CSc 120
Introduction to Computer Programming II

CODE EXAMPLES 02 

Assignment 4
Short Problems

The problem

```
def odds_and_evens(arglist):
    evens = []
    odds = []
    for i in range(len(arglist)):
        ith_element = arglist[i]

        if i % 2 == 0:
            evens.append(ith_element)
        else:
            odds.append(ith_element)

        assert ith_element_is_in_correct_list(arglist, i, evens, odds)

    return (evens,odds)
```

Booleans only:
no if (or while, or …) statements allowed

Suppose `if`s were allowed…

We could write:

```
def ith_element_is_in_correct_list(arglist, i, evens, odds):
    if i % 2 == 0:
        assert something1
    else:
        assert something2
```

i % 2 == 0 and *something1*

i % 2 != 0 and *something2*

Suppose `if`s were allowed…

We could write:

```
def ith_element_is_in_correct_list(arglist, i, evens, odds):
    if  i % 2 == 0:
        assert  something1
    else:
        assert  something2
```

either i % 2 == 0
or i % 2 != 0

Suppose `if`s were allowed…

We could write:

```
def ith_element_is_in_correct_list(arglist, i, evens, odds):
    if  i % 2 == 0:
        assert  something1
    else:
        assert  something2
```

either i % 2 == 0 and *something1*

or i % 2 != 0 and *something2*

Solution

```
def ith_element_is_in_correct_list(arglist, i, evens, odds):
    return  (i % 2 == 0 and something1) \
                or (i % 2 != 0 and something2)
```

Solution

```
def ith_element_is_in_correct_list(arglist, i, evens, odds):
    return  (i % 2 == 0 and something1) \
                or (i % 2 != 0 and something2)
```

infinite loops, break, and continue

Problem spec

“Repeatedly read and process queries from the user ... until the user enters an empty line”

Attempt 1

```
def process_query(avg_db, max_avgs):
    user_queries = ...

    done = False
    while not done:
        query = input()
        if query == "":
            done = True
        else:
            ...process the query...
```

Attempt 1

```
def process_query(avg_db, max_avgs):
    user_queries = ...

    done = False
    while not done:
        query = input()
        if query == "":
            done = True
        else:
            ...process the query...
```

- Does not express the program logic in a direct and straightforward way
- The resulting code is unnecessarily convolutificated

Attempt 2

```
def process_query(avg_db, max_avgs):
    user_queries = ...

    while True:
        query = input()
        if query == "":
            break

        ...process the query...
```

```
def process_query(avg_db, max_avgs):
    user_queries = ...

    while True:
        query = input()
        if query == "":
            break

        ...process the query...
```

Problem spec:
"repeatedly read and process queries from the user ... until ..."

```
def process_query(avg_db, max_avgs):
    user_queries = ...

    while True:
        query = input()
        if query == "":
            break

        ...process the query...
```

termination condition is not known at the top of the loop

Using break/continue in loops

- Use a **break** statement if:
  - the termination condition for the loop cannot be determined at the top of the loop

- Use a **continue** statement if:
  - part of the loop body should (sometimes) be skipped
  - but the condition for skipping cannot be determined at the top of the loop

Using break/continue in loops

```
...
for line in infile:
    if line[0] == '#':
        continue

    db = update_db(db, line)
    ...
```

Constants

- There is no constant declaration, per se
  - the convention is to use all capital letters with underscores separating words, e.g.,

```
MAX_SIZE = 100
TOTAL = 0
HP = 1
```

http://legacy.python.org/dev/peps/pep-0008/#constants

- Very useful for creating readable code