CS 160
CS Orientation

Input/Output, Conditionals, and Loops

OSU Oregon State University

Relational Operators and Symbols

- >
- >=
- <
- <=
- ==
- !=

Logical Operators and Symbols

- not
- and
- or

Python Examples

- not True or False
- 3 > 2 + 4
- True and True or True and False
- ((True and True) or True) and False
- not 3 < 2 and True or False

Python Decision Logic:
Print 1, 3, 5, or 7 stars
Differences/Similarities in these?

```
x=int(input(“Print 1, 3, 5, 7 stars?”));
```

```
if(x==1):
   print("   *   ")
if(x==3):
   print("  ***  ")
if(x==5):
   print(" ***** ")
if(x==7):
   print("*******")
```

```
if(x==1):
   print("   *   ")
elif(x==3):
   print("  ***  ")
elif(x==5):
   print(" ***** ")
elif(x==7):
   print("*******")
```

OSU Oregon State University

Python Decision Logic: Print 1, 3, 5, or 7 (for any other #) stars Differences/Similarities in these?

```
x=int(input(“Print 1, 3, 5, 7 stars?”));
```

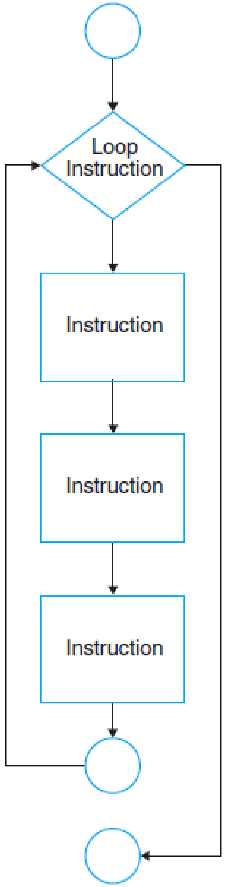
```
if(x==1):
   print("   *   ")
if(x==3):
   print("  ***  ")
if(x==5):
   print(" ***** ")
else:
   print("********")
```

```
if(x==1):
   print("   *   ")
elif(x==3):
   print("  ***  ")
elif(x==5):
   print(" ***** ")
else:
   print("********")
```

Exercise

- Write an algorithm that will tell a user whether they have entered a valid triangle using the triangle inequality property (any sum of 2 sides cannot be less than the third side).

Loop Logic Structure

Algorithm	Flowchart	Pseudocode
⋮ 5. Loop Instruction Instruction Instruction Until <logical expression> 6. ⋮	 Loop Instruction Instruction Instruction Instruction	⋮ Loop Instruction Instruction Instruction Until <logical expression> ⋮

Python Loop Logic

```
for x in range(7):
    print("*”, end=“”)
```

OR

```
x=1
while(x<=7):
     print("*”, end=“”)
     x+=1
```

Exercise

- How about if we alter this to allow a user to do this for any number of triangles?

Strings

- Create a string

```
my_string=“hello”;
```

- Access a character

```
my_string[0] #gives you first character
```

- Length

```
len(my_string)
```

Exercise

- Write an algorithm to determine if input is bad without using exceptions, i.e. it would work in any language!!! ☺

Functions

- May need to import a library
- Use the function from library/object
- Example:

```
import math
math.sqrt(4)
```

In-class Exercise #4

Design a Python program that takes a **positive whole number** *n* as input and **outputs the square root of n** using the Babylonian algorithm. The Babylonian algorithm computes the square root of a positive number, *n*, as follows:

1. Make a guess at the answer (you can pick n/2 as your initial guess).
2. Compute r = n / guess
3. Set guess = (guess + r) / 2
4. Go back to step 2 for as many iterations as necessary. The more steps 2 and 3 are repeated, the closer guess will become to the square root of n.
5. Compare your calculated square root with the math.sqrt() result.