CSI31
Introduction to Computer Programming I

Dr. Sharon Persinger
November 19, 2018

Topics

- post-test loops
- loop-and-a-half
- event loops

Post-test loops

- In a post-test-loop, the loop condition is tested after the body of the loop is executed. So the loop body will be executed at least once.

- Some programming languages have a do-while loop construct. Python doesn’t have a do-while loop.

Input validation in Python, various ways of creating a post-test loop

Problem: Get a non-negative number as input.

- Seed the loop to get started:

```
number = -1 #illegal value to get started
while number < 0:
    number = float(input("Enter a positive number: "))
```

- break from an infinite loop:

```
while True:
    number = float(input("Enter a positive number: "))
    if number >= 0: break
```

One more version

- add information to the user

```
while True:
    number = float(input”Enter a positive number: “))
    if number > = 0:
        break #exit with valid input
    else:
        print(“The number you entered was not positive.”)
```

break statement

- The break statement can appear only inside a for loop or a while loop.
- Executing break causes the innermost loop to terminate.
- Execution continues with the next statement immediately following the loop.

Loop and a half version of sentinel loop

- General pattern:

```
while True:
    get next data item
    if the item is the sentinel, break
    process the item
```

- The sentinel is not processed.
- The break comes in the middle of the loop.

break statement: good or bad?

- Some programmers never (or almost never) use break statements. They think that the reason a loop terminates should be obvious - the loop condition is False.
- In any case, it is bad practice to use multiple break statements in a loop. The logic of the loop will be confusing.

GUI and Event loop

- Programs that use a graphical user interface or GUI are written in an event driven style.
- The user interface is displayed and then the interface waits for user events - mouse clicks, keys typed.
- The program processes the event.
- Underlying this is the event loop.

```
draw the GUI
while True:
        get next event
        if event is quit signal:  break
        process the event
clean up and quit
```

Example

- event_loop1.py Change the background of a window in response to key presses,

- The program waits for the user to press a key.
- Single input mode - keyboard

Another example with multimodal input - either key or mouse

- Write a program that changes the color of the window in response to keyboard click and allows the user to type text into the window by clicking the window and then typing.

- Problem - win.getKey() waits for a key to be pressed, win.getMouse() waits for the mouse to be clicked.

- How can we get input from either? Multimodal input
  - use methods checkKey() and checkMouse()
  - checkKey() returns key if one is pressed, returns the empty string if no key is pressed since last call to checkKey() or getkey(),
  - checkMouse() returns Point if mouse is clicked, returns None if the mouse has not been clicked since the last call to getMouse() or checkMouse()

Design of solution

```
Draw the GUI
while True:
    key = checkKey()
    if key is quit signal: break
    if key is valid key:
            process key

    click = checkMouse():
    if click is valid:
            process click

clean up and exit
```

Implementation of solution

event_loop3.py

The program uses functions to handle the key press and mouse click.

Handling the mouse is complicated.

- Display an Entry box.
- Get the text typed into the box until Enter is pressed.
- Remove the entry box and display the text in a Text object.