# Lab 4 Pseudocode

## Van Robbins 9/23/2024

Sorry about the underscores tabs were screwed up

### Executable File

import utility file  
 Create a function that asks for the user to select a shape name it \_\_\_\_chooseshape  
 create shape and validShape["1","2","3"] variables, valid shape array holds valid entrys for shape  
 print("What shape would you like to draw?");  
 shape=input("1| Square\n2| Rectangle\n 3| Equilateral Triangle");  
   
 #Checks that shape is valid allows user to enter any value not just int without error  
 if shape in validShape:   
 convert shape to int  
 send shape to makeShape function in utility file  
 else: (shape is not valid)  
 print("Please pick a valid shape by entering a number 1-3\n");  
 restarts function for user to try again  
 chooseShape();

### Utility File

first function that is called by main file  
def makeShape(selectedShape):  
 if shae isSquare or Triangle  
 sizeEq = sizeEqualWL(),function asks user for one side  
 Checks that entry is valid with while loop, will loop size function until it is  
 uses sizeCheck(sizeEq) function;  
 Entry is valid:  
 Converts size to int  
 if selectedShape==1:Square  
 square(sizeEq); function that creates square  
 else:  
 triangle(sizeEq); function that creates triangle  
 else shape is rectangle  
 sizeDiff = sizeDiffWL(),function that asks user for both length and \_\_\_\_\_\_\_\_width, returns both values in array  
 Checks that entries are valid with while loop, will loop size \_\_\_\_\_\_\_\_function until they are  
 uses sizeCheck() function to check both values  
 else:  
 convert both of the values in the array into ints and assign them to their own variables  
 rectangle(width,length); function that creates rectangle  
  
def sizeEqualWL() One size length and width function  
 wl=input("Please enter the size of a side or radius of the shape (2-\_\_\_\_10)");  
 return wl;  
  
def sizeDiffWL()  
 print("Please enter the width and length of the shape of the shape (2-\_\_\_\_10)")  
 w=input("Width: ")  
 l=input("Length: ")  
 else:  
 return w,l;  
  
def sizeCheck(sizeInput):  
 validSize=["3","4","5","6","7","8","9","10"];  
 if sizeInput not in validSize:  
 print("Try again! The size must be between 3 and 10");  
 return False;  
 else size is valid  
 return True;   
  
def square(size):  
 width=size;  
 height=size;  
 loop that is only false if height is equal to 0  
 If statement checks if the height is the top or bottom, runs first \_\_\_\_\_\_\_\_because it is true for most values  
 Resets width to size-1, accounts for printing left outer symbol  
 print("⬜") except make sure print statement does not create new \_\_\_\_\_\_\_\_\_\_\_\_line  
 Use while loop to print spaces until reach outward ⬜  
 print("  ") make sure print statement does not create new \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_line  
 width-=1;  
 print("⬜"); right outer symbol prints new line as well  
 height-=1; Subtracts 1 from height as this layer of the square is \_\_\_\_\_\_\_\_\_\_\_\_done  
 else: top or bottom of square  
 width=size; width reset for correct symbols to be printed  
 while(width>0):  
 print("⬜"); except make sure print statement does not \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_create new line  
 width -=1; Subtracts 1 from width every time symbol printed  
 height-=1; Subtracts 1 from height as this layer of the square is \_\_\_\_\_\_\_\_\_\_\_\_done  
  
def triangle(size):  
 spaceInside=0; space between two sides   
 spaceOutside=size-1; space outside the sides of the triangle  
 width=size-2; used to create bottom of triangle minus the two sides  
 height=size;  
 count=0; needed to count spaces in loops  
 loop that is only false if height is below 1  
 count=0; ensures count is correctly set to 0  
 If statement checks if the height is the bottom, runs first because \_\_\_\_\_\_\_\_it is true for most values  
 Loop prints proper outer spaces to left side of triangle  
 while(spaceOutside>count):  
 print(" "); ensure doesn't print new line   
 count+=1;  
 print("/") Prints out left side after spaces ensure doesn't print \_\_\_\_\_\_\_\_\_\_\_\_new line  
 count=0; Resets count to be used to find proper inner space  
 Loop prints inside spaces of triangle  
 while(spaceInside>count):   
 print(" "); ensure doesn't print new line   
 count+=1;  
 print("\"); Prints out right side after spaces ensure doesn't \_\_\_\_\_\_\_\_\_\_\_print new line  
 Layer is done remove 1 from outerspace and height for next layer  
 Add two spaces to inside of triangle for next layer  
 else bottom of triangle  
 print("/"); Print left edge  
 while(width>0): loop print \_ until reach right edge  
 print("\_"); ensure does not print new line  
 width -=1;   
 print("\"); Prints out single \, right outer side  
 Subtract 1 from height, height should be 0 ending loop  
  
def rectangle(sizeW, sizeL):  
 Very similar to square just width and height are set to 2 different \_\_\_\_values  
 width=sizeW;  
 height=sizeL;  
 loop that is only false if height is equal to 0  
 If statement checks if the height is the top or bottom, runs first \_\_\_\_\_\_\_\_because it is true for most values  
 Resets width to size-1, accounts for printing left outer symbol  
 print("⬜") except make sure print statement does not create new \_\_\_\_\_\_\_\_\_\_\_\_line  
 Use while loop to print spaces until reach outward ⬜  
 print("  ") make sure print statement does not create new \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_line  
 width-=1;  
 print("⬜"); right outer symbol prints new line as well  
 height-=1; Subtracts 1 from height as this layer of the square is \_\_\_\_\_\_\_\_\_\_\_\_done  
 else: top or bottom of square  
 width=size; width reset for correct symbols to be printed  
 while(width>0):  
 print("⬜"); except make sure print statement does not \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_create new line  
 width -=1; Subtracts 1 from width every time symbol printed  
 height-=1; Subtracts 1 from height as this layer of the square is \_\_\_\_\_\_\_\_\_\_\_\_done