memo

プログラミング演習 I

授業開始までしばらくお待ちください。

プログラミング演習 I
Exercise on Programming I

本スライドは Teams で配布しています。

小林裕之・瀬尾昌孝
prog1@oitech.ddns.net

大阪工業大学 RD 学部システムデザイン工学科
OSAKA INSTITUTE OF TECHNOLOGY

13 of 14

a LaTeX + Beamer slideshow

授業前にこれを見ている人へ。

- 内容を先に読んで予習しておくことは大いに結構なことだと思います。ぜひ予習してください。
- 内容は随時更新しています。授業前に再度最新版を確認してください。
- 課題の提出 (Google Forms によるもの) は **授業当日の指示があるまで行わないように** してください。

次回予告

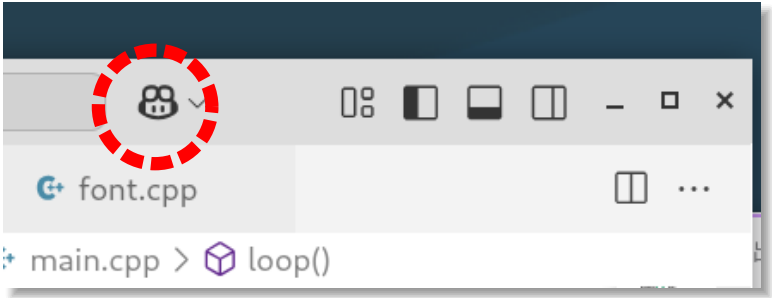
次回 (第 14 回) は期末演習をやります。

成績上、100 点満点中 19 点に相当する《期末演習》をやります。《期末演習》の欠席は単位修得の欠格事由とします。

- 座席は事前に e-mail で通知する指定席 (いつもどおり)。
- 4 ないし 5 問程度の課題を時間内に提出。
- ノート参考書等、電子機器でないものは何でも持ち込み可。
- 電子機器は PC1 台のみ。スマートフォン等は禁止。
- PC を用いた**検索・サイト閲覧・AI 使用等は全面不可**。ネットの利用は課題提出および本科目の Teams の閲覧のみ可。
- VScode の **copilot 等は停止**しておくこと。(次ページ参照)
- 検索・AI などの不正は **41 点の減点**。

Copilot 禁止

Copilot (および類似の AI 拡張機能) が有効化されている者は必ず無効化もしくはアンインストールしておくこと。試験当日に有効化してあったことが確認された場合は理由を問わず不正行為とみなして 41 点の減点とする。



スライス

ㄱㅅㄷㅅㄱㅅ

リストを斬る

スライス

リストや文字列などに対して以下のような**スライス**が使える。

`lst[i:j]`

`lst`の i 番目から j-1 番目までを抜き出したもの。**`lst`**がリストならそのスライスもリスト、文字列ならそのスライスは文字列。

```
nums = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5]
print(nums[3:6])    # [1, 5, 9]
print(nums[0:1])    # [3]
print(nums[5:5])    # []
print("A quick brown fox jumps over the lazy dog."[2:7]) # quick
```

特殊なよく使うスライス

`lst`の`i` 番目から最後までは`lst[i:]`と書ける。

head と tail (そういう名前の関数があるという意味ではなく、プログラミング界の一般名詞。)

Python に限った話ではなく、プログラミングで広く使われるリストの頭としっぽを head と tail と言う。

- **head**: リストの先頭「**要素**」のこと。Python の場合は　　で取得できる。
- **tail**: リストの先頭以外の残り全部の「**リスト**」のこと。Python の場合は　　で取得できる。

例: 前ページの`nums`の場合、head は　　で、tail は　　。

特殊なよく使うスライス

`lst`の`i` 番目から最後までは`lst[i:]`と書ける。

head と tail (そういう名前の関数があるという意味ではなく、プログラミング界の一般名詞。)

Python に限った話ではなく、プログラミングで広く使われるリストの頭としっぽを head と tail と言う。

- **head**: リストの先頭「**要素**」のこと。Python の場合は `lst[0]` で取得できる。
- **tail**: リストの先頭以外の残り全部の「**リスト**」のこと。Python の場合は で取得できる。

例: 前ページの`nums`の場合、head は で、tail は

。

特殊なよく使うスライス

`lst`の`i` 番目から最後までは`lst[i:]`と書ける。

head と tail (そういう名前の関数があるという意味ではなく、プログラミング界の一般名詞。)

Python に限った話ではなく、プログラミングで広く使われるリストの頭としっぽを head と tail と言う。

- **head**: リストの先頭「**要素**」のこと。Python の場合は `lst[0]` で取得できる。
- **tail**: リストの先頭以外の残り全部の「**リスト**」のこと。Python の場合は`lst[1:]` で取得できる。

例: 前ページの`nums`の場合、head は　　で、tail は

。

特殊なよく使うスライス

`lst`の`i` 番目から最後までは`lst[i:]`と書ける。

head と tail (そういう名前の関数があるという意味ではなく、プログラミング界の一般名詞。)

Python に限った話ではなく、プログラミングで広く使われるリストの頭としっぽを head と tail と言う。

- **head**: リストの先頭「**要素**」のこと。Python の場合は `lst[0]` で取得できる。
- **tail**: リストの先頭以外の残り全部の「**リスト**」のこと。Python の場合は`lst[1:]` で取得できる。

例: 前ページの`nums`の場合、head は `3` という 1 つの**整数値**で、tail は

。

特殊なよく使うスライス

`lst`の`i` 番目から最後までは`lst[i:]`と書ける。

head と tail (そういう名前の関数があるという意味ではなく、プログラミング界の一般名詞。)

Python に限った話ではなく、プログラミングで広く使われるリストの頭としっぽを head と tail と言う。

- **head**: リストの先頭「**要素**」のこと。Python の場合は `lst[0]` で取得できる。
- **tail**: リストの先頭以外の残り全部の「**リスト**」のこと。Python の場合は`lst[1:]` で取得できる。

例: 前ページの`nums`の場合、head は `3` という 1 つの**整数値**で、tail は `[1, 4, 1, 6, 9, 2, 6, 5, 3, 5]` という**リスト**。

復習とスライスの実践

問. 整数のリストを引数として受け取り、『先頭要素以下のものを集めたリスト**left**』および『先頭要素より大きなものを集めたリスト**right**』を作って表示する関数**qs()**を定義し、以下のプログラム**qst.py**を完成させよ。ただし、先頭要素自体は**left**にも**right**にも入れないこと。

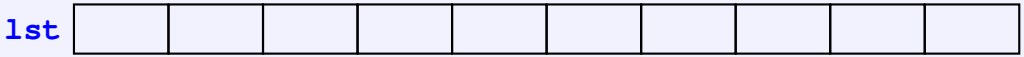
```
def qs(xs):
    # この部分を書く
    print(left)    # -> 1, 1, 2, 3
    print(right)   # -> 4, 5, 9, 6, 5, 3, 5, 8, 9, 7, 9

lst = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5, 8, 9, 7, 9]
qs(lst)
```

再帰

リストの最大値とは

- 整数リストの中から最大値を求めたい。
- 求め方の「手順」を考えず、「定義」を与えてみよう。こんなのはどう？

リスト `lst` の最大値とは（定義）



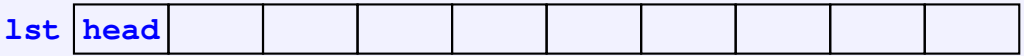
1. **`lst`**を「先頭要素**`head`**」と「残りのリスト**`tail`**」に分ける。
2. **`lst`**の要素数が 1 なら、**`head`**が**`lst`**の最大値で確定、終了。
3. 「**`tail`**の**最大値**」を**`x`**とする。
4. もし**`head`**>**`x`**ならば**`head`**が**`lst`**の最大値である。
5. そうじゃなければ**`x`**が**`lst`**の最大値である。

変な定義だけどウソじゃないように思えるので実装してみよう。

リストの最大値とは

- 整数リストの中から最大値を求めたい。
- 求め方の「手順」を考えず、「定義」を与えてみよう。こんなのはどう？

リスト `lst` の最大値とは（定義）



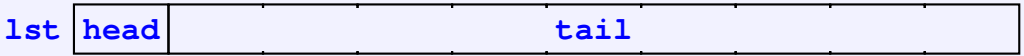
1. **`lst`**を「先頭要素**`head`**」と「残りのリスト**`tail`**」に分ける。
2. **`lst`**の要素数が 1 なら、**`head`**が**`lst`**の最大値で確定、終了。
3. 「**`tail`**の**最大値**」を**`x`**とする。
4. もし**`head`**>**`x`**ならば**`head`**が**`lst`**の最大値である。
5. そうじゃなければ**`x`**が**`lst`**の最大値である。

変な定義だけどウソじゃないように思えるので実装してみよう。

リストの最大値とは

- 整数リストの中から最大値を求めたい。
- 求め方の「手順」を考えず、「定義」を与えてみよう。こんなのはどう？

リスト `lst` の最大値とは（定義）



1. **`lst`**を「先頭要素**`head`**」と「残りのリスト**`tail`**」に分ける。
2. **`lst`**の要素数が 1 なら、**`head`**が**`lst`**の最大値で確定、終了。
3. 「**`tail`**の**最大値**」を**`x`**とする。
4. もし**`head`**>**`x`**ならば**`head`**が**`lst`**の最大値である。
5. そうじゃなければ**`x`**が**`lst`**の最大値である。

変な定義だけどウソじゃないように思えるので実装してみよう。

練習

問. 前ページの定義通りに「リストの最大値」を返す関数**mymax()**を実装して、以下のプログラム**recmax.py**を完成させよ。

```
def mymax(lst):
    head = lst[0]    # 前ページの手順 1
    tail = lst[1:]   # 前ページの手順 1
    # 続きを書こう

nums = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5, 8, 9, 7, 9]
print(mymax(nums))
```

ソートを再帰的に定義する

「ソートしたリスト」の定義

1. もしリスト長が 1 以下ならソートの必要なし。それ自体が**ソートしたリスト**である。
2. head を基準値とする。
3. 基準値を真ん中に置き、「基準値以下の要素」と「基準値より大きい要素」を左右に分ける。
4. 左右に分けたものをそれぞれ「ソートしたリスト」にしてつなげたものが全体を**ソートしたリスト**である。

3	1	4	7	8	6	2	0

ソートを再帰的に定義する

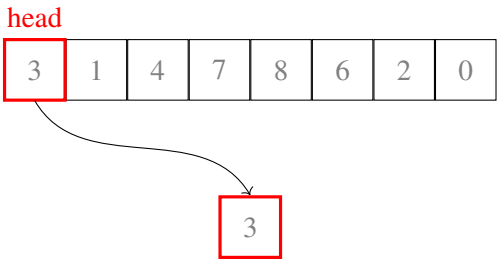
「ソートしたリスト」の定義

1. もしリスト長が 1 以下ならソートの必要なし。それ自体が**ソートしたリスト**である。
2. head を基準値とする。
3. 基準値を真ん中に置き、「基準値以下の要素」と「基準値より大きい要素」を左右に分ける。
4. 左右に分けたものをそれぞれ「ソートしたリスト」にしてつなげたものが全体を**ソートしたリスト**である。

ソートを再帰的に定義する

「ソートしたリスト」の定義

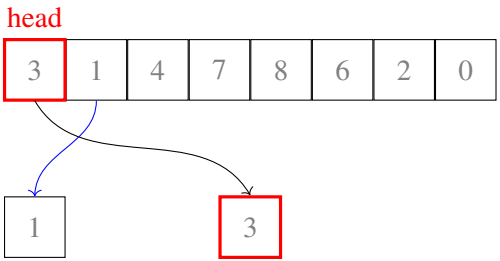
1. もしリスト長が 1 以下ならソートの必要なし。それ自体が**ソートしたリスト**である。
2. head を基準値とする。
3. 基準値を真ん中に置き、「基準値以下の要素」と「基準値より大きい要素」を左右に分ける。
4. 左右に分けたものをそれぞれ「ソートしたリスト」にしてつなげたものが全体を**ソートしたリスト**である。



ソートを再帰的に定義する

「ソートしたリスト」の定義

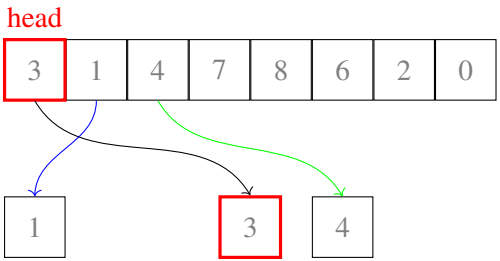
1. もしリスト長が 1 以下ならソートの必要なし。それ自体が**ソートしたリスト**である。
2. head を基準値とする。
3. 基準値を真ん中に置き、「基準値以下の要素」と「基準値より大きい要素」を左右に分ける。
4. 左右に分けたものをそれぞれ「ソートしたリスト」にしてつなげたものが全体を**ソートしたリスト**である。



ソートを再帰的に定義する

「ソートしたリスト」の定義

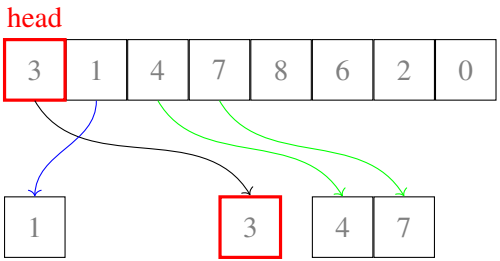
1. もしリスト長が 1 以下ならソートの必要なし。それ自体が**ソートしたリスト**である。
2. head を基準値とする。
3. 基準値を真ん中に置き、「基準値以下の要素」と「基準値より大きい要素」を左右に分ける。
4. 左右に分けたものをそれぞれ「ソートしたリスト」にしてつなげたものが全体を**ソートしたリスト**である。



ソートを再帰的に定義する

「ソートしたリスト」の定義

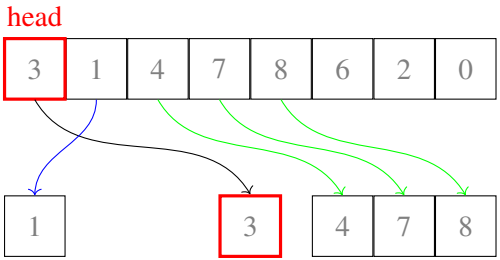
1. もしリスト長が 1 以下ならソートの必要なし。それ自体が**ソートしたリスト**である。
2. head を基準値とする。
3. 基準値を真ん中に置き、「基準値以下の要素」と「基準値より大きい要素」を左右に分ける。
4. 左右に分けたものをそれぞれ「ソートしたリスト」にしてつなげたものが全体を**ソートしたリスト**である。



ソートを再帰的に定義する

「ソートしたリスト」の定義

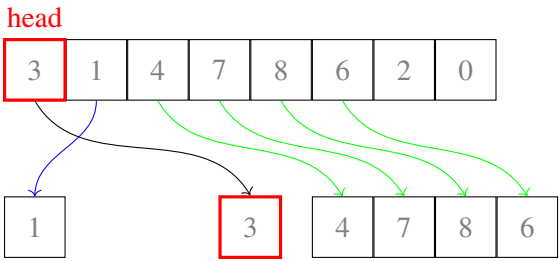
1. もしリスト長が 1 以下ならソートの必要なし。それ自体が**ソートしたリスト**である。
2. head を基準値とする。
3. 基準値を真ん中に置き、「基準値以下の要素」と「基準値より大きい要素」を左右に分ける。
4. 左右に分けたものをそれぞれ「ソートしたリスト」にしてつなげたものが全体を**ソートしたリスト**である。



ソートを再帰的に定義する

「ソートしたリスト」の定義

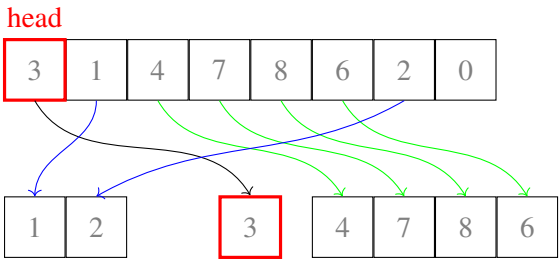
1. もしリスト長が 1 以下ならソートの必要なし。それ自体が**ソートしたリスト**である。
2. head を基準値とする。
3. 基準値を真ん中に置き、「基準値以下の要素」と「基準値より大きい要素」を左右に分ける。
4. 左右に分けたものをそれぞれ「ソートしたリスト」にしてつなげたものが全体を**ソートしたリスト**である。



ソートを再帰的に定義する

「ソートしたリスト」の定義

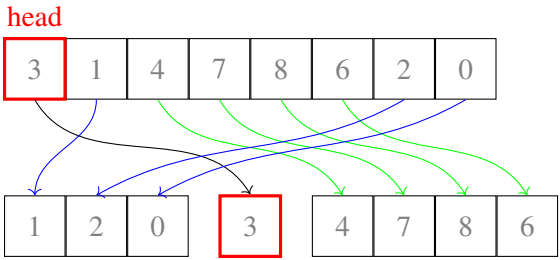
1. もしリスト長が 1 以下ならソートの必要なし。それ自体が**ソートしたリスト**である。
2. head を基準値とする。
3. 基準値を真ん中に置き、「基準値以下の要素」と「基準値より大きい要素」を左右に分ける。
4. 左右に分けたものをそれぞれ「ソートしたリスト」にしてつなげたものが全体を**ソートしたリスト**である。



ソートを再帰的に定義する

「ソートしたリスト」の定義

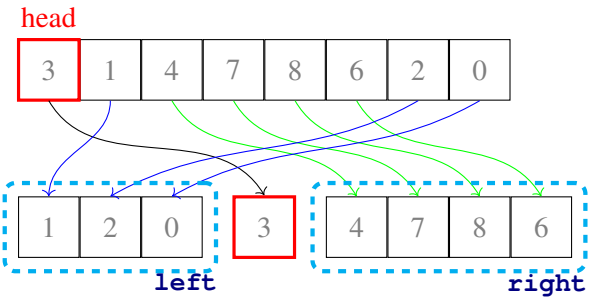
1. もしリスト長が 1 以下ならソートの必要なし。それ自体が**ソートしたリスト**である。
2. head を基準値とする。
3. 基準値を真ん中に置き、「基準値以下の要素」と「基準値より大きい要素」を左右に分ける。
4. 左右に分けたものをそれぞれ「ソートしたリスト」にしてつなげたものが全体を**ソートしたリスト**である。



ソートを再帰的に定義する

「ソートしたリスト」の定義

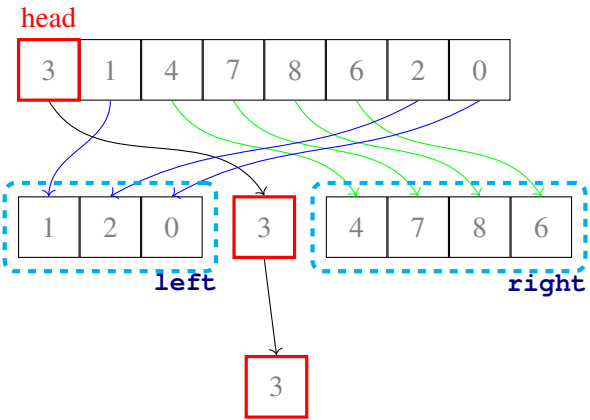
1. もしリスト長が 1 以下ならソートの必要なし。それ自体が**ソートしたリスト**である。
2. head を基準値とする。
3. 基準値を真ん中に置き、「基準値以下の要素」と「基準値より大きい要素」を左右に分ける。
4. 左右に分けたものをそれぞれ「ソートしたリスト」にしてつなげたものが全体を**ソートしたリスト**である。



ソートを再帰的に定義する

「ソートしたリスト」の定義

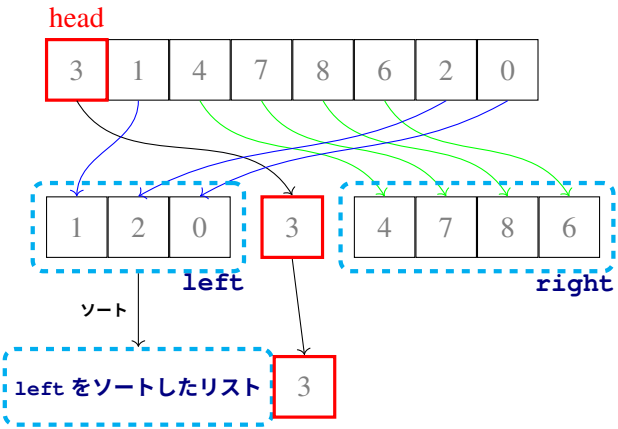
1. もしリスト長が 1 以下ならソートの必要なし。それ自体が**ソートしたリスト**である。
2. head を基準値とする。
3. 基準値を真ん中に置き、「基準値以下の要素」と「基準値より大きい要素」を左右に分ける。
4. 左右に分けたものをそれぞれ「ソートしたリスト」にしてつなげたものが全体を**ソートしたリスト**である。



ソートを再帰的に定義する

「ソートしたリスト」の定義

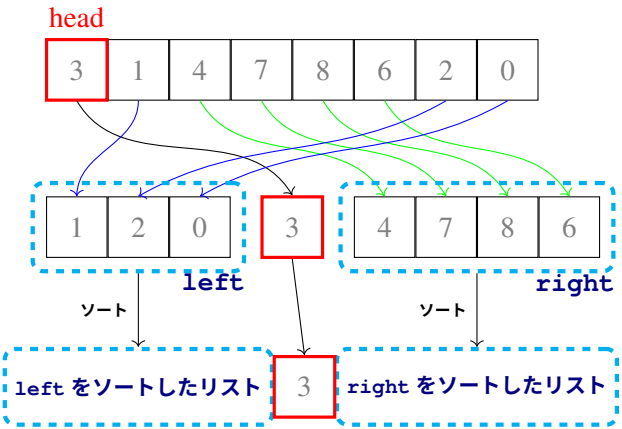
1. もしリスト長が 1 以下ならソートの必要なし。それ自体が**ソートしたリスト**である。
2. head を基準値とする。
3. 基準値を真ん中に置き、「基準値以下の要素」と「基準値より大きい要素」を左右に分ける。
4. 左右に分けたものをそれぞれ「ソートしたリスト」にしてつなげたものが全体を**ソートしたリスト**である。



ソートを再帰的に定義する

「ソートしたリスト」の定義

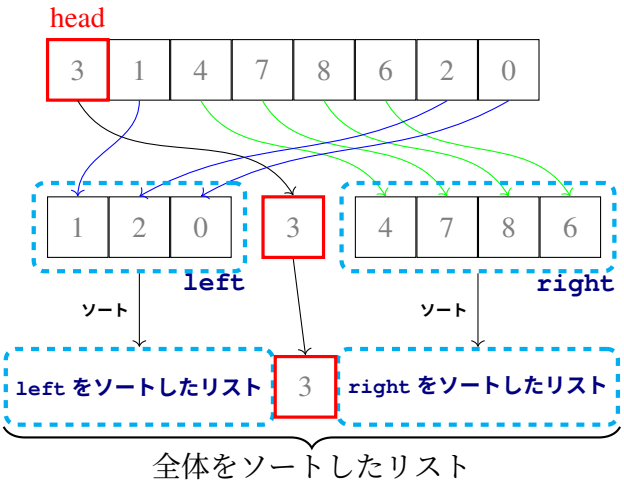
1. もしリスト長が 1 以下ならソートの必要なし。それ自体が**ソートしたリスト**である。
2. head を基準値とする。
3. 基準値を真ん中に置き、「基準値以下の要素」と「基準値より大きい要素」を左右に分ける。
4. 左右に分けたものをそれぞれ「ソートしたリスト」にしてつなげたものが全体を**ソートしたリスト**である。



ソートを再帰的に定義する

「ソートしたリスト」の定義

1. もしリスト長が 1 以下ならソートの必要なし。それ自体が**ソートしたリスト**である。
2. head を基準値とする。
3. 基準値を真ん中に置き、「基準値以下の要素」と「基準値より大きい要素」を左右に分ける。
4. 左右に分けたものをそれぞれ「ソートしたリスト」にしてつなげたものが全体を**ソートしたリスト**である。



実は Quick SorT の qs(), qst.py だった！

問. さっき作った**qst.py**を改変し、前ページの手順でクイックソートを行うように（以下のプログラムを完成）せよ。

```
def qs(xs):          # リストを与えるとソート済みリストを返す関数
    # この部分を書く

lst = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5, 8, 9, 7, 9]
print(lst)           # before
print(qs(lst))       # after
```

練習問題

paiza 練習問題

「チャレンジする言語」を Python3 に設定する。

- 問題集 > C ランク・スキルチェック過去問題セット > 野球の審判
- 問題集 > C ランク・スキルチェック過去問題セット > 宝くじ

練習問題: ドリル計算機 (2024 期末演習より)

1〜9 の数と **+**, **−**, *****, **/** をランダムに使った計算問題を出力する。それに対して正しい答えを入力すると “correct”、そうでない場合は “incorrect” を出力する。正しい答えが n 回連続した時点で “bye” と出力して終了する。$n = 1$ なら 3 点、$n = 3$ なら 4 点。ただし割り算は（余りや小数を用いず）商を答えるのみとすること。数値は入力も出力もすべて整数とする。

実行例

```
4*7                 # output
28                  # input
correct             # output
3*5                 # output
12                  # input
incorrect           # output
6-4                 # output
2                   # input
correct             # output
9+9                 # output
18                  # input
correct             # output
4/6                 # output
0                   # input
correct             # output
bye                 # output
```

授業アンケート

UNIPA からお願いします。

練習問題解答

recmax.py

```python
def mymax(lst):
    head = lst[0]
    tail = lst[1:]
    if len(lst) == 1:
        return head
    x = mymax(tail)
    if head > x:
        return head
    else:
        return x

nums = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5, 8, 9, 7, 9]
print(mymax(nums))
```

qst.py

```python
def qs(xs):
    if len(xs) <= 1:
        return xs
    left = []
    right = []
    head = xs[0]
    tail = xs[1:]
    for x in tail:
        if x <= head:
            left.append(x)
        else:
            right.append(x)
    return qs(left) + [head] + qs(right)

lst = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5, 8, 9, 7, 9]
print(qs(lst))
```