**Python Lab 7a Intro to CodeSkulptor**

Py3.CodeSkulptor.org is a website that runs a version of Python suitable for making interactive games.

After you complete the directions in each box, test your program with many possible inputs to be sure it works.

|  |  |
| --- | --- |
| 0. Go to py3.codeskulptor.org  click on the button that says “Docs”  in the search bar, enter the phrase “draw text”  click on the link for “Draw Text on Canvas”  copy in the space below the syntax for canvas.draw\_text(  At the top of the program given on this page, add a new line with your name as a comment  # Your Name  Run the program. | 1. Change the initial message to “Computer.”  Change the second one to “Science.”  Change the color and font size.  Change the name on the button to “please click.”  Change the message to "You did it !!" |
| 2. make a new line after import simplegui  fontsize = 48  at the bottom of your program, add this:  frame.add\_button(“smaller” , smaller\_font)  add this function near the other functions  - def smaller\_font():  global fontsize  fontsize = fontsize - 4  Test your new button..  Then, make another button called Larger font  that increases the font by 4 | 3. Click on the Docs button and search for ‘color’  Write down the names of 8 different colors that Code Skulptor accepts  create a variable called font color  Make a button that changes font color to red.  Test your program.  Make 3 more buttons, for 3 different font colors  Change the name of the frame to ‘Program 1’ |
| 4. To save this program:  -generate a URL  -create a separate document for all your CodeSkulptor links  -write information about this program in the document  - paste the URL in the document | 5. Change the vertical size of the frame to be 500 pixels. Run your program.  Make a variable for the y-position of the message  Make the message slowly scroll down the screen  Run your program  If the message gets to the bottom of the screen, it should wrap back up to the top |

**Python Lab 7b Using the Random Module**

|  |  |
| --- | --- |
| 0. Go to Py3.CodeSkulptor.org and start a new program. At the top, add a comment for your name  click on the button that says “Docs”  in the search bar, enter the word “random”  in the space below, write four different random functions  random.randint()  random.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  random.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  random.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  At the start of your program, type the following:  import random | 1.  - make a variable called fontsize  - set its value to a random integer between 2 and 100  - in the draw\_text function use the variable fontsize  - in the click function, set fontsize to a random number again  - test that your program has a random fontsize when run, and also a different random fontsize each time the button is clicked |
| 2. Look up in the Docs how to set the canvas background color.  Also look up in the Docs how to make a random element of a sequence.  Set the background color to a random choice of five colors that are chosen from a list. | 3. At the start of your program, make a variable called font color to be set to a random choice of five different colors, none of which are the same as your background colors.  font color = random. \_\_\_\_\_\_\_\_\_\_\_ ( [ ‘blue’ , , , ] )  The initial color should be random.  Then, inside the click function, do this again. Each time you click the button, a random color should appear. |
| 4. Save your work by clicking on the save button. Then, change the message that appears when you click the button to be one of three different short phrases. You may need to change your range of font sizes to make sure every message can fit. | 5. Inside the draw function, add on to the y coordinate of the message a random integer between -3 and 3.  What effect will this have on your message?  6. Adjust the position of message so that it is always centered vertically and horizontally on your screen, no matter what the fontsize or the message is (just like a PowerPoint slide with centering) |

**Python Lab 7c String-Int Conversions**

|  |  |
| --- | --- |
| 0. Go to Py3.CodeSkulptor.org and add a comment for your name.  Click on the button that says “Docs”  In the search bar, enter the word “string"  In the space below, write a command to convert the value 556 into a string:  Also, write the function that turns a string into a number: | 1. At the top of your program, make a variable:  number = 0  In the draw function, add this line:  print (number)  Run your program.  Then, add code in the draw function to increase number by 1. Run your program, you should see increasing numbers in the output pane.  Stop your program by clicking on the ‘reset’ button in CodeSkulptor. |
| 2. Let’s try to get the number to appear on the canvas.  In the draw function, change *message* to number.  **Run your program.** It will not work.  Look at step 0 above to convert the number to a string. | 3. Let’s try to allow the user to input the original number. At the top of the program,change the line to  number = int(input(“enter a number”))  Run your program.  One that works, at the top of the program, set a variable fontsize to a random integer between 30 and 60. Use this variable in the draw handler.  Save your work. |
| 4. Let’s make the button change the numbers from increasing to decreasing. With the other global variables, add this line:  **num\_to\_add = 1**  Change the wording on your button to say “change direction.” Put code in this function so that when you click on the button, you multiply num\_to\_add by -1.  Test your program to see if the numbers can change direction. | 5. These numbers go by really quickly...is there a way to make the numbers increment or decrement more slowly?  Try to solve this problem by thinking mathematically, and by changing only one line of code.  6. Make a button so that each time you click on it, the process slows down a bit more than before, but the button will only work five times. After that, the button can be clicked, but it won’t change anything. |

**Python Lab 7d Drawing Circles and KeyDown Handlers**

|  |  |
| --- | --- |
| 0. Look up how to draw a circle | At the top of your program, make a comment for your name.  In the Docs, search for **circle**. In the space below, write the general syntax for drawing a circle.  Put code in your draw handler to draw a circle with a **filled color**.  The circle should appear near the **center** of the canvas. |
| 1. Ask the user to enter numbers for the circle’s x, y, and radius. | Near the top of your program, enter this code:  **x = int ( input ( “ enter the x coordinate”))**  Do the same for the y coordinate and the radius.  Modify the code to draw the circle with the parameters you entered. |
| 2. Make buttons to move the circle Left and Right. | Add two buttons to your frame, one called Left and another one called Right.  Write functions called:  **def left(): and def right():**  The functions change the x coordinate by five pixels.  Then do the same for up and down. |
| 3. Learn how to call functions with keyboard commands.  You can use different keys if you want, such as:  “A,” “S,” “D,” “F” | In the documentation, look up “keyboard input.”.  After the line where you created your frame, enter this code:  **frame.set\_keydown\_handler(keydown)**  Near the lines where you wrote the other functions, write a function like this:  **def keydown(key):**  **if key == simplegui.KEY\_MAP['down']:**  **down()**  **elif key == simplegui.KEY\_MAP['up']:**  **up()**  You write the code that handles left and right. |
| 4. Don’t let the user move the circle off the screen. | Add code in each function up, down, left, right, that will check to make sure the circle can make this move and still stay completely on the screen. If it can, then make the intended move. Otherwise, do nothing:  if \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ :  command  command |
| 5. Constrain/Wrap | Make a button that toggles between this “constrain” mode and “wrap” mode, which allows the circle to smoothly wrap to the other side of the screen. Show a small message “constrain” or “wrap” in the upper left corner. The wrap must be smooth, so that it shows both halves of the circle as it slowly wraps around. This should work for any frame size. |

**Python Lab 7e Drawing Rectangles**

|  |  |
| --- | --- |
| 0. Look up how to draw a rectangle (polygon) | At the top, Make a comment for your name.  In the Docs, search for **polygon**. Write the general syntax for drawing a polygon.  Put code in your draw handler to draw a rectangle with a **filled color**.  The rectangle should appear near the **center** of the canvas. |
| 1. Create variables for x, y, width, and height | On the back of this paper, sketch a rectangle with:  a top left coordinate of (x, y)  a top right coordinate of (x + width, y)  a bottom right coordinate of (x + width, y + height)  a bottom left coordinate of (x, y + height)  Fill in this list of the polygon’s coordinates, using x, y, width, height  coordinates = [ ] |
| 2. Ask the user to enter numbers for the rectangle’s x, y, width, and height | Near the top of your program, enter this code:  **x = \_\_\_\_\_\_\_\_\_\_\_\_\_\_enter the x coordinate”))**  Do the same for the y coordinate, width, and the height.  Modify the code to draw the rectangle with the parameters you entered. |
| 3. Make buttons to move the rectangle up, down, left, right, and teleport | Add four buttons to your frame, up, down, left, and right,  Write four functions that handle the button clicks.  Also add a button that teleports the rectangle to a random x and y location, while keeping the width and height the same. |
| 4. Write a keyup function  You can use different keys if you want,  such as  ‘A’, ‘S’, ‘D’, ‘F’, ‘T’ | Add a keyup handler to your frame and write a function like this:  **def keyup(key):**  **if key == simplegui.KEY\_MAP['down']:**  **down()**  **elif key == simplegui.KEY\_MAP['up']:**  **up()**  **…..**    Implement all your movement functions with keys |
| 5. Challenge: Implement a counter. | A counter is incremented by 1 every time a key is pressed and the rectangle covers the middle pixel of your frame.  Draw the value of the counter on the top right corner of your canvas.  canvas.draw\_text…………  Change the frame size and run it again, it still should work. |