APS106

UNIVERSITY OF TORONTO

Tutorial 6 – Week 7

We'll be starting at the 10 minute mark

if nothing else, write #cleancode

Agenda

- Lab 3 review
- Lecture review
  - Python iterators: ranges
  - Iterative structures: `for` loops
  - Applications: iteration over iterative objects, e.g., strings and ranges
  - Comparison between `for` and `while` loops
- Practice questions

Learning Objectives

After this tutorial, learners should be able to:

- recognize and describe range objects
- understand how to use range objects
- recognize and describe for loops
- understand the execution for loops
- design and implement for loops for string and range objects

APS106

UNIVERSITY OF TORONTO

Lab Review

Detecting the overlap of two rectangles

if nothing else, write #cleancode

Lab 3: Rectangle Overlap

```
def rectangle_overlap(rect1_bl_x,rect1_bl_y,
                      rect1_tr_x,rect1_tr_y,
                      rect2_bl_x,rect2_bl_y,
                      rect2_tr_x,rect2_tr_y):
    """
    (int,int,int,int,int,int,int,int) -> str

    Function determines whether two rectangles overlap. When rectangles
    overlap, the function checks for the following scenarios
        1. The two rectangles share the same coordinates
        2. The first rectangle is contained within the second
        3. The second rectangle is contained within the first
        4. The rectangles have overlapping area, but neither is completely
           contained within the other

    Function inputs represent x and y coordinates of bottom left and top right
    corners of rectangles (see lab document)

    The function return one of the following strings corresponding to the
    scenario
        "no overlap"
        "identical coordinates"
        "rectangle 1 is contained within rectangle 2"
        "rectangle 2 is contained within rectangle 1"
        "rectangles overlap"

    """
```

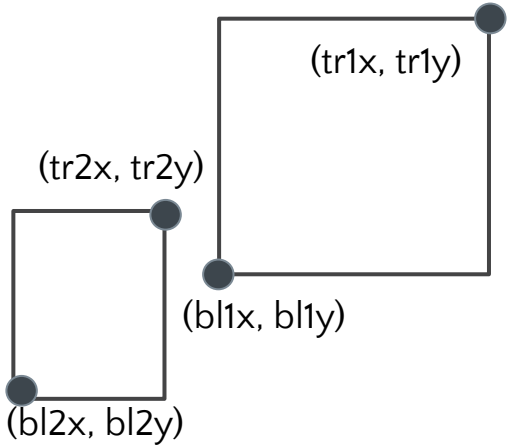
Lab 3: Rectangle Overlap

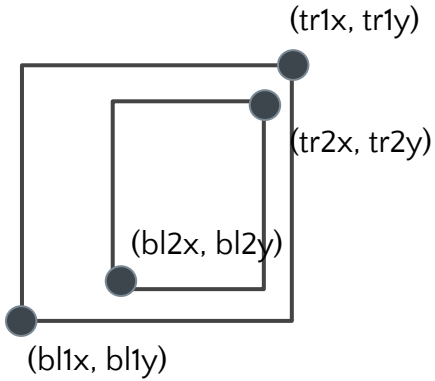
- **It is always a good idea to have a plan before coding anything:**
  - Can this problem be broken down into smaller steps?
  - Is there an easy part that we can start working on first?

- Make sure you test your code when you are working on **each part** of your algorithm plan!! (**do not wait until you finish everything!!**)

Lab 3: Sample Pseudo-code

- If bl1x > tr2x or bl1y > tr2y or vice versa, “no overlap”
- Else, if two coordinates are identical, “identical rectangles”
- Else, if the R2’s bottom left corner is on the upper right of the R1’s one, and the R2’s top right corner is on the lower left of the R1’s one, “R2 in R1”
- Else, if the R1’s bottom left corner is on the upper right of the R2’s one, and the R1’s top right corner is on the lower left of the R2’s one, “R1 in R2”
- Else, “overlap”





APS106

UNIVERSITY OF TORONTO

Review of Lecture

Python iterable objects: ***range*** objects

Range objects

Range objects are a type of iterable object in Python. They can be thought of as sequences of numbers.

Range objects can be generated using the built-in function `range()`:

```
range(start, stop, step)
```

- the `stop` value is not included in sequence of numbers generated
- `step` is optional. When omitted, `step` will be 1
- `step` can also be a negative value

NOTE: when only one value is passed to the range function, it is treated as the **stop** parameter. (i.e. `range(n)` → `range(0, n, 1)`)

APS106

UNIVERSITY OF TORONTO

Review of Lecture

Iterative structures: `for` loops

`for` loop general syntax

Iterables we have covered so far:

- strings
- ranges

```
for item in iterable:

    body
```

Body of the for loop needs to be indented !

The colon after the iterable is mandatory

APS106

UNIVERSITY OF TORONTO

Review of Lecture

Iterating over iterable objects: s***trings*** and ***ranges***

Iterating over ***string*** objects

Example: printing each character in a string:

```
string1 = 'Hello'

for character in string1:
    print(character)
```

Iterating over *range* objects

Example: printing each number in a range

```
for n in range(6, 0, -2):
    print(n)
```

Iterating using `for` and `while` loops

- `while` loops can also be used to iterate over iterable objects.

```
string1 = 'Hello'

for character in string1:
    print(character)
```

Equivalent to

```
string1 = 'Hello'
i = 0
while i < len(string1):
    print(string1[i])
    i += 1
```

Define and initialize the loop variable (In this example the loop variable holts the intext at which we start the iteration.

Update the loop variable

```
rng = range(4)

for r in rng:
    print(r)
```

Equivalent to

```
rng = range(4)
i = 0
while i < len(rng):
    print(rng[i])
    i += 1
```

APS106

UNIVERSITY OF TORONTO

Review of Lecture

Interrupting iteration

Interrupting the iteration of a loop

Two useful instructions:

- **`continue:`** terminates the **current iteration** immediately and continue to the next iteration.

- **`break:`** terminates the **loop** immediately

APS106

UNIVERSITY OF TORONTO

Review of Lecture

Nesting loops

Nesting loops

The body of a `for` or `while` loop can contain other loops (and much more).

- This structure is called a **nested loop**.
- **Example:**

```
for i in range(10, 13):
    for j in range(1, 5):
        print(i, j)
```

What does this code output ?

APS106

UNIVERSITY OF TORONTO

Practice Problems

slido

What is the value of the variable total after running the following code?

ⓘ Start presenting to display the poll results on this slide.

Review Practice Problem 1

Q1. What is the value of the variable **`total`** after running the following code?

```
total = 0

for x in range(1, 5, 2):
    for y in range(1, 3):
        total += x * y
```

A. 30

B. 54

C. 24

D. 12

E. None of the above / an error is thrown

slido

What is printed after running the following code?

ⓘ Start presenting to display the poll results on this slide.

Review Practice Problem 2

Q2. What is printed after running the following code?

```
def mystery(string):

    out = ""

    for char in string:

        if char == "i":
            break
        if char == "a":
            continue
        out += char

    return out

print(mystery("walking"))
```

A. walking

B. wlking

C. wlk

D. wlkng

E. None of the above or an error is thrown

Coding Question 1

- **Problem statement**
  - Complete function `alternate_letters` according to its docstring

```
def alternate_letters(s1, s2):
    """ (str, str) --> str
    Return a string made up of alternating letters from s1
and
   s2. Start with s1[0], then s2[1], s1[2], and so on.
  Assume len(s1) == len(s2).

    >>> alternate_letters ('abc', '123')
    'a2c'

    >>> alternate_letters ('abcd', '1234')
    'a2c4'

    """
```

slido

Any questions?

ⓘ Start presenting to display the poll results on this slide.