情報処理（12週目）
Pythonプログラミング基礎

王 忠奎
立命館大学　ロボティクス学科

11週目レポート

- 割り勘ツールをつくろう．金額と，人数，割引券をキーボードから入力し，1 人あたり何円払うかと，その上で何円不足するかを表示しよう．例えば，13547 円を4 人で払い，全体で1000 円分の割引券を持っていたとすると，1 人3136 円払って3 円不足する．

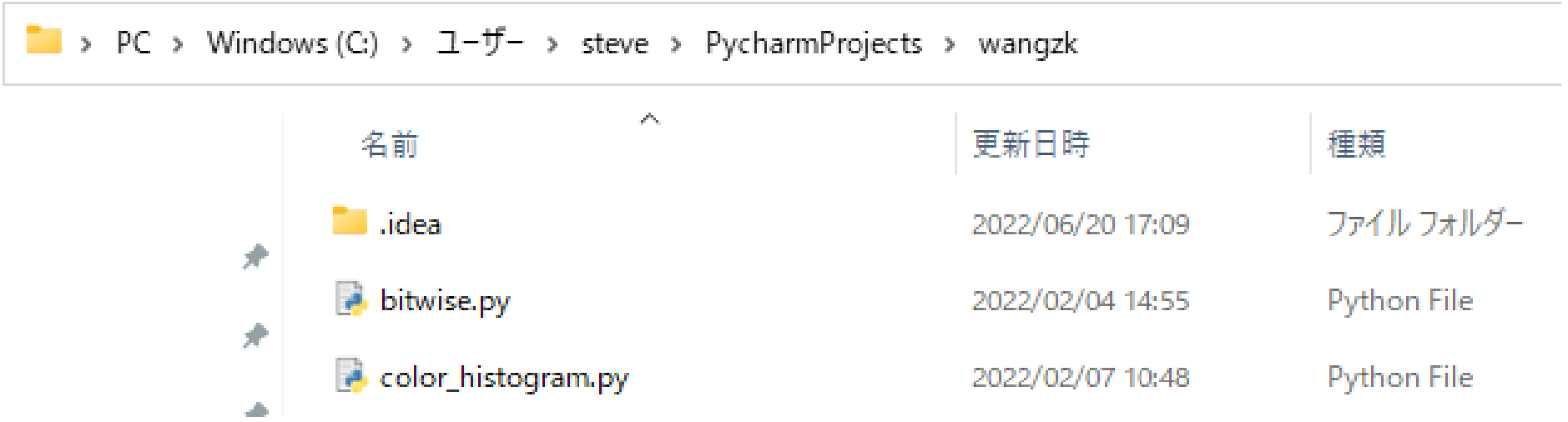
```
bill = int(input('金額をご入力ください: '))
people = int(input('人数をご入力ください: '))
discount = int(input('割引券をご入力ください: '))

payment_each = (bill - discount) // people  # 商の整数部の取得
remainder = (bill - discount) % people  # あまりの取得

print(f'一人当たり払う金額: {payment_each}円')
print(f'不足する分: {remainder}円')
```

プログラムファイルの保存場所と実行方法

デフォルトの
保存場所



実行方法

1．PyCharmでの実行
2．コマンドプロンプトでの実行

```
2022/01/27  11:27        31,371,084 video1_rescale.mp4
              40 個のファイル          76,397,226 バイト
               3 個のディレクトリ  613,896,740,864 バイトの空き領域

C:¥Users¥steve¥PycharmProjects¥wangzk>python report1
```

講義の流れ

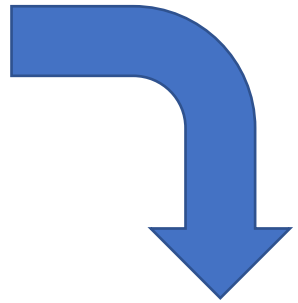
- リスト
- ディクショナリ（辞書）
- 文字列
- 分岐（if文）
- 繰り返し（while文、for文）
- 関数
- レポート

リスト

任意の型（整数、浮動小数点数、文字列など）のデータを格納できる。要素には**順序があり**、インデックスを用いて要素を指定できる。リストの要素は**変更可能**。他のプログラミング言語における「配列」的な使い方をする

```
num_list = [10, 20, 30]  # 数字のリスト
color_list = ['Red', 'Green', 'Blue']  # 文字列のリスト
mix_list = [num_list, color_list]  # リストの連結


print(mix_list)
print(num_list * 3)
```



```
[[10, 20, 30], ['Red', 'Green', 'Blue']]
[10, 20, 30, 10, 20, 30, 10, 20, 30]
```

リスト要素の取得

```
color_list = ['Red', 'Green', 'Blue']
print(color_list[0])  # 1番目の要素
print(color_list[1])  # 2番目の要素
print(color_list[-1])  # 最後の要素
print(color_list[-2])  # 後ろから2番目要素
print(len(color_list))  # リストの長さ
```

```
Red
Green
Blue
Green
3
```

リストの一部を取り出す

リスト[最初ID：最後ID]

最初ID≦取得＜最後ID]

最後IDの要素が含まれない

```
mylist = ['A', 'B', 'C', 'D', 'E']
print(f'mylist[1:4] = {mylist[1:4]}')
print(f'mylist[0:3] = {mylist[0:3]}')
print(f'mylist[:4] = {mylist[:4]}')
print(f'mylist[2:] = {mylist[2:]}')
print(f'mylist[:] = {mylist[:]}')
```

```
mylist[1:4] = ['B', 'C', 'D']
mylist[0:3] = ['A', 'B', 'C']
mylist[:4] = ['A', 'B', 'C', 'D']
mylist[2:] = ['C', 'D', 'E']
mylist[:] = ['A', 'B', 'C', 'D', 'E']
```

要素の入れ替え

```
mylist = ['Apple', 'Peach', 'Orange']
print(f'mylist = {mylist}')

mylist[1] = 'Grape'  # 2番目の要素を変更
print(f'mylist = {mylist}')
```

```
mylist = ['Apple', 'Peach', 'Orange']
mylist = ['Apple', 'Grape', 'Orange']
```

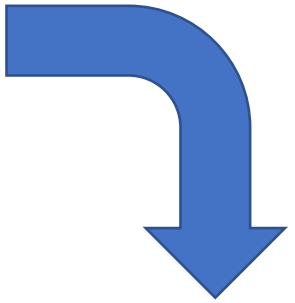
要素の削除、追加、挿入

```
mylist = ['A', 'B', 'C', 'D']
del mylist[1]  # 2番目の要素を削除
print(mylist)

mylist.append('D')  # 最後に要素の1個を追加
print(mylist)

mylist.extend(['D', 'E'])  # 最後に複数の要素を追加
print(mylist)

mylist.insert(2,'F')  # 3番目のところ要素を挿入
print(mylist)
```



```
['A', 'C', 'D']
['A', 'C', 'D', 'D']
['A', 'C', 'D', 'D', 'D', 'E']
['A', 'C', 'F', 'D', 'D', 'D', 'E']
```

ディクショナリ（辞書）

辞書型は { から } までの間に複数の要素をカンマ(,)で区切って定義します。要素はキーと対応する値の組み合わせを キー:値 の形式で記述します。マッピング型なので要素には順序はありません。

```
mydict = {1: 'Orange', 2: 'Lemon', 3: 'Peach'}
print(mydict)
```

```
{1: 'Orange', 2: 'Lemon', 3: 'Peach'}
```

値の取得

インデックスは存在せず、スライスも使用できません。その代わりに**キーを指定**して対応するオブジェクトを取得することができます。

```
mydict = {'JP':'Japan', 'DE':'Germany', 'FR':'France'}


print(f'Key: JP, Val: {mydict["JP"]}')
print(f'Key: FR, Val: {mydict["FR"]}')


print(f'Key: JP, Val: {mydict.get("JP", "NotFound")}')
print(f'Key: EN, Val: {mydict.get("EN", "NotFound")}')
```

```
Key: JP, Val: Japan
Key: FR, Val: France
Key: JP, Val: Japan
Key: EN, Val: NotFound
```

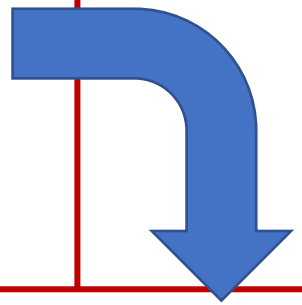
getメソッドを利用

要素の変更、追加、更新

```
my_hobby = {'one': 'Travel', 'two': 'Foods', 'three': 'Movie'}
my_hobby['three'] = 'Sports'  # キー'three'の値を変更
print(my_hobby)

my_hobby = {1: 'Travel', 2: 'Foods', 3: 'Movie'}
my_hobby[4] = 'Sports'  # 要素を追加
print(my_hobby)

mydict = {1:"A", 2:"B", 3:"C"}
otherdict = {2:"b", 4:"d"}
mydict.update(otherdict)  # 辞書を更新
print(mydict)
```



```
{'one': 'Travel', 'two': 'Foods', 'three': 'Sports'}
{1: 'Travel', 2: 'Foods', 3: 'Movie', 4: 'Sports'}
{1: 'A', 2: 'b', 3: 'C', 4: 'd'}
```

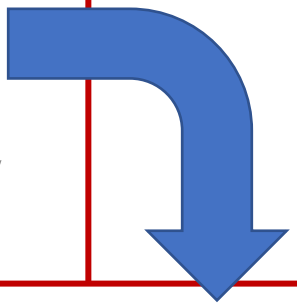
辞書の長さと要素の削除

```
colordict = {1: 'blue', 2: 'red', 3: 'green', 4: 'white'}
print(f'Number of element is {len(colordict)}.')  # 長さを取り出す

del colordict[2]  # キー'2'の内容を削除
print(colordict)

val = colordict.pop(3, 'Not Found')  # キー'3'の内容を取り出す
print(val)  # 取り出した内容を表示
print(colordict)  # 残る辞書を表示

colordict.clear()  # すべての内容を削除
print(colordict)
```



```
Number of element is 4.
{1: 'blue', 3: 'green', 4: 'white'}
green
{1: 'blue', 4: 'white'}
{}
```

キーの確認

in と not in演算子を使用します

```
mydict = {'L': 'Lemon', 'O': 'Orange', 'G': 'Grapes'}
print('O' in mydict)
print('P' in mydict)
print('O' not in mydict)
print('P' not in mydict)
```

```
True
False
False
True
```

文字列

- プログラムの中で文字列を記述する場合、複数の文字の集まりをダブルクオーテーション(")で囲うかシングルクオーテーション(')で囲います。
- 文字列を構成する文字はアルファベットや数値、空白、日本語などの全角文字でも大丈夫です。
- ダブルクオーテーションとシングルクオーテーションで囲った場合で違いはありません。

```
print('Hello 2022, こんにちは')
print("Hello 2022, こんにちは")


print("Tom's toy")
```

```
Hello 2022, こんにちは
Hello 2022, こんにちは
Tom's toy
```

文字列の連結、繰り返す、長さ取得、要素取得

```
print('ABC' + 'DEF' + 'GHI')  # 文字列の連結

print("My age is " + str(18))

print('ABC' * 4)  # 文字列の繰り返す

string = 'Hello'

print(f'{string} の文字数は ' + str(len(string)))

print(string[1])
```

```
ABCDEFGHI
My age is 18
ABCABCABCABC
Hello の文字数は 5
e
```

大文字と小文字の操作

```
print('Hello'.lower())  # すべて小文字に変換
print('APPLE'.lower())

print('Hello'.upper())  # すべて大文字に変換
print('apple'.upper())

print('hello python'.capitalize())  # 最初の文字を大文字に
print('hello python'.title())  # 各単語に最初の文字を大文字に

print('apple'.islower())  # すべての文字が小文字の判定
print('Apple'.isupper())  # すべでの文字が大文字の判定
```

```
hello
apple
HELLO
APPLE
Hello python
Hello Python
True
False
```

文字列の分割と置換

```
print('A B C D E'.split())  # デフォルトは空白で分割

print('A,B,C,D,E'.split(','))  # 指定した区切り文字で分割

print('A B C D E'.split(' ', 3))  # 指定した分割回数で分割

print('A B C D E'.split(' ', 4))

print('Copyright 2018'.replace('2018', '2022'))  # 文字の置換
```

```
['A', 'B', 'C', 'D', 'E']
['A', 'B', 'C', 'D', 'E']
['A', 'B', 'C', 'D E']
['A', 'B', 'C', 'D', 'E']
Copyright 2022
```

if文

```
if 条件式:
    条件式が真の時に実行する文1
    条件式が真の時に実行する文2
    条件式が真の時に実行する文3
```

インデント

```
if 条件式:
    条件式が真の時に実行する文
    ...
else:
    条件式が偽の時に実行する文
    ...
```

※ Python ではブロック（複数の文）をインデントを使って定義します。

if文

インデント毎にスペースの4つを使用すること

```
if 条件式1:
    条件式1が真の時に実行する文
    ...
elif 条件式2:
    条件式1が偽で条件式2が真の時に実行する文
    ...
elif 条件式3:
    条件式1及び条件式2が偽で条件式3が真の時に実行する文
    ...
else:
    すべての条件式が偽のときに実行する文
    ...
```

```
if 条件式1:
    条件式1が真の時に実行する文
    ...
else:
    if 条件式2:
        条件式1が偽で条件式2が真の時に実行する文
        ...
    else:
        if 条件式3:
            条件式1及び条件式2が偽で条件式3が真の時に実行する文
            ...
        else:
            すべての条件式が偽のときに実行する文
            ...
```

複数の条件式を使った条件分岐(if...elif...else)

比較演算子

	`x == y`	x と y が等しい
	`X != y`	x と y が等しくない
	`x > y`	x は y よりも大きい
	`x < y`	x は y よりも小さい
	`x >= y`	x は y と等しいか大きい
	`x <= y`	x は y と等しいか小さい
★	`x in y`	x という要素 が y に存在する
★	`x not in y`	x という要素 が y に存在しない

論理演算子

x or y	x または y の少なくともどちらか 1 つが True なら True
	それ以外は False
x and y	x と y がどちらも True なら True
	それ以外は False
not x	x が True なら False、 x が False なら True

プログラム例－－－肥満判定

```
h = float(input('身長[m]をご入力ください：　'))
w = float(input('体重[kg]をご入力ください：　'))

x = w / h / h

if x < 18.5:
    print(f'あなたのBMIは{x:.3f}で、痩せています。')
elif x < 25:
    print(f'あなたのBMIは{x:.3f}で、普通です。')
else:
    print(f'あなたのBMIは{x:.3f}で、肥満です。')

print('The end')
```

BMI の値を x，身長を h[m]，体重を w[kg] すると，

$$x = \frac{w}{h^2}$$

```
身長[m]をご入力ください：　1.76
体重[kg]をご入力ください：　82.5
あなたのBMIは26.634で、肥満です。
The end
```

while文

while 文は指定した条件式が真の間、処理を繰り返し実行します。

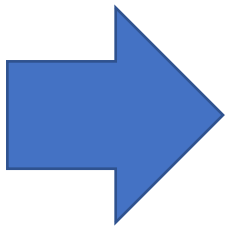
```
while 条件式:
    条件式が真の時に実行する文1
    条件式が真の時に実行する文2
    条件式が真の時に実行する文3
```

```
while 条件式:
    条件式が真の時に実行する文1
    条件式が真の時に実行する文2
else:
    条件式が偽の時に実行する文1
    条件式が偽の時に実行する文2
```

プログラム例

```
num = 1
total = 0
print("Start")

while num < 6:
    print("num = " + str(num))
    total += num
    num += 1
else:
    print("Total = " + str(total))
```



```
Start
num = 1
num = 2
num = 3
num = 4
num = 5
Total = 15
```

for文

for 文は別途指定したイテラブルなオブジェクトの要素の数だけ要素を１つずつ取り出しながら繰り返しを行います。

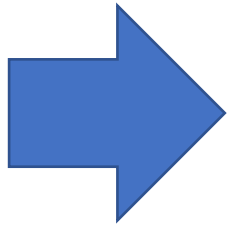
```
for 変数 in イテラブルオブジェクト:
    実行する文1
    実行する文2
    実行する文3
```

```
for 変数 in イテラブルオブジェクト:
    実行する文1
    実行する文2
else:
    実行する文1
    実行する文2
```

イテラブルオブジェクト：文字列、リスト、タプル、辞書など

プログラム例

```
count = 0
mylist = ["Orange", "Peach", "Lemon", "Apple"]
for val in mylist:
    print("value:" + val)
    count += 1
else:
    print("要素の数 = " + str(count))
```

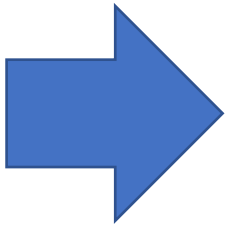


```
value:Orange
value:Peach
value:Lemon
value:Apple
要素の数 = 4
```

range関数で指定した回数で繰り返す

```
r = range(8)
print(list(r))

for i in r:
    print(f'2の{i}乗は：　{2**i}')
```



```
2の0乗は：　1
2の1乗は：　2
2の2乗は：　4
2の3乗は：　8
2の4乗は：　16
2の5乗は：　32
2の6乗は：　64
2の7乗は：　128
```

break文で繰り返し処理の強制終了

while 文や for 文の繰り返し処理の中で break 文を使用すると繰り替えし処理の強制終了を行うことができます。

```
num = 1

while True:
    print(f'num = {num}')
    num *= 2

    if num > 10:
        break

print('The End')
```

=

```
num = 1

for i in range(10):
    print(f'num = {num}')
    num *= 2

    if num > 10:
        break

print('The End')
```

→

```
num = 1
num = 2
num = 4
num = 8
The End
```

関数

関数の定義

```
def 関数名(引数1, 引数2, ...):
    関数内で実行する処理1
    関数内で実行する処理2
    関数内で実行する処理3


    return 戻り値
```

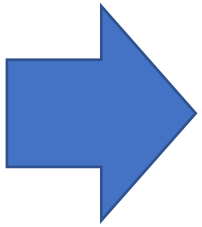
命名ルール

1. 使用できる文字はａ～ｚ、Ａ～Ｚ、０～９、アンダーバー(_)、漢字など
2. 一文字目に数値(0～9)は使用できない
3. 一文字目にアンダーバーは使用できるが、通常は使用しない方がいい
4. 大文字と小文字は区別される
5. 予約語は使用できない

関数例

```
def myfunc(str1, num1):
    print(f'Name: {str1}, Age: {num1}')


myfunc('Tanaka', 23)
myfunc('Yamada', 28)
```



```
Name: Tanaka, Age: 23
Name: Yamada, Age: 28
```

```
def average(val1, val2):
    return (val1 + val2) / 2.0


print(f'6と4の平均は {average(6, 4)}')
print(f'8.5と7.5の平均は {average(8.5, 7.5)}')
```

```
6と4の平均は 5.0
8.5と7.5の平均は 8.0
```

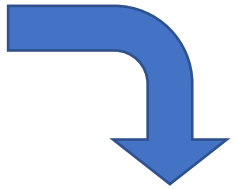
引数にデフォルト値の設定

注意点：デフォルト値のない引数がデフォルト値のある引数よりも後に記述することはできません。

```
def myfunc2(num2, str2 = '未入力', str3 = '不明'):
    print(f'年齢は{num2}です。', end = '')
    print(f'名前は{str2}です。', end = '')
    print(f'住所は{str3}です。')



myfunc2(28, 'Yamada', 'Kusatsu')
myfunc2(23, 'Tanaka')
myfunc2(19)
```



```
年齢は28です。名前はYamadaです。住所はKusatsuです。
年齢は23です。名前はTanakaです。住所は不明です。
年齢は19です。名前は未入力です。住所は不明です。
```

レポート

自然対数の底 $e = 2.7182818\cdots$ は，以下の式で求めることができる．

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \cdots + \frac{1}{n!}$$

上式を使って，$n = 10$ のときの e を求めてみよう．

ここで，！は階乗を表し，$n! = 1 \times 2 \times 3 \times \cdots \times n$ である．

- 関数を作って、階乗を計算してみよう
- 「report2_name.py」で保存して、manaba+Rで提出してください