File Edit View Run Kernel Diagram Tabs Settings Help

Assignment 9.05

Code

Python (Pyodide)

[24]:
```
import numpy as np
```

[25]:
```
def order_corner_points(corner_points):
    """ This method takes a list of corner points (a list of 2-tuples) and returns the same structure such that
     the first tuple is the top-left point, second tuple is top-right point, third tuple is bottom-left point, and
     fourth tuple is bottom-left point.  The scale of this problem will always be small, so we worry about human
     read-ability above all else. """
```

[26]:
```
# First for loop will find the mins and maxs in each dimension.
# Assumption: Values do not go above 100,000,000
```

[27]:
```
min_x, min_y = 100000000, 100000000
```

[28]:
```
max_x, max_y = 0, 0
```

[*]:
```
for tup in corner_points:
    if tup[0] < min_x:
            min_x = tup[0]
        if tup[1] < min_y:
            min_y = tup[1]
        if tup[0] > max_x:
            max_x = tup[0]
        if tup[1] > max_y:
            max_y = tup[1]
```

Simple 0 1 Python (Pyodide) | Idle Mode: Edit Ln 1, Col 26 Assignment 9.05


Assignment 9.05
Code
Python (Pyodide)
continue
[30]: # Now, the mins and maxs have been set.
[31]: # We loop through again to label the points.
[32]: new_corner_points = []
for tup in corner_points:
if tup[0] == min_x and tup[1] == max_y:
new_corner_points.append(tup)
for tup in corner_points:
if tup[0] == max_x and tup[1] == max_y:
new_corner_points.append(tup)
for tup in corner_points:
if tup[0] == min_x and tup[1] == min_y:
new_corner_points.append(tup)
for tup in corner_points:
if tup[0] == max_x and tup[1] == min_y:
new_corner_points.append(tup)
NameError Traceback (most recent call last)
Input In [29], in <cell line: 2>()
1 new_corner_points = []
----> 2 for tup in corner_points:
3 if tup[0] == min_x and tup[1] == max_y:
Simple
Python (Pyodide) | Idle
Mode: Command
Ln 1, Col 26
Assignment 9.05

File Edit View Run Kernel Diagram Tabs Settings Help

Assignment 9.05

Code

Python (Pyodide)

[20]:
```
return new_corner_points
```

```
  Input In [18]
    return new_corner_points
    ^
SyntaxError: 'return' outside function
```

[33]:
```
def get_pixel_dimensions(corner_points, image_dimension):
    """ This method takes ordered corner points and an image dimension and finds the width and height of the pixels. """

    frame_width = corner_points[1][0] - corner_points[0][0] # top-right x-val minus top-left x-val
    frame_height = corner_points[0][1] - corner_points[2][0] # top-left y-val minus bottom-left y-val

    # The -1 is because n pixels actually have n-1 line segments to traverse them all.
    pixel_width = frame_width / (image_dimension[0] - 1.0)
    pixel_height = frame_height / (image_dimension[1] - 1.0)

    return pixel_width, pixel_height
```

[34]:
```
def pretty_print_solution(solution):
    """ This method prints the solution in a nice human read-able format. """
    print("  [")
    for row in solution:
        print(row)
    print("  ]")
    return
```

Simple 0 1 Python (Pyodide) | Idle Mode: Command Ln 1, Col 26 Assignment 9.05

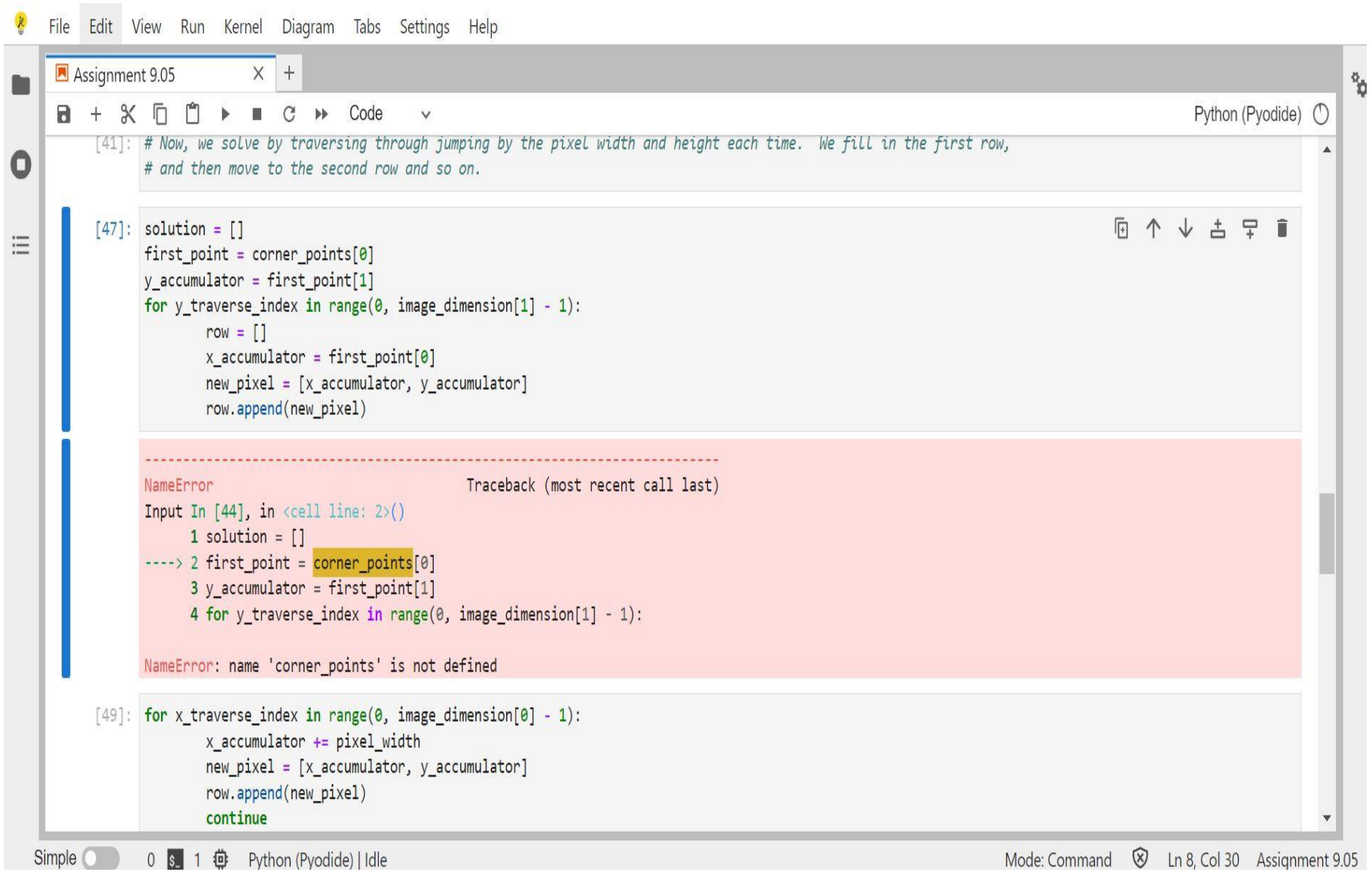
[35]:
```
def find_solution(corner_points, image_dimension):
    """ This method is the master method to find the solution to the main problem.  UN-ORDERED corner points are given
    and a nxmx2 python list is returned.  """
```

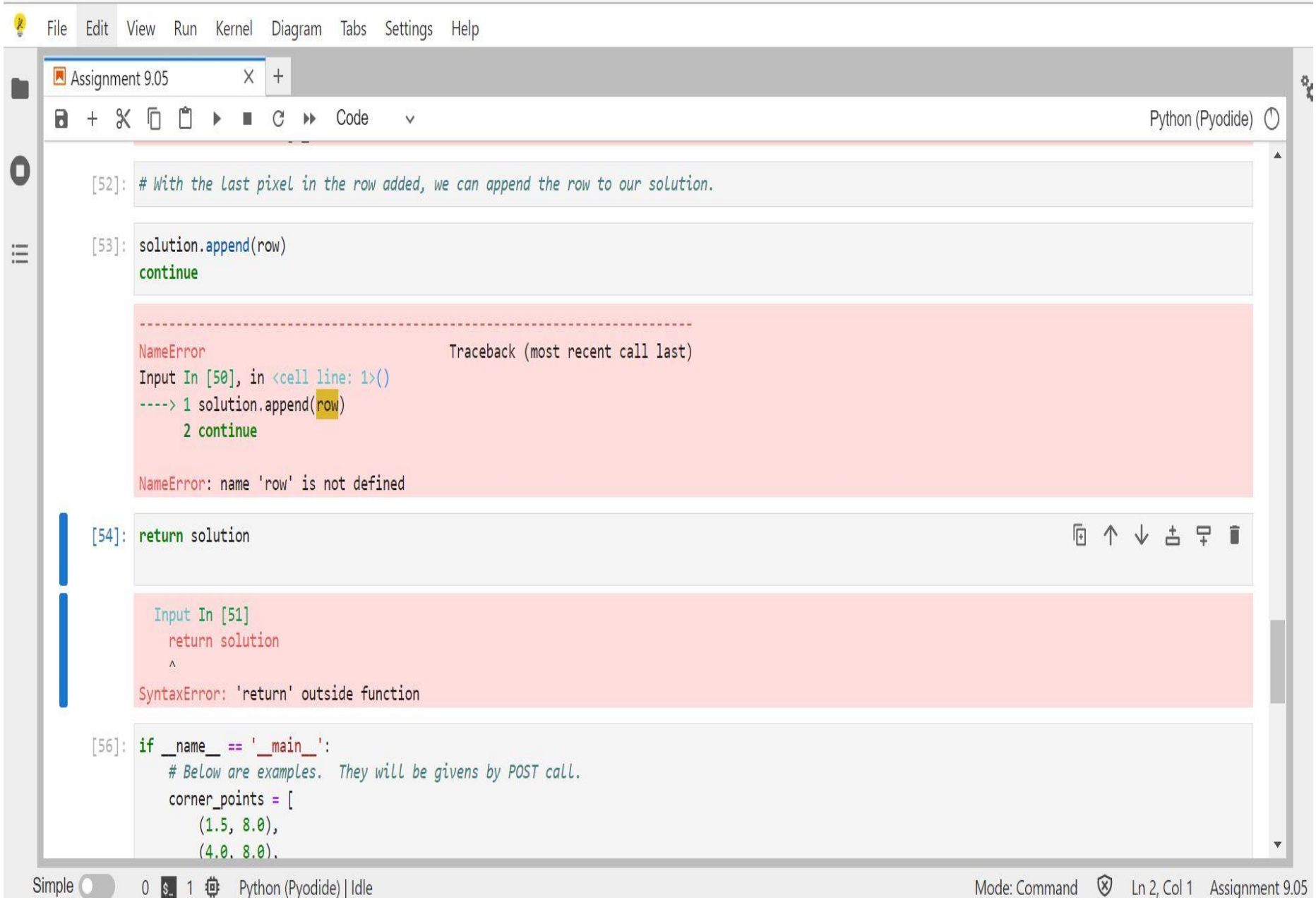
[38]:
```
# First, order the corner points.
corner_points = order_corner_points(corner_points)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [35], in <cell line: 2>()
      1 # First, order the corner points.
----> 2 corner_points = order_corner_points(corner_points)

NameError: name 'corner_points' is not defined
```

[39]:
```
# Next, find the "width" and "height" of each pixel.
pixel_width, pixel_height = get_pixel_dimensions(corner_points, image_dimension)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [36], in <cell line: 2>()
      1 # Next, find the "width" and "height" of each pixel.
----> 2 pixel_width, pixel_height = get_pixel_dimensions(corner_points, image_dimension)

NameError: name 'corner_points' is not defined
```


File Edit View Run Kernel Diagram Tabs Settings Help
Assignment 9.05
Code
Python (Pyodide)
[41]: # Now, we solve by traversing through jumping by the pixel width and height each time. We fill in the first row,
and then move to the second row and so on.
[47]: solution = []
first_point = corner_points[0]
y_accumulator = first_point[1]
for y_traverse_index in range(0, image_dimension[1] - 1):
row = []
x_accumulator = first_point[0]
new_pixel = [x_accumulator, y_accumulator]
row.append(new_pixel)
NameError Traceback (most recent call last)
Input In [44], in <cell line: 2>()
1 solution = []
----> 2 first_point = corner_points[0]
3 y_accumulator = first_point[1]
4 for y_traverse_index in range(0, image_dimension[1] - 1):
NameError: name 'corner_points' is not defined
[49]: for x_traverse_index in range(0, image_dimension[0] - 1):
x_accumulator += pixel_width
new_pixel = [x_accumulator, y_accumulator]
row.append(new_pixel)
continue
Simple
0 $_ 1
Python (Pyodide) | Idle
Mode: Command
Ln 8, Col 30
Assignment 9.05


File Edit View Run Kernel Diagram Tabs Settings Help
Assignment 9.05
Code
Python (Pyodide)
[52]: # With the last pixel in the row added, we can append the row to our solution.
[53]: solution.append(row)
continue
NameError Traceback (most recent call last)
Input In [50], in <cell line: 1>()
----> 1 solution.append(row)
2 continue
NameError: name 'row' is not defined
[54]: return solution
Input In [51]
return solution
^
SyntaxError: 'return' outside function
[56]: if __name__ == '__main__':
# Below are examples. They will be givens by POST call.
corner_points = [
(1.5, 8.0),
(4.0, 8.0),
Simple
0 1
Python (Pyodide) | Idle
Mode: Command
Ln 2, Col 1
Assignment 9.05

File Edit View Run Kernel Diagram Tabs Settings Help

Assignment 9.05

Code

Python (Pyodide)

```
SyntaxError: 'return' outside function
```

[56]:
```
if __name__ == '__main__':
    # Below are examples.  They will be givens by POST call.
    corner_points = [
        (1.5, 8.0),
        (4.0, 8.0),
        (1.5, 1.5),
        (4.0, 1.5)
    ]
    image_dimension = (5, 5)
```

[60]:
```
solution = find_solution(corner_points, image_dimension)
  pretty_print_solution(solution)

print("DONE!")
```

```
  Input In [57]
    pretty_print_solution(solution)
    ^
IndentationError: unexpected indent
```

[ ]:

Simple 0 1 Python (Pyodide) | Idle Mode: Command Ln 8, Col 20 Assignment 9.05