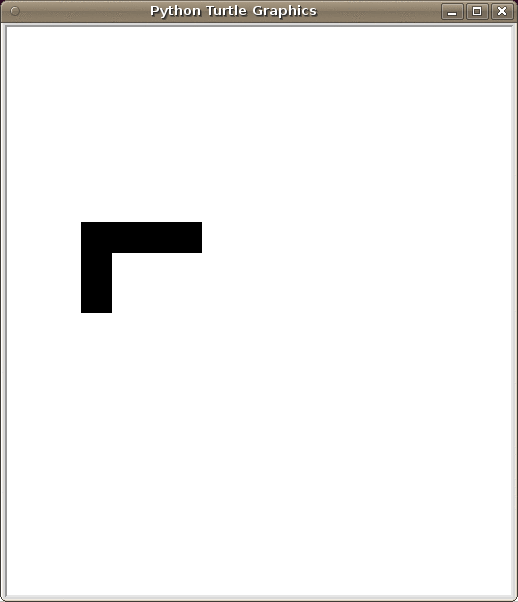


**π3**

**Python, the Python python**



Start this lab using startlab, and today you are starting the **Mini Project!** So when it asks for the lab/module number, type mini-proj

In the google docs, go to the snake game folder open the **starter\_code.py** file. Copy all the code. Make a new file in IDLE3 and copy the code there. Make sure to save it in your personal folder. For example: caleb20\_mini-proj

There’s already some code here, but you need to fill in most of it yourself!

**Part 1: Building a snake!**

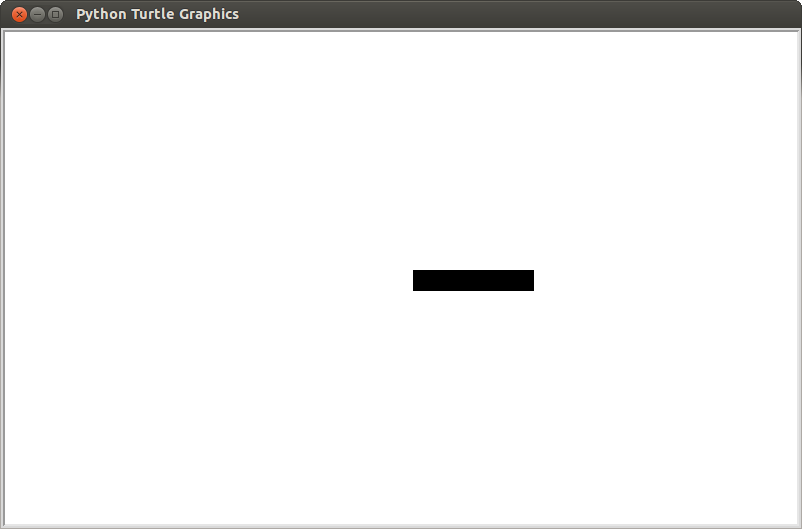
Find this part of code in your file:

|  |
| --- |
| import turtle  import random #We'll need this later in the lab  turtle.tracer(1,0) #This helps the turtle move more smoothly  SIZE\_X=800  SIZE\_Y=500  turtle.setup(SIZE\_X, SIZE\_Y) #Curious? It's the turtle window  #size.  turtle.penup()  SQUARE\_SIZE = 20  START\_LENGTH = 6  #Initialize lists  pos\_list = []  stamp\_list = []  food\_pos = []  food\_stamps = []  #Set up positions (x,y) of boxes that make up the snake  snake = turtle.clone()  snake.shape("square")  #Hide the turtle object (it's an arrow - we don't need to see it)  turtle.hideturtle() |

You will make a for-loop to draw the snake body by filling in the missing lines.

|  |
| --- |
| #Draw a snake at the start of the game with a for loop  #for loop should use range() and count up to the number of pieces  #in the snake (i.e. START\_LENGTH)  for \_\_\_\_\_\_ in \_\_\_\_\_\_\_\_\_\_ :  x\_pos=snake.\_\_\_\_\_\_\_\_\_\_\_\_\_ #Get x-position with snake.pos()[0]  y\_pos=snake.\_\_\_\_\_\_\_\_\_\_\_\_\_  #Add SQUARE\_SIZE to x\_pos. Where does x\_pos point to now?  # You're RIGHT!  x\_pos+=\_\_\_\_\_\_\_\_\_\_  my\_pos=(x\_pos,y\_pos) #Store position variables in a tuple  snake.goto(\_\_\_\_\_\_\_\_\_\_\_\_,\_\_\_\_\_\_\_\_\_\_\_\_) #Move snake to new (x,y)    #Append the new position tuple to pos\_list  \_\_\_\_\_\_.append(\_\_\_\_\_\_\_\_\_\_\_\_\_)  #Save the stamp ID! You'll need to erase it later. Then append  # it to stamp\_list.  \_\_\_\_\_\_\_\_\_\_\_\_\_ = snake.stamp()  \_\_\_\_\_\_.append(\_\_\_\_\_\_\_\_\_\_\_\_\_) |

Run the code. What did you see? It should look something like this.



Change START\_LENGTH to 10 and run the code again. What happened?

Go to the shell. Type

>>>pos\_list

and hit enter. What appears? Why?

Change START\_LENGTH to your favorite number between 1 and 10.

**Part 2: Move around!**

Now, add the following code to your file, piece by piece. Some parts of the code will change the way the snake looks on the screen; others will set up variables internally. Also, you need to fill in parts of the code yourself -- look at the comments to find out where.

|  |
| --- |
| UP\_ARROW = "Up" #Make sure you pay attention to upper and lower  #case  LEFT\_ARROW = "Left" #Pay attention to upper and lower case  DOWN\_ARROW = "Down" #Pay attention to upper and lower case  RIGHT\_ARROW = "Right" #Pay attention to upper and lower case  TIME\_STEP = 100 #Update snake position after this many  #milliseconds  SPACEBAR = "space" # Careful, it's not supposed to be capitalized!  UP = 0  #1. Make variables LEFT, DOWN, and RIGHT with values 1, 2, and 3  ####WRITE YOUR CODE HERE!! |

1. Make a variable called DOWN and give it the value 1. Then, make variables, LEFT and RIGHT, with values 2 and 3.
2. Write four functions that will control what happens when the user presses the up, down, left, or right keys. The up() function is defined already for you (see the next page). You will write the down(), left(), and right() functions by yourself.

|  |
| --- |
| direction = UP  def up():  global direction #snake direction is global (same everywhere)  direction=UP #Change direction to up  move\_snake() #Update the snake drawing <- remember me later  print("You pressed the up key!")  #2. Make functions down(), left(), and right() that change direction  ####WRITE YOUR CODE HERE!!  turtle.onkeypress(up, UP\_ARROW) # Create listener for up key  #3. Do the same for the other arrow keys  ####WRITE YOUR CODE HERE!!  turtle.listen()  def move\_snake():  my\_pos = snake.pos()  x\_pos = my\_pos[0]  y\_pos = my\_pos[1]    if direction==RIGHT:  snake.goto(x\_pos + SQUARE\_SIZE, y\_pos)  print(“You moved right!”)  elif direction==LEFT:  snake.goto(x\_pos - SQUARE\_SIZE, y\_pos)  print(“You moved left!”)  #4. Write the conditions for UP and DOWN on your own  ##### YOUR CODE HERE  #Stamp new element and append new stamp in list  #Remember: The snake position changed - update my\_pos()  my\_pos=snake.pos()  pos\_list.append(my\_pos)  new\_stamp = snake.stamp()  stamp\_list.append(new\_stamp)  ######## SPECIAL PLACE - Remember it for Part 5  #pop zeroth element in pos\_list to get rid of last the last  #piece of the tail  old\_stamp = stamp\_list.pop(0)  snake.clearstamp(old\_stamp)  pos\_list.pop(0) |

1. Add “listeners” using the turtle.onkeypress function for DOWN\_ARROW, LEFT\_ARROW, and RIGHT\_ARROW. The listener for UP\_ARROW is already shown as an example.
2. Add elif conditional statements in move\_snake to move the snake to the correct spot. The LEFT and RIGHT cases are already shown as examples.

***Now, run the code and try moving the snake around the screen!***

**Part 3: Detecting collisions with the wall**

|  |
| --- |
| #Go to the top of your file, and after the line that says direction = UP, write:  UP\_EDGE = 250  DOWN\_EDGE = -250  RIGHT\_EDGE = 400  LEFT\_EDGE = -400 |
| #Now go add code to the end of your move\_snake() function  def move\_snake():  . . .  #Add new lines to the end of the function  #Grab position of snake  new\_pos = snake.pos()  new\_x\_pos = new\_pos[0]  new\_y\_pos = new\_pos[1]  # The next three lines check if the snake is hitting the  # right edge.  if new\_x\_pos >= RIGHT\_EDGE:  print(“You hit the right edge! Game over!”)  quit()  # You should write code to check for the left, top, and bottom edges.  #####WRITE YOUR CODE HERE |

**Part 4: (Optional) Snakes in motion: moving automatically**

Go back to the up(), down(), left(), and right() functions and remove the line that calls move\_snake(). The up() function will now look like this:

|  |
| --- |
| def up():  #Listen for global direction  direction=UP  print("You pressed the up key!")  # You should do the same for the up(), down(), left(), and right() functions  ##### YOUR CODE HERE |

Now, go back to the move\_snake() function and add one line to the end of the function.

|  |
| --- |
| def move\_snake():  ...  #Add this line to the end of the move\_snake function  turtle.ontimer(move\_snake,TIME\_STEP) #<--- new line here  #Now, call the move\_snake() function. This starts moving the snake. Once it starts  #moving, it keeps moving by itself:  move\_snake() |

turtle.ontimer(move\_snake,TIME\_STEP)takes two input arguments:

move\_snake = the name of the function that is called at each time step

TIME\_STEP = wait this amount of time in **milliseconds** (1000 milliseconds = 1 second), then call function, then wait again, then call function,....

Now that you know this, explain how this code works so that the move\_snake() is called once every 100 milliseconds. ***Don’t continue until you have shown an instructor or TA your explanation.***

**Challenge: Part 5: Food for thought: how to grow your snake**

Now, add the following lines to the end of your code.

|  |
| --- |
| turtle.register\_shape("trash.gif") #Add trash picture  # Make sure you have downloaded this shape  # from the Google Drive folder and saved it  # in the same folder as this Python script  food = turtle.clone()  food.shape("trash.gif")  #Locations of food  food\_pos = [(100,100), (-100,100), (-100,-100), (100,-100)]  food\_stamps = []  # Write code that:  #1. moves the food turtle to each food position  #2. stamps the food turtle at that location  #3. saves the stamp by appending it to the food\_stamps list using  # food\_stamps.append( )  #4. Don’t forget to hide the food turtle!  for this\_food\_pos in food\_pos :  ####WRITE YOUR CODE HERE!! |

If you run your code, you should see something like this (not including the snake):



Now, find out whether the head of the snake is in the same position as the food. Inside of the move\_snake() function, in the location before you pop the zeroth element in pos\_list (this spot is indicated by the words “SPECIAL PLACE” in the comment), add code to do the task described below:

|  |
| --- |
| def move\_snake():  ...  ######## SPECIAL PLACE - Remember it for Part 5  global food\_stamps, food\_pos  #If snake is on top of food item  if snake.pos() in food\_pos:  food\_ind=food\_pos.index(snake.pos()) #What does this do?  food.clearstamp(food\_stamps[food\_ind]) #Remove eaten food  #stamp  food\_pos.pop(food\_ind) #Remove eaten food position  food\_stamps.pop(food\_ind) #Remove eaten food stamp  print(“You have eaten the food!”)    #HINT: This if statement may be useful for Part 8  ...  #Don't change the rest of the code in move\_snake() function:  #If you have included the timer so the snake moves  #automatically, the function should finish as before with a  #call to ontimer()  turtle.ontimer(move\_snake,TIME\_STEP) #<--Last line of function |

**Challenge: Part 6: Make food appear randomly**

Now, make the food appear in random positions, like a real game!!

Copy the code below above your move\_snake() function. Read the comments telling you which parts you need to write.

|  |
| --- |
| def make\_food():  #The screen positions go from -SIZE/2 to +SIZE/2  #But we need to make food pieces only appear on game squares  #So we cut up the game board into multiples of SQUARE\_SIZE.  min\_x=-int(SIZE\_X/2/SQUARE\_SIZE)+1  max\_x=int(SIZE\_X/2/SQUARE\_SIZE)-1  min\_y=-int(SIZE\_Y/2/SQUARE\_SIZE)-1  max\_y=int(SIZE\_Y/2/SQUARE\_SIZE)+1    #Pick a position that is a random multiple of SQUARE\_SIZE  food\_x = random.randint(min\_x,max\_x)\*SQUARE\_SIZE  food\_y = random.randint(min\_y,max\_y)\*SQUARE\_SIZE  ##1.WRITE YOUR CODE HERE: Make the food turtle go to the randomly-generated  ## position  ##2.WRITE YOUR CODE HERE: Add the food turtle's position to the food positions list  ##3.WRITE YOUR CODE HERE: Add the food turtle's stamp to the food stamps list |

Now go to the bottom of your move\_snake function and add something like this so that new food will appear:

if len(food\_stamps) <= 6 :

make\_food()

Note: turtle.ontimer should still be the last line of your function!!!

**Challenge: Part 7: Don’t eat yourself!**

In the real snake game, the game is over if the head of the snake ever touches any part of its body. You now know enough to add this rule to the game. Get to it!

**Challenge: Part 8: Make your snake grow!**

In the real snake game, the snake grows each time it eats some food! Adjust your code to make the snake increase by one unit length with each food eaten!

(Hint: Go back to the section of code added from Part 5!)