

Python Functions, Decision Structures, Loop Structures, and Booleans

Reading: Chapters 6-8 from Zelle text

- Mainly a review of concepts we have already seen when we studied algorithms
 - Functions—*sub-algorithms*
 - Decision structures—*if then else*
 - Loops—*for, while*
 - Booleans—*true, false, and, or, negation*
- A presentation of the Python version of these concepts
 - Python syntax
 - Python examples

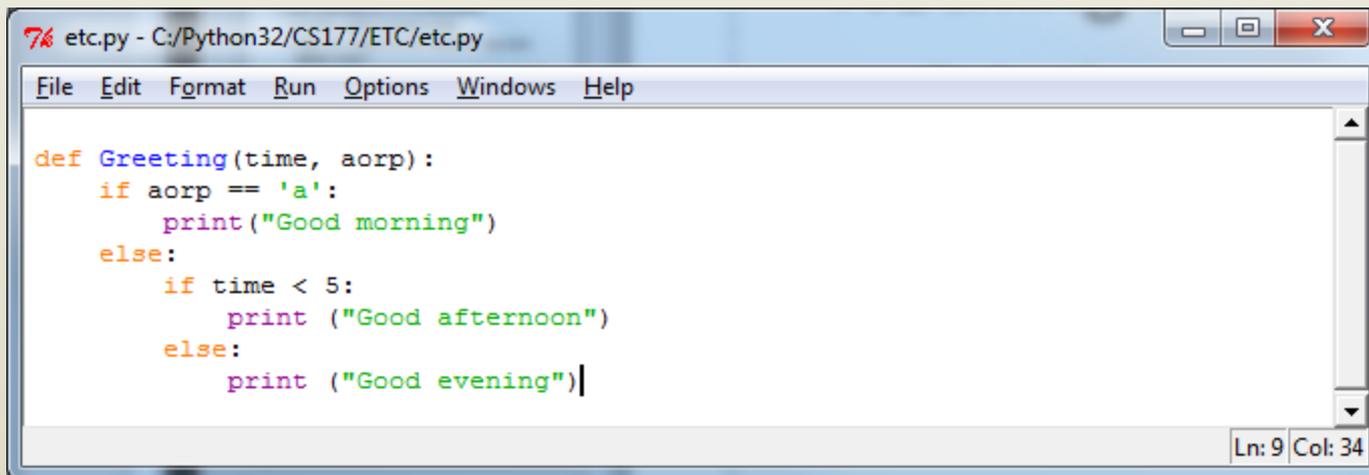
Python Functions

Functions

- Enable abstraction of functionality
 - Function called to achieve a task
 - Caller only needs to know interface: input and desired output
 - Details of how the output is computed are hidden, i.e. abstracted away, from caller
- Facilitate code development, debugging, maintenance, upgrade

Output of functions (1)

- Nothing is returned to caller
 - Printing on screen
 - Saving to file
 - Example: prints greeting given time and 'a' for a.m. or 'p' for p.m.



```
etc.py - C:/Python32/CS177/ETC/etc.py
File Edit Format Run Options Windows Help

def Greeting(time, aorp):
    if aorp == 'a':
        print("Good morning")
    else:
        if time < 5:
            print ("Good afternoon")
        else:
            print ("Good evening")|

Ln: 9 Col: 34
```

Output of functions (2)

- Value(s) returned explicitly to caller
 - Using the “return” statement
 - Multiple values can be returned

```
7% etc.py - C:/Python32/CS177/ETC/etc.py
File Edit Format Run Options Windows Help

def Min(A):
    min = A[0]
    for a in A:
        if min > a:
            min = a
    return min

def MinMax(A):
    min = A[0]
    max = A[0]
    for a in A:
        if min > a:
            min = a
        if max < a:
            max = a
    return min, max
```

```
>>> ===== RESTART =====
>>>
>>> min, max = MinMax([1, 3, 4, 2, -10, 10, 100, -20])
>>> min, max
(-20, 100)
>>> |
```

Output of functions (3)

- One or more values are returned implicitly to the caller by modifying input parameters
 - Only “mutable” parameters can be modified, such as Python lists

```
def AbsArray(A):  
    for i in range(len(A)):  
        if A[i] < 0:  
            A[i] = -A[i]
```

```
>>> A = [10, -1, 2, -13, -3, -2, 10, 1, 3]  
>>> AbsArray(A)  
>>> A  
[10, 1, 2, 13, 3, 2, 10, 1, 3]  
>>>
```

Variable scope

- Scope of a variable
 - Area of the program where the variable is known
 - Variable can only be referenced (i.e. read or written) within its scope
- Function parameters and variables are local to the function

Python Decision Structures

If statements

- Simple decision
 - One if statement with no else branch
- Two-way decision
 - One if statement with an else branch
- Multi-way decision
 - Nested if statements
 - “elif” can be used instead of “else:” followed by “if”

Example

7% etc.py - C:/Python32/CS177/ETC/etc.py

File Edit Format Run Options Windows Help

```
import graphics

def ImProc():
    while True:
        cmd = input("ImProc> ")
        if cmd == "new":
            w = eval(input("w: "))
            h = eval(input("h: "))
            win = graphics.GraphWin("Image", w, h)
        else:
            if cmd == "set":
                r = eval(input("r: "))
                g = eval(input("g: "))
                b = eval(input("b: "))
                win.setBackground(graphics.color_rgb(r,
            else:
                if cmd == "circle":
                    radius = eval(input("radius: "))
                    circle = graphics.Circle(graphics.P
                    circle.draw(win)
                else:
                    print("Unknown command: ", cmd)
```

7% etc.py - C:/Python32/CS177/ETC/etc.py

File Edit Format Run Options Windows Help

```
import graphics

def ImProc():
    while True:
        cmd = input("ImProc> ")
        if cmd == "new":
            w = eval(input("w: "))
            h = eval(input("h: "))
            win = graphics.GraphWin("Image", w, h)
        elif cmd == "set":
            r = eval(input("r: "))
            g = eval(input("g: "))
            b = eval(input("b: "))
            win.setBackground(graphics.color_rgb(r, g,
        elif cmd == "circle":
            radius = eval(input("radius: "))
            circle = graphics.Circle(graphics.Point(w/
            circle.draw(win)
        else:
            print("Unknown command: ", cmd)
```

Equivalent programs, elif improves readability, avoids excessive indentation

iClicker Question

At the *ImProc*> prompt the user types in *circle 20* followed by the enter key.

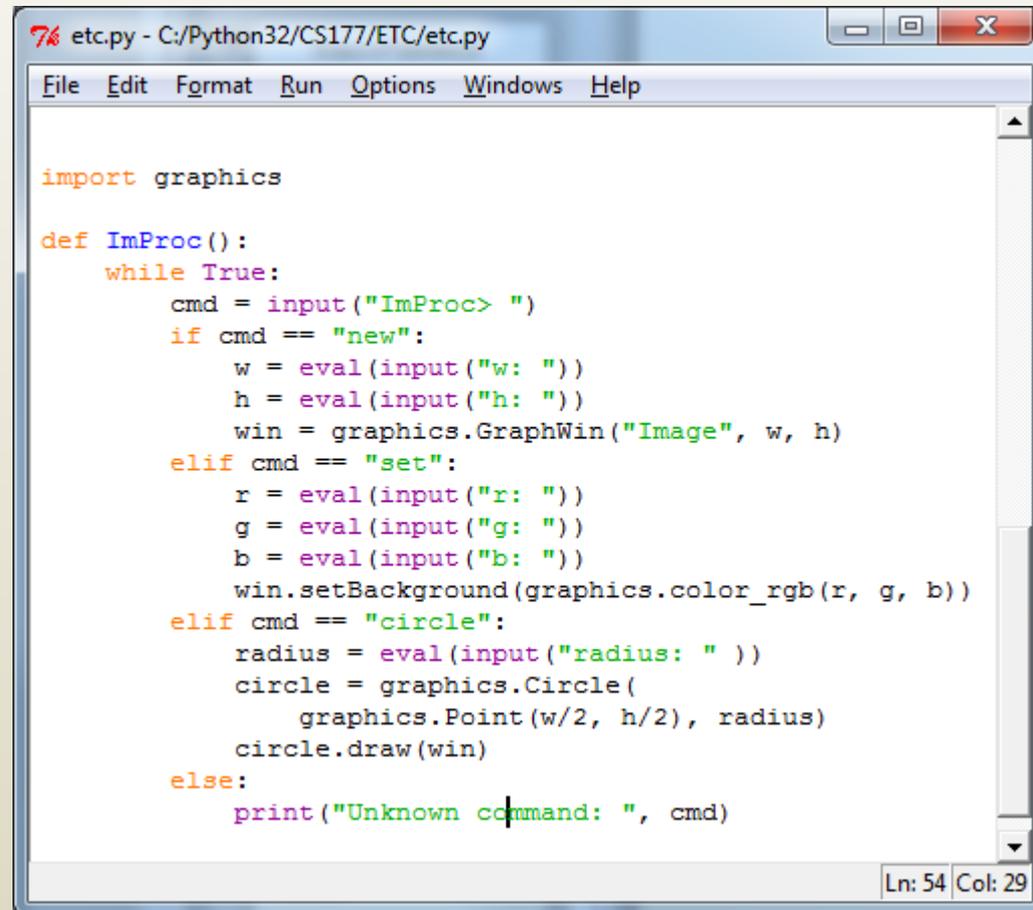
A. The function draws a circle of radius 20

B. The function prints out *Unknown command: circle 20*

C. The function prompts for the radius

D. A and then C

E. A and then B



```
etc.py - C:/Python32/CS177/ETC/etc.py
File Edit Format Run Options Windows Help

import graphics

def ImProc():
    while True:
        cmd = input("ImProc> ")
        if cmd == "new":
            w = eval(input("w: "))
            h = eval(input("h: "))
            win = graphics.GraphWin("Image", w, h)
        elif cmd == "set":
            r = eval(input("r: "))
            g = eval(input("g: "))
            b = eval(input("b: "))
            win.setBackground(graphics.color_rgb(r, g, b))
        elif cmd == "circle":
            radius = eval(input("radius: "))
            circle = graphics.Circle(
                graphics.Point(w/2, h/2), radius)
            circle.draw(win)
        else:
            print("Unknown command: ", cmd)

Ln: 54 Col: 29
```

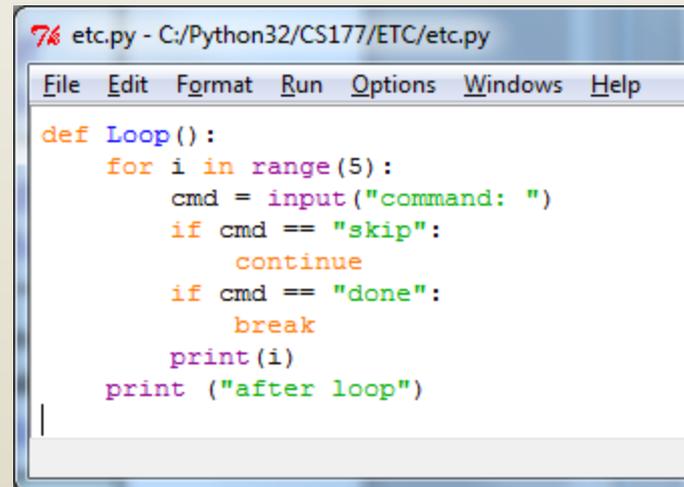
Loop Structures and Booleans

Loop basics

- For loops
 - When one knows the number of iterations
- While loops
 - When one doesn't know the number of iterations a priori
- Repeat until loops
 - No direct support in Python, can be simulated with while loop
- Controlling execution within loops
 - break
 - Abandon loop immediately
 - Continue with instruction immediately following loop
 - continue
 - Abandon current iteration immediately
 - Continue with first instruction of the loop body for the next iteration

iClicker Question

- What does the function Loop print out after the following commands: skip, skip, ha, done
- A. <nothing>, <nothing>, 2, after loop
 - B. 0, 1, 2, after loop
 - C. 0, 1, 3, 4, after loop
 - D. 0, 0, 1, 1, 2, after loop
 - E. None of the above



```
7% etc.py - C:/Python32/CS177/ETC/etc.py
File Edit Format Run Options Windows Help

def Loop():
    for i in range(5):
        cmd = input("command: ")
        if cmd == "skip":
            continue
        if cmd == "done":
            break
        print(i)
    print("after loop")
```

Boolean Arithmetic

- Boolean variable
 - Can have one of two values: true or false
- Boolean operators
 - Equal, not equal, greater, smaller, equal, and, or, etc.
- Boolean expression
 - Evaluates to true or false
 - Composed of Boolean variables and operators

DeMorgan's Laws

- $\text{not}(a \text{ or } b)$ is equivalent to $(\text{not } a) \text{ and } (\text{not } b)$
 - False when either a or b is true
 - True only when both a and b are false
- $\text{not}(a \text{ and } b)$ is equivalent to $(\text{not } a) \text{ or } (\text{not } b)$
 - False only when both a and b are true
 - True when either a or b is false
- Example: interval $(-3, 4)$ can be defined as
 - $(x > -3) \text{ and } (x < 4)$
 - $\text{not}(\text{not}(x > -3) \text{ or } \text{not}(x < 4))$
 - $\text{not}((x \leq -3) \text{ or } (x \geq 4))$