける例

手順/考え方

・　データはPlayerの二次元配列に格納するPayer[i][d]でアクセス可能でd=0は名前、d=1で次のプレイヤーの配列の添え字を示す。

実行例

```
Player情報= [['小池', 3], ['天野', 0], ['石村', 1], ['渡辺', 4], ['高橋', 2]]
プレイ順番=小池, 渡辺, 高橋, 石村, 天野,
6 人　追加後のPlayer情報= [['小池', 3], ['天野', 0], ['石村', 1], ['渡辺', 4], ['高橋', 5], ['三田', 2]]
4 人　削除後のPlayer情報= [['小池', 4], ['天野', 0], ['石村', 1], ['渡辺', 4], ['高橋', 2]]
```

続く

11. ゲームプレイヤーの管理(続き)

共通テスト情報関係基礎追加2024年改

プログラムの穴埋め問題

```
#大学入学共通テスト情報関係基礎2024追加-A
Player = [["小池", 3], ["天野", 0], ["石村", 1], ["渡辺", 4], ["高橋", 2]]

print("Player情報=",Player)
# 課題1:play順に名前を表示する。
p = 0
print("プレイ順番=",end="")
for i in range(5):
    print([????], end=", ")
    p =[????]
print()

# 課題2:新規のプレーヤーの追加
n = len(Player)
Player.append(["三田", 0])
tuika = len(Player)-1
x = 4
Player[tuika][1] = [????]
Player[x][1] = [????]
n = [????]
print(n, "人　追加後のPlayer情報=",Player)

# 課題3:新規のプレーヤーの削除
# プレーヤーの配列を初期化している。
Player = [["小池", 3], ["天野", 0], ["石村", 1], ["渡辺", 4], ["高橋", 2]]
n = len(Player)
p = 0
nuke = 3
while([????]):
    p = [????]
Player[p][1] = [????]
n = [????]
print(n, "人　削除後のPlayer情報=",Player)
```

12. 選挙のドント表

入試センター試作問題2021年3月改

問題説明

ドント表でA～D等の当選人数を計算するプログラムを考える。

各政党の得票数と整数で割った商

	A党	B党	C党	D党
得票数	1200	660	1440	180
1で割った商	❷1200	❹ 660	❶1440	180
2で割った商	❺ 600	330	❸ 720	90
3で割った商	400	220	❻ 480	60
4で割った商	300	165	360	45

ドント表を使用した場合、得票数を1～順番の数で割った場合に、その値が大きい人から順番に当選させると考える。

手順1　配列 Tokuhyo の各要素の値を配列 Hikaku の初期値として格納する。
手順2　配列 Hikaku の要素の中で最大の値を調べ，その添字 maxi に対応する配列 Tosen[maxi] に 1 を加える。
手順3　Tokuhyo[maxi] を Tosen[maxi]＋1 で割った商を Hikaku[maxi] に格納する。
手順4　手順2と手順3を当選者数の合計が議席数の6になるまで繰り返す。
手順5　各政党の党名（配列 Tomei）とその当選者数（配列 Tosen）を順に表示する。

手順を書き出した文章

プログラムの穴埋め問題

```
#入試センター試作問題2021年3月
Tomei = ["A党","B党", "C党", "D党"]
Hikaku = [0,0,0,0]
Tokuhyo = [ 1200, 660, 1440, 180]
Tosen = [0,0,0,0]
tosenkei =0
giseki = 6
maxi = 0
for m in [????]:
    Hikaku[m] = Tokuhyo[m]
while [????] < giseki:
    max = 0
    for i in [????]:
        if max < Hikaku[i]:
            [????]
            maxi = i
    Tosen[maxi] = Tosen[maxi] + 1
    tosenkei = tosenkei +1
    Hikaku[maxi] = [????]//[????]
for k in range(4):
    print(Tomei[k], ":", Tosen[k], "名“, end=“  “)
```

実行例

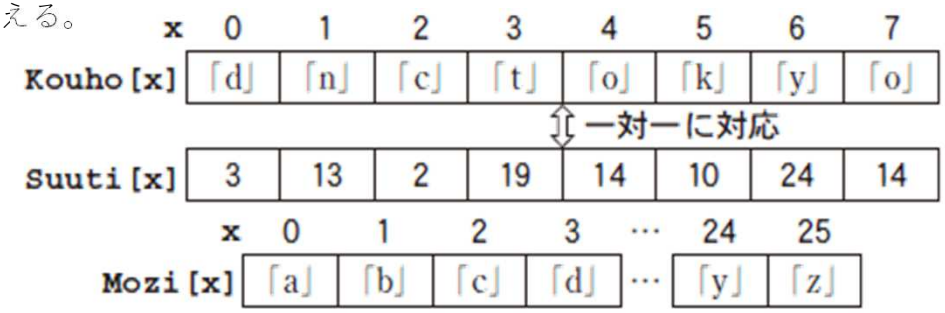
A党：2 名　B党：1 名　C党：3 名　D党：0 名

13. パスワードの総当たり攻撃

共通テスト情報関係基礎2022年追加改

問題説明

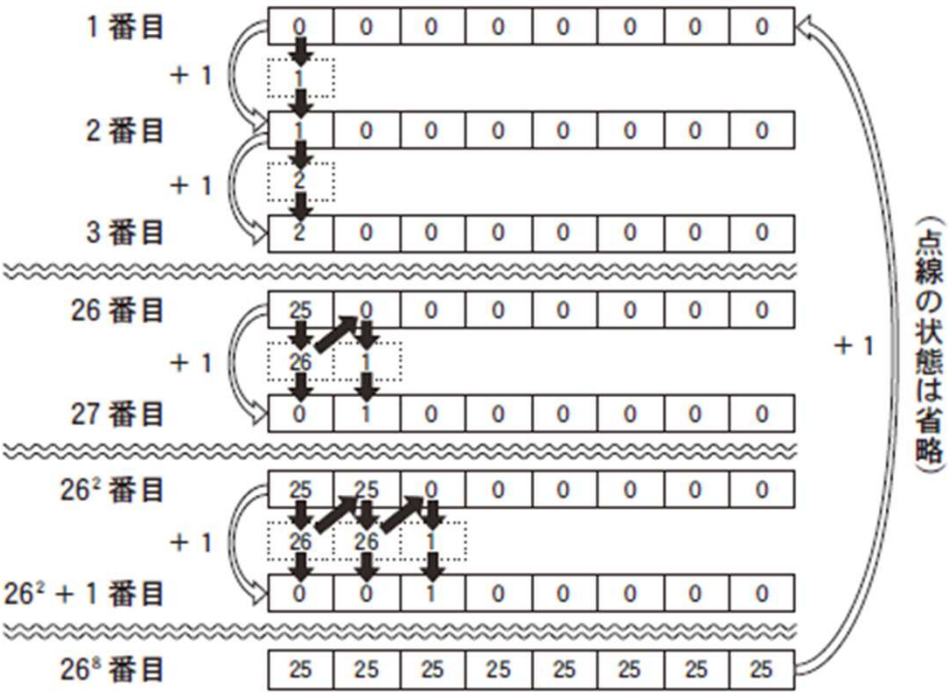
悪意を持ってパスワードを探す方法に総当たり攻撃がある。これは、すべての文字列をパスワードに一致するまで変更していく方法であり、この総当たり攻撃のプログラムを考える。



パスワードは8文字ですべて小文字を想定し、その候補の文字列も8文字になり、それを格納する配列をKouhoとする。この候補文字列を「aaaaaaaa」から「zzzzzzzz」まで、もれなく規則ただしく生成するため上図のような文字列と数値の対応を考える。Kouho内の文字とSuuti内の数値の対応はa=0, b=1,…となり、この対応関係を定義したMoji配列を用意する。

手順/考え方

・Suutiの内容を[0, 0, 0, 0, 0, 0, 0, 0]から順番に次のように増やして数を生成する。



すべての候補文字列に対応する配列 Suuti の変化

・Pythonでは文字列では個々の文字の添え字で利用できるため、配列の代わりに文字列を配列のように利用する。

Mozi = “abcdefghijklmnopqrstuvwxyz” の場合Moji[2]の内容はcとなる。

・補足: Seikaiに設定する文字列によっては、非常に処理時間がかかるため、次の良ような簡単な文字列で実行にしている。

Seikai = "abcaaaaa"

続く

13. パスワードの総当たり攻撃

共通テスト情報関係基礎2022年追加改

プログラムの穴埋め問題

```
#大学入学共通テスト情報関係基礎2022追加
# 次の数を求める関数
def nextSu(Suuti):
    t = Suuti[0]+1
    m = [????]
    n = [????]
    Suuti[0] = t % n
    j = 0
    while(j < m and [????]):
        j = j + 1
        t = Suuti[j] + 1
        Suuti[j] = t % n
    return(Suuti)

Seikai = "abcaaaaa"
Suuti = [0, 0, 0, 0, 0, 0, 0, 0]
Mozi = "abcdefghijklmnopqrstuvwxyz"
#正解の文字列と一致する推定文字列を見つけ出す
print("Seikai=", Seikai)
owari = 0
while(owari == 0):
    c = 0
    Kouho = ""
    for i in range(8):
        Kouho = Kouho + Mozi[Suuti[i]]
        if Seikai[i] == [????]:
            c = c + 1
    if [????]:
        print("Suuti=",Suuti)
        print("(文字)=", Seikai)
        for j in range(8):
            print(Mozi[Suuti[j]], end="")
        owari = 1
    else:
        Suuti = nextSu(Suuti)
```

実行例

```
Seikai= abcaaaaa
Suuti= [0, 1, 2, 0, 0, 0, 0, 0]
(文字)= abcaaaaa
abcaaaaa

Seikai= aaxaaaaa
Suuti= [0, 0, 23, 0, 0, 0, 0, 0]
(文字)= aaxaaaaa
aaxaaaaa
```

14. 講師配置

共通テスト情報関係基礎追加2023年改

問題説明

個別指導塾で1～5限に配置する講師数を最適化するプログラムを考える。

表1 すべての講師が1時限のみ担当

開始時限＼終了時限	1	2	3	4	5
1	10	0	0	0	0
2	0	8	0	0	0
3	0	0	19	0	0
4	0	0	0	14	0
5	0	0	0	0	7

表2 1時限を担当した講師の中で8名が2限まで担当

開始時限＼終了時限	1	2	3	4	5
1	2	8	0	0	0
2	0	0	0	0	0
3	0	0	19	0	0
4	0	0	0	14	0
5	0	0	0	0	7

表3 1時限を担当した講師の中で8名が3限まで担当

開始時限＼終了時限	1	2	3	4	5
1	2	0	8	0	0
2	0	0	0	0	0
3	0	0	11	0	0
4	0	0	0	14	0
5	0	0	0	0	7

個別指導を受ける生徒は1時限=10名, 2時限=8名, 3時限=19名, 4時限=14名, 5時限=7名と想定する。各講師が1時限しか担当しない場合は、各時間表１に示した人数が必要になる。また、時限を担当した講師の中で8名が2限まで担当すると、表2のような人数が必要になる。そして、 8名が3限まで担当すると表3に示した人数が必要になる。

手順/考え方

開始時限＼終了時限	1	2	3	4	5
1	3	0	0	0	7
2	0	1	0	0	0
3	0	0	12	0	0
4	0	0	0	7	0
5	0	0	0	0	0

初めに開始が1時限で連続時数が5時限配置(開始:1,終了:5)できる人数を検討するになる(表3の終了時限5では7) 。同様に4,3,2時限連続配置できる配置を考える。次に開始が2時限、開始が3時限から配置の考えていく。

プログラムでは。この表をHyouの二次元配列で格納していて、saisyouにその時の対角の最小値

続く

14. 講師配置

##共通テスト情報関係基礎追加2023年改

プログラムの穴埋め問題

```
#大学入学共通テスト情報関係基礎2023追加
Hyou =[[9, 9, 9, 9, 9, 9],
       [9, 10, 0, 0, 0, 0],
       [9, 0, 8, 0, 0, 0],
       [9, 0, 0, 19, 0, 0],
       [9, 0, 0, 0, 14, 0],
       [9, 0, 0, 0, 0, 7]]

jigensu = 5
for renzoku in range(5,1,-1):
    hajime = 1
    owari = [????]
    while([????] jigensu):
        saisyou = Hyou[hajime][hajime]
        for i in range(hajime, owari+1, 1):
            if [????]:
                saisyou = Hyou[i][i]
        if saisyou > 0:
            Hyou[Hajime][????]= saisyou
            for i in range(hajime, owari+1, 1):
                Hyou[i][i] = Hyou[i][i] - [????]
        hajime = hajime +1
        owari = owari +1
for i in range(6):
    print(Hyou[i])
```

実行例

```
[9, 9, 9, 9, 9, 9]
[9, 2, 0, 0, 1, 7]
[9, 0, 0, 0, 0, 0]
[9, 0, 0, 5, 6, 0]
[9, 0, 0, 0, 0, 0]
[9, 0, 0, 0, 0, 0]
```

プログラム例

プログラムはいろいろな記述方法があり、正答が複数あることから、「正答プログラム」ではなく「プログラム例」としている。

1. プログラム例: nまでの素数の個数を求める

```
#南山大学試作問題2024年7月
n = 10     #素数か判断する最大の数
P = [1]*(n+1)
P[0] = 0
P[1] = 0
for i in range(2, n, 1):
    for j in range(i+1, n+1, 1):
        if j % i == 0:
            P[j] = 0
c = 0
k = 1
while(k<n):
    if P[k] == 1:
        c = c + 1
    k = k + 1

print("P=",P)
print("素数の数=",c)
```

2. プログラム例:連続した数になる確率

```
#東北学院大学試作問題2024年
x = 0 #総数
y = 0 #連続した組み合わせ数
for i in range(1,13,1):
    for j in  range(i+1, 14, 1):
        x = x + 1
        if i + 1 == j:
            y = y + 1
p = y/x
print("総数=",x)
print("連続した組み合わせ数=",y)
print("p=",p)
```

3. プログラム例:支払いとお釣りの硬貨の枚数を最小にする

```
#入試センター試作問題2023年10月
def min_maisu(k):
    Kouka = [0, 1, 5, 10, 50, 100]
    maisu = 0
    nokori = k
    for i in range(5, 0, -1):
        maisu = maisu + nokori // Kouka[i]
        nokori =  nokori % Kouka[i]
    return maisu

kakaku = 46  #想定した価格
kekka_maisu = 100 #初期値として最大枚数を100とする
for turi in range(100): #釣銭は最大99円と想定
    shiharai = kakaku + turi
    t_maisu = min_maisu(shiharai) + min_maisu(turi)
    if t_maisu < kekka_maisu:
        kekka_maisu = t_maisu
print("最小交換枚数=",kekka_maisu)
```

4. プログラム例:生徒名簿の読み込みと集計

```
# 日本大学 試作問題2024年
data_pos = 0
Org_Data = [4, 0b00100100, 3, "A","B","C", 0b01001001, 2,
"E","E",0b01001010, 3, "O","P","Q",0b01100100, 2, "X","Y"]
#問題外開始
def readData():
    global data_pos
    t_data = Org_Data[data_pos]
    data_pos = data_pos + 1
    return t_data
#問題外終了
sum_tokyo = 0
i = 0
n = readData()
while(i < n):
    b=readData()
    if (b & 3) == 0:
    # if (b & 0b11) == 0:
        sum_tokyo = sum_tokyo +1
    b=readData()
    for j in range(b):
        c = readData()
    i = i + 1
print("東京の生徒数=",sum_tokyo)
```

5. プログラム例:魔法陣の確認

```
#大学入学共通テスト情報関係基礎2024問題3-2
Mahojin = [[4, 9, 2],
           [3, 5, 7],
           [8, 1, 6]]
for i in range(3):
    print(Mahojin[i])

#各行の和の一致
n = 3
hatei_wa = 0
batu = 0
for i in range(3):
    wa = 0
    for j in range(3):
        wa = wa + Mahojin[i][j]

    if hatei_wa == 0:
        hatei_wa = wa
    if hatei_wa != wa:
        batu = 1
if batu == 1:
    print("各行の和は一致しません!")
else:
    print("各行の和は一致しました")


#重複無しの確認
n = 3
Kakunin = [0] * (n * n)
f_batu = 0
for gyou in range(3):
    for retu in range(3):
        Kakunin[Mahojin[gyou][retu] -1 ] = 1
for i in range(n*n):
    if  Kakunin[i] == 0:
        f_batu = 1
if f_batu == 1:
    print("数の重複がありました")
else:
    print("数の重複はありません")
```

6. プログラム例:あみだくじ

```
#大学入学共通テスト情報関係基礎2022

# 課題1:あみだくじを表示する
def dip_amida(tate, Yokosen, yoko):
    for y in range(tate):
        x = 1
        while( x <= yoko):
            if Yokosen[y] == x:
                print(" ┣┫ ", end = "")
                x = x + 1
            else:
                print(" ┃ ", end = "")
            x = x + 1
        print("")
    return()

tate = 4
yoko = 3
Koma = [1, 2, 3]
Yokosen = [2, 1, 2, 1]
print(Koma)
dip_amida(tate, Yokosen, yoko)

#課題2:あみだくじの結果を求める
for y in range(len(Yokosen)):
    t = Koma[Yokosen[y]]
    Koma[Yokosen[y]] = Koma[Yokosen[y]-1]
    Koma[Yokosen[y]-1] = t
print( Koma)
```

7. プログラム例:すごろくゲーム

```
#大学入学共通テスト情報関係基礎2021-A
import random
Masu = [0, 1, 1, -1, 0.5 , 1, 0, 1, 0, 1, 1, -1, 0.5, 1 -1, 2]

ninzu = 2
owari = 0
r = 0

Koma =  [[0] for i in range(ninzu+1)]
Saikoro = [[0] for i in range(ninzu+1)]
for i in range(1, ninzu+1, 1):
    Koma[i][r] = 1
while(owari == 0):
    r = r + 1
    for i in range(1, ninzu+1, 1):
        Saikoro[i].append(random.randint(1,6))
        k = Koma[i][r-1]
        bairitu = Masu[k]
        if bairitu == 0 and Saikoro[i][r] >= 4:
            bairitu = 1
        idou = int(Saikoro[i][r] * bairitu)
        Koma[i].append(k + idou)
        if Koma[i][r] < 1:
            Koma[i][r] = 1
        if Koma[i][r] >=15:
            Koma[i][r] = 15
            owari =1
print("サイコロの目")
for i in range(1, ninzu+1, 1):
    print(i,"番さん",Saikoro[i])
print("コマの位置")
for i in range(1, ninzu+1, 1):
    print(i,"番さん", Koma[i])
```

8. プログラム例:6つの数字の次に大きな数

```
#電気通信大試作問題2023年11月
#問題外開始
def ureverse(my_list, i, j):
  # iからjまでの要素をスライスで切り出し、逆順にして元のリストに代入
  my_list[i:j+1] = my_list[j:i-1:-1]
#問題外終了

Narabi = [1,2,3,5,4,6]
print("元の数の並び=", Narabi)
ketasu = len(Narabi)
i = ketasu -2
while(i>=0 and Narabi[i] > Narabi[i+1]):
    i= i -1
if(i>=0):
    j = ketasu - 1
    while( i< j and Narabi[i]>Narabi[j]):
        j = j -1
    Narabi[i], Narabi[j] = Narabi[j], Narabi[i]
    ureverse(Narabi, i+1, ketasu-1)
    print("次に大きな数の並び=",Narabi)
else:
    print("最も大きな数の並びです")
```

9. プログラム例:平面でのロボットの移動

```
#大学入学共通テスト情報関係基礎追加2021
Tairu = [[ ".", ".", ".",".",","."],
         [ ".", "R", "R","D","L"],
         [ ".", "R", "D","R","U"],
         [ ".", "U", "D","L","U"]]
tate = 3
yoko = 4
Yonda = [[0 for i in range(yoko+1)] for j in range(tate+1)]
for i in range(tate+1):
    print(Tairu[i])

x = int(input("x="))
y = int(input("y="))

print("現在位置(x,y)",x,y)

owari = 0
while( owari == 0):
    if (x >=1 and x <= yoko) and (y>=1 and y<= tate):
        t = Tairu[y][x]
        Yonda[y][x] = 1
    if t =="U":
        y = y -1
    elif t =="D":
         y = y + 1
    elif t =="L":
         x = x -1
    elif t =="R":
        x = x + 1
    print("現在位置(x,y)",x,y)
    if (x <1 or x > yoko) or (y <1 or y >tate):
        print("壁にぶつかる")
        owari = 1
    elif Yonda[y][x] == 1:
        print("動き続ける")
        owari = 1
```

10. プログラム例:データの圧縮

```
# 京都産業大学試作問題2024年-B
data_pos = 0
Org_Data = [66, 0, 7, 0, 0, 7, 7, 0, 0, 0, 29]
print("圧縮前データ=",Org_Data)
#問題外開始
def eofData():
    global data_pos
    global Org_Data
    ret = 0
    if data_pos == len(Org_Data):
        ret = -1
    return ret
def getData():
    global data_pos
    global Org_Data
    t_data = Org_Data[data_pos]
    data_pos = data_pos + 1
    return t_data
def putData(data):
    print(data, end = " ")
#問題外終了

def putZeros(n):
    count = n
    while(count>156):
        putData(101)
        count = count - 156
    if count > 0:
        if count==1:
            data = 0
        else:
            data =257-count
        putData(data)
    return

print(  "圧縮後データ= ", end = "")
length = 0
while(eofData() == 0):
    data = getData()
    if data == 0:
        length = length + 1
    else:
        if length >0:
            putZeros(length)
            length = 0
        putData(data)
if length > 0:
    putZeros(length)
```

11. プログラム例:ゲームプレイヤーの管理

```
#大学入学共通テスト情報関係基礎2024追加-A
Player = [["小池", 3], ["天野", 0], ["石村", 1], ["渡辺", 4], ["高橋", 2]]

print("Player情報=",Player)
# 課題1:play順に名前を表示する。
p = 0
print("プレイ順番=",end="")
for i in range(5):
    print(Player[p][0], end=", ")
    p =Player[p][1]
print()

# 課題2:新規のプレーヤーの追加
n = len(Player)
Player.append(["三田", 0])
tuika = len(Player)-1
x = 4
Player[tuika][1] = Player[x][1]
Player[x][1] = tuika
n = n +1
print(n, "人　追加後のPlayer情報=",Player)

# 課題3:新規のプレーヤーの削除
# プレーヤーの配列を初期化している。
Player = [["小池", 3], ["天野", 0], ["石村", 1], ["渡辺", 4], ["高橋", 2]]
n = len(Player)
p = 0
nuke = 3
while( Player[p][1] != nuke):
    p = Player[p][1]
Player[p][1] = Player[nuke][1]
n = n -1
print(n, "人　削除後のPlayer情報=",Player)
```

12. プログラム例:選挙のドント表

```
#入試センター試作問題2021年3月
Tomei = ["A党","B党", "C党", "D党"]
Hikaku = [0,0,0,0]
Tokuhyo = [ 1200, 660, 1440, 180]
Tosen = [0,0,0,0]
tosenkei =0
giseki = 6
maxi = 0
for m in range(4):
    Hikaku[m] = Tokuhyo[m]
while tosenkei < giseki:
    max = 0
    for i in range(4):
        if max < Hikaku[i]:
            max = Hikaku[i]
            maxi = i
    Tosen[maxi] = Tosen[maxi] + 1
    tosenkei = tosenkei +1
    Hikaku[maxi] = Tokuhyo[maxi]//Tosen[maxi]+1
for k in range(4):
    print(Tomei[k], ":", Tosen[k], "名“, end=“  “)
```

13. プログラム例:パスワードの総当たり攻撃

```
#大学入学共通テスト情報関係基礎2022追加
# 次の数を求める関数
def nextSu(Suuti):
    t = Suuti[0]+1
    m = len(Suuti)
    n = 26
    Suuti[0] = t % n
    j = 0
    while(j < m and t >=n):
        j = j + 1
        t = Suuti[j] + 1
        Suuti[j] = t % n
    return(Suuti)

Seikai = "abcaaaaa"
Suuti = [0, 0, 0, 0, 0, 0, 0, 0]
Mozi = "abcdefghijklmnopqrstuvwxyz"
#正解の文字列と一致する推定文字列を見つけ出す
print("Seikai=", Seikai)
owari = 0
while(owari == 0):
    c = 0
    Kouho = ""
    for i in range(8):
        Kouho = Kouho + Mozi[Suuti[i]]
        if Seikai[i] == Mozi[Suuti[i]]:
            c = c + 1
    if c == 8:
        print("Suuti=",Suuti)
        print("(文字)=", Seikai)
        for j in range(8):
            print(Mozi[Suuti[j]], end="")
        owari = 1
    else:
        Suuti = nextSu(Suuti)
```

14.プログラム例:講師配置

```
#大学入学共通テスト情報関係基礎2023追加
Hyou =[[9, 9, 9, 9, 9, 9],
       [9, 10, 0, 0, 0, 0],
       [9, 0, 8, 0, 0, 0],
       [9, 0, 0, 19, 0, 0],
       [9, 0, 0, 0, 14, 0],
       [9, 0, 0, 0, 0, 7]]

jigensu = 5
for renzoku in range(5,1,-1):
    hajime = 1
    owari = renzoku
    while(owari <= jigensu):
        saisyou = Hyou[hajime][hajime]
        for i in range(hajime, owari+1, 1):
            if saisyou > Hyou[i][i]:
                saisyou = Hyou[i][i]
        if saisyou > 0:
            Hyou[hajime][owari] = saisyou
            for i in range(hajime, owari+1, 1):
                Hyou[i][i] = Hyou[i][i] - saisyou
        hajime = hajime +1
        owari = owari +1
for i in range(6):
    print(Hyou[i])
```