CSci 127: Introduction to Computer Science

hunter.cuny.edu/csci

Today's Topics

- Recap: Parameters & Functions
- Folium

Recap: Input Parameters & Return Values

```
def totalWithTax(food,tip):
    total = 0
    tax = 0.0875
    total = food + food * tax
    total = total + tip
    return(total)

lunch = float(input('Enter lunch total: '))
lTip = float(input('Enter lunch tip:' ))
lTotal = totalWithTax(lunch, lTip)
print('Lunch total is', lTotal)

dinner= float(input('Enter dinner total: '))
dTip = float(input('Enter dinner tip:' ))
dTotal = totalWithTax(dinner, dTip)
print('Dinner total is', dTotal)
```

Formal Parameters

Actual Parameters

- When called, the actual parameter values are copied to the formal parameters.
- All the commands inside the function are performed on the copies.
- The actual parameters do not change.
- The copies are discarded when the function is done.
- The time a variable exists is called its **scope**.

In Pairs or Triples:

- What are the formal parameters? What is returned?

```
def enigma1(x,y,z):
    if x == len(y):
        return(z)
    elif x < len(y):
        return(y[0:x])
    else:
        s = cont1(z)
        return(s+y)
```

```
def cont1(st):
    r = ""
    for i in range(len(st)-1,-1,-1):
        r = r + st[i]
    return(r)
```

(a) `enigma1(7,"caramel","dulce de leche")`

Return: _____

(b) `enigma1(3,"cupcake","vanilla")`

Return: _____

(c) `enigma1(10,"pie","nomel")`

Return: _____

Python Tutor

```
def enigma1(x,y,z):
    if x == len(y):
        return(z)
    elif x < len(y):
        return(y[0:x])
    else:
        s = cont1(z)
        return(s+y)
```

(a) enigma1(7,"caramel","dulce de leche")

Return:

(b) enigma1(3,"cupcake","vanilla")

Return:

(c) enigma1(10,"pie","nomel")

Return:

```
def cont1(st):
    r = ""
    for i in range(len(st)-1,-1,-1):
        r = r + st[i]
    return(r)
```

(Demo with `pythonTutor`)

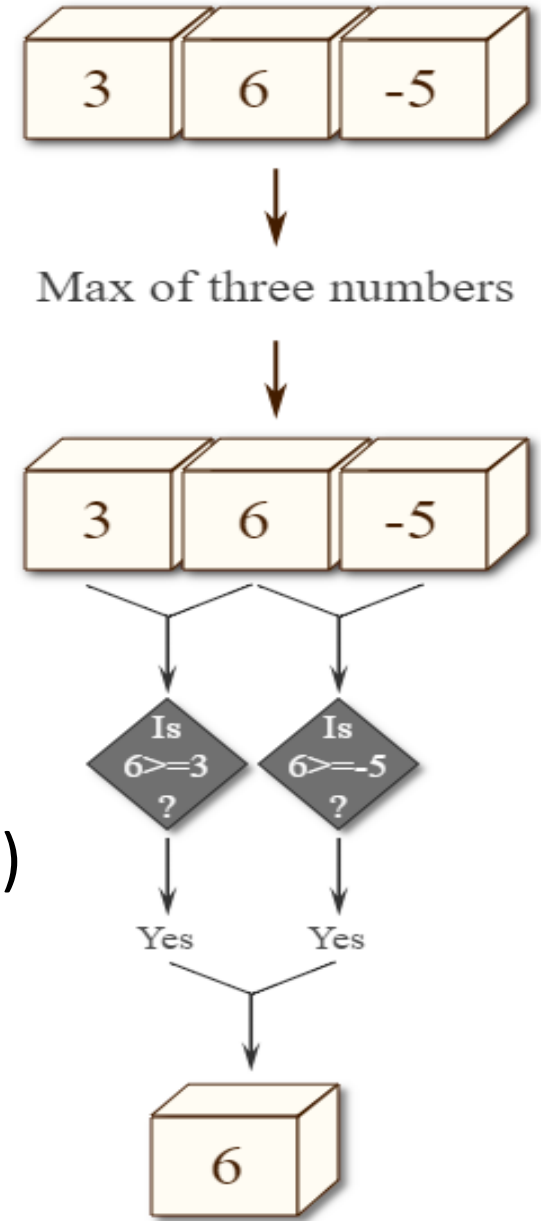
Python Exercises

Exercise – 1 (5 min)

Write a Python function to find the Max of three numbers.

Python Exercises

Exercise – 1 (5 min)

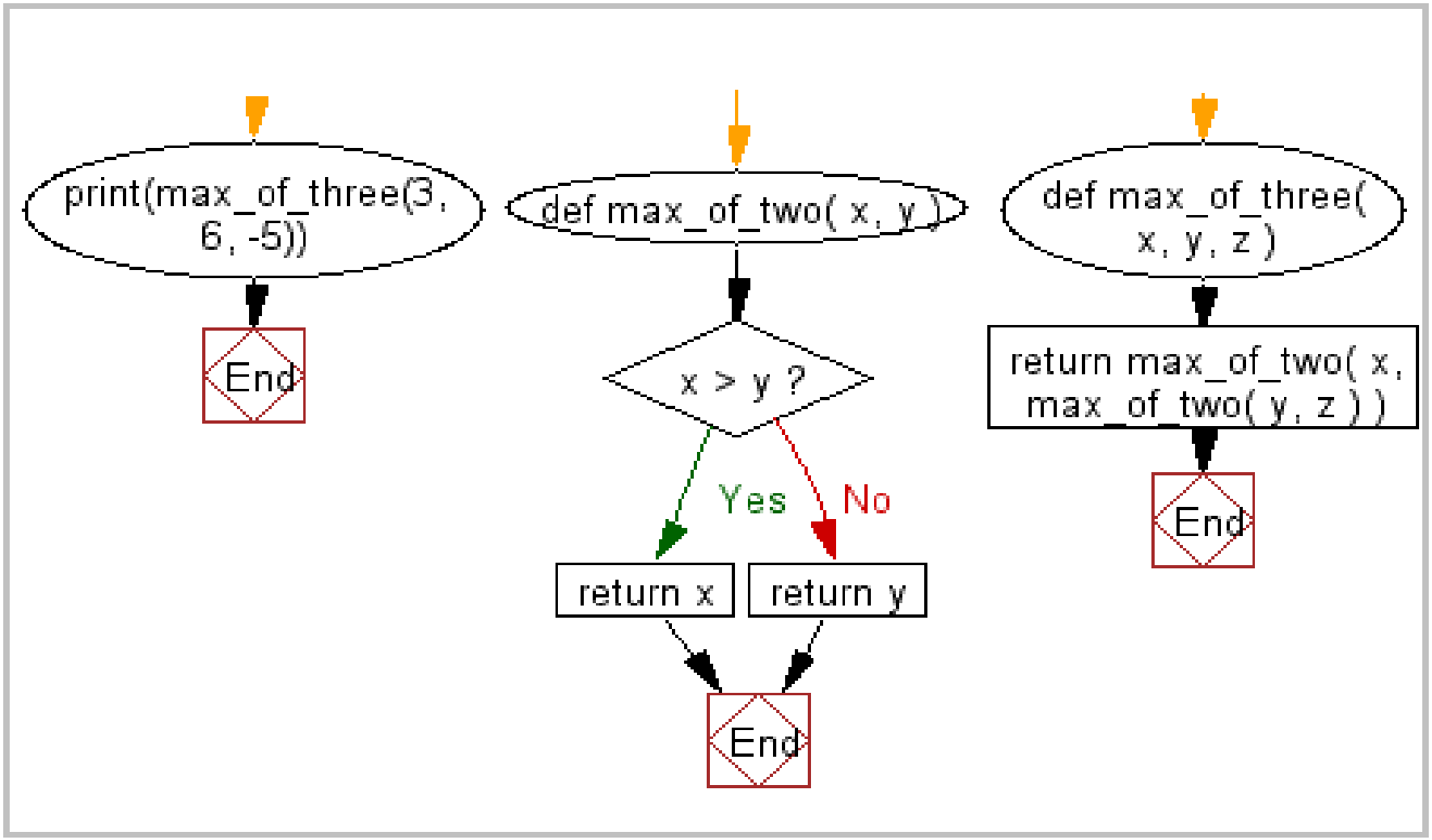



```
def max_of_two( x, y ):
    if x > y:
        return x
    return y
def max_of_three( x, y, z ):
    return max_of_two( x, max_of_two( y, z ) )

print(max_of_three(3, 6, -5))
```

Python Exercises

Exercise – 1 (5 min)



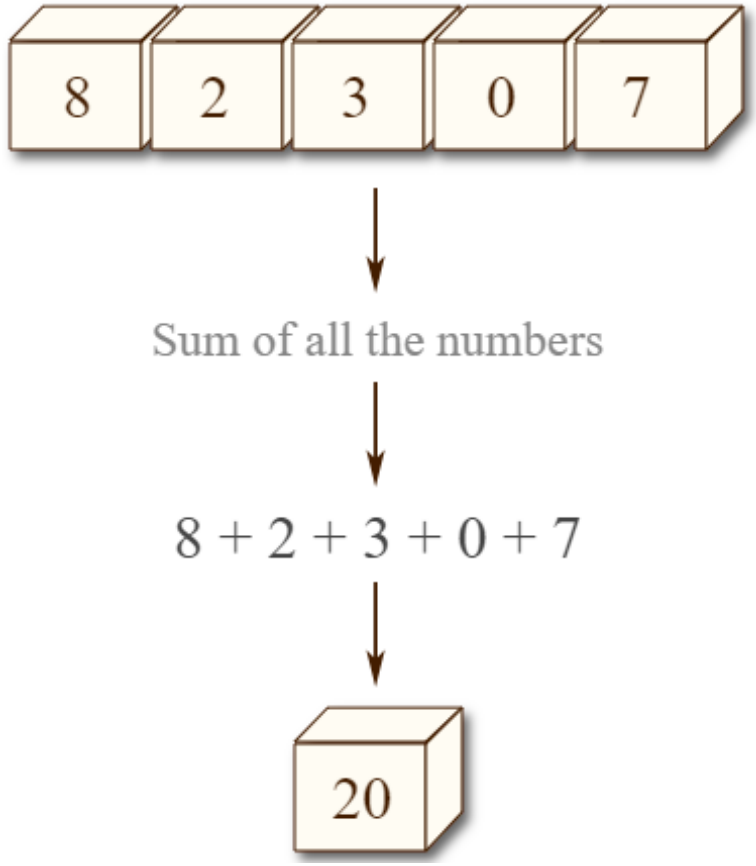
Python Exercises

Exercise – 2 (5 min)

Write a Python function to sum all the numbers in a list

Python Exercises

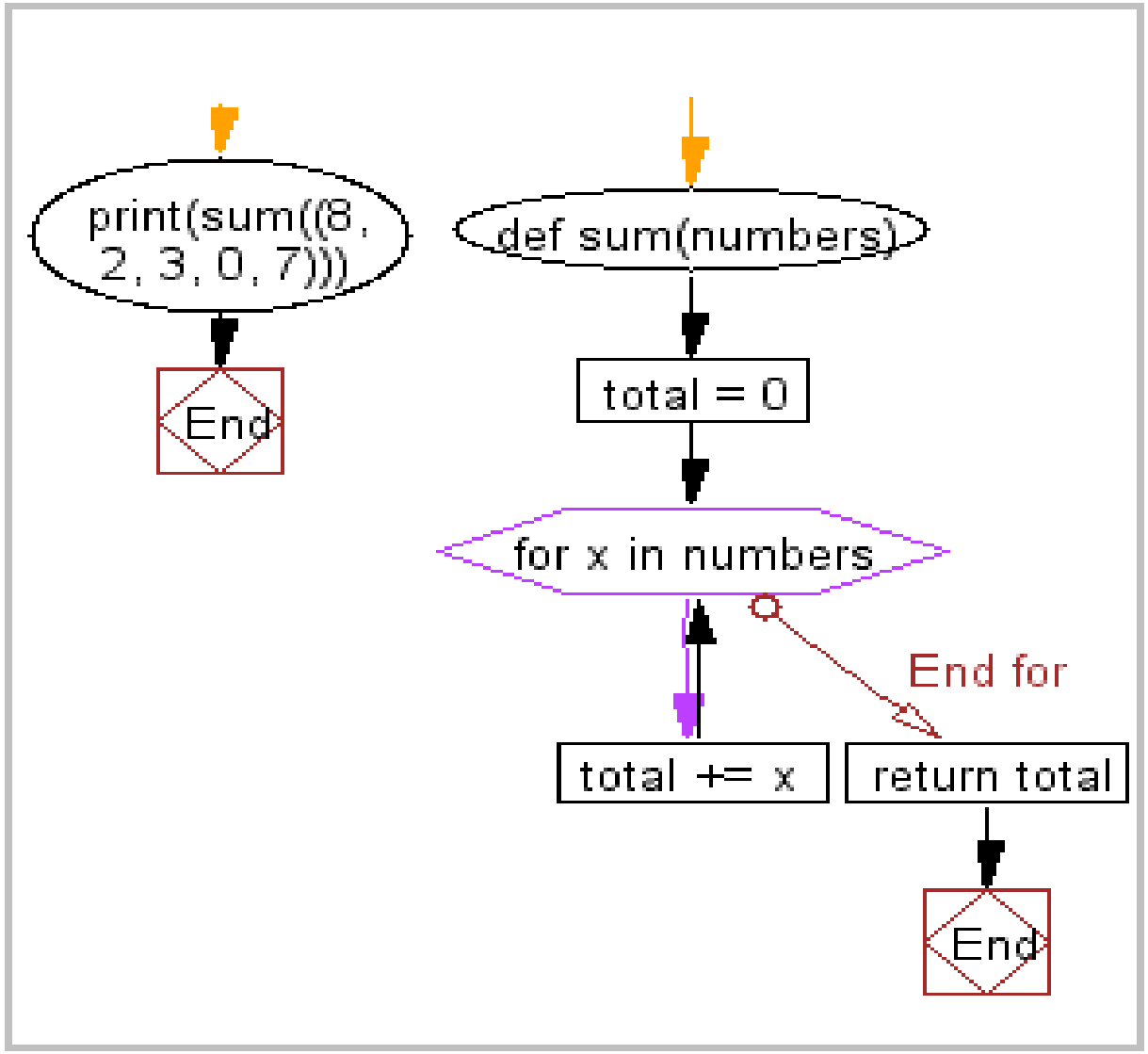
Exercise – 2 (5 min)




```
def sum(numbers):
    total = 0
    for x in numbers:
        total += x
    return total
print(sum([8, 2, 3, 0, 7]))
```

Python Exercises

Exercise – 2 (5 min)



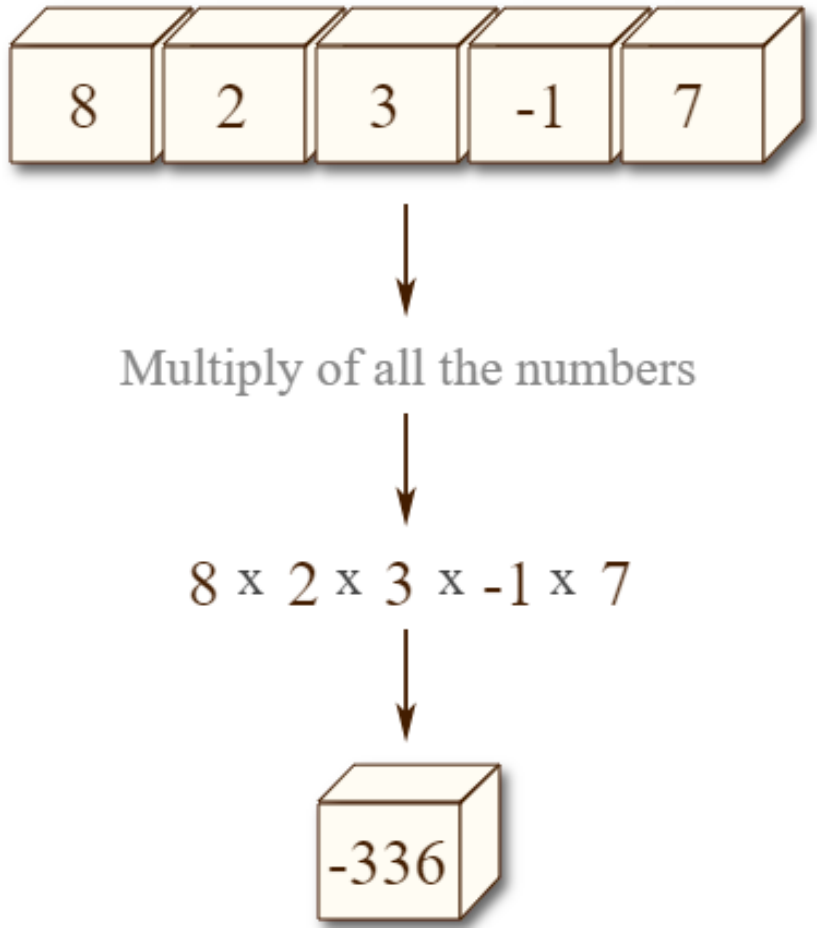
Python Exercises

Exercise – 3 (5 min)

Write a Python function to multiply all the numbers in a list

Python Exercises

Exercise – 3 (5 min)




```
def multiply(numbers):
    total = 1
    for x in numbers:
        total *= x
    return total
print(multiply([8, 2, 3, -1, 7]))
```

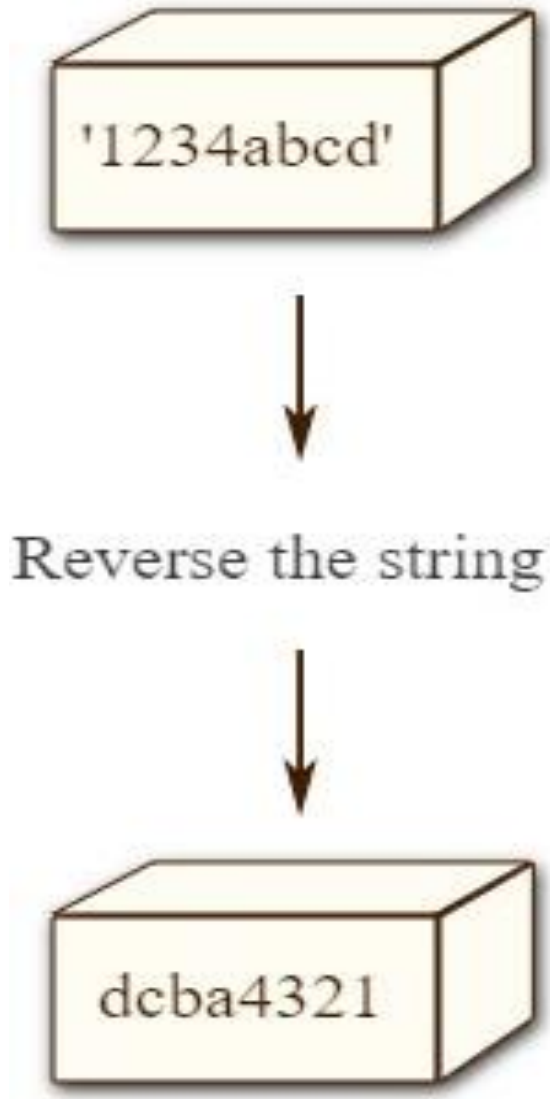
Python Exercises

Exercise – 4 (10 min)

Write a Python program to reverse a string

Python Exercises

Exercise – 4 (Solution)

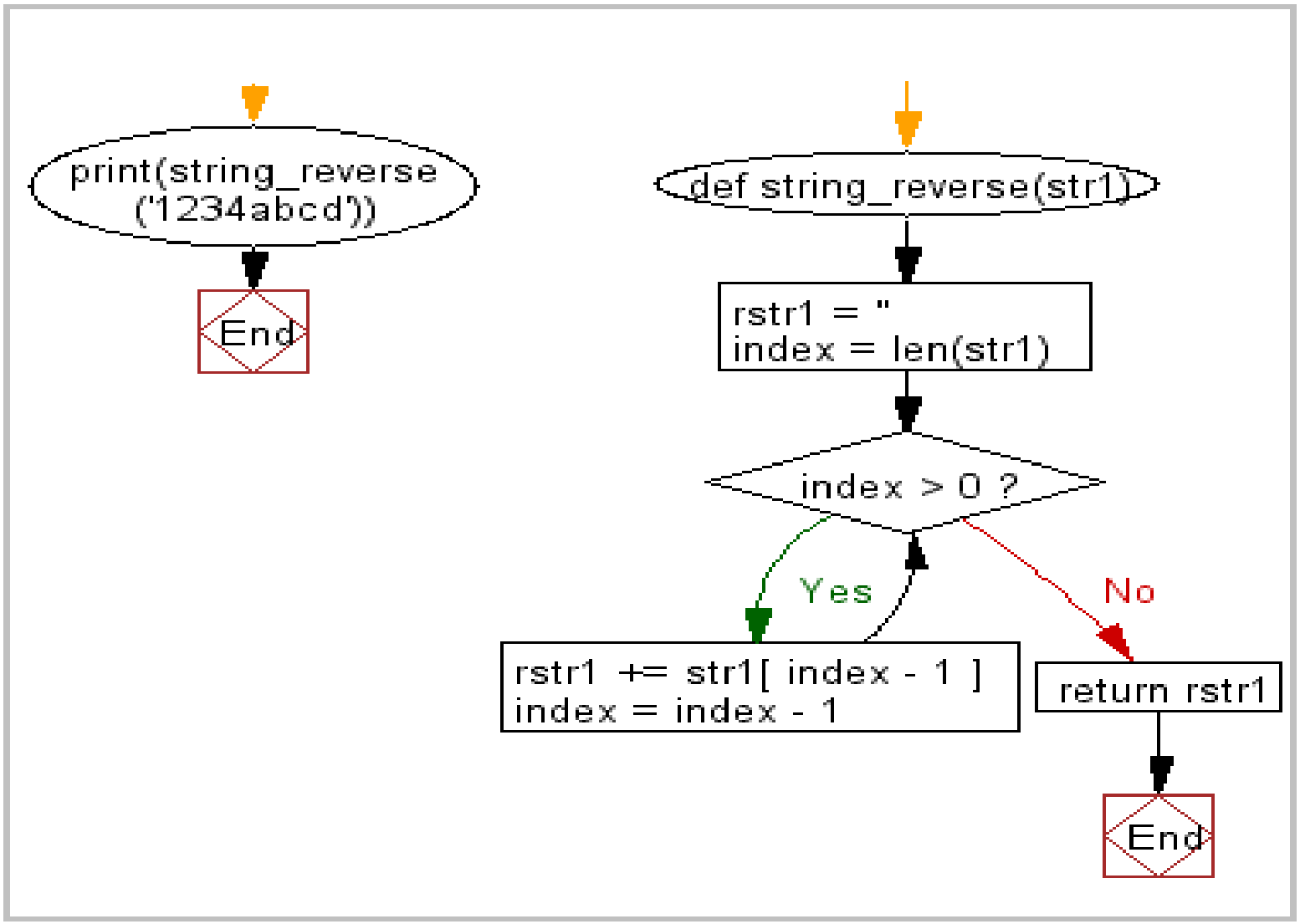



```
def string_reverse(str1):

    rstr1 = ''
    index = len(str1)
    while index > 0:
        rstr1 += str1[ index - 1 ]
        index = index - 1
    return rstr1
print(string_reverse('1234abcd'))
```

Python Exercises

Exercise – 4 (Solution)



In Pairs or Triples:

- Write the missing functions for the program:

```
def main():
    tess = setUp()      #Returns a purple turtle with pen up.
    for i in range(5):
        x,y = getInput()        #Asks user for two numbers.
        markLocation(tess,x,y) #Move tess to (x,y) and stamp.
```

Group Work: Fill in Missing Pieces

```
def main():
    tess = setUp()     #Returns a purple turtle with pen up.
    for i in range(5):
        x,y = getInput()        #Asks user for two numbers.
        markLocation(tess,x,y) #Move tess to (x,y) and stamp.
```

Third Part: Fill in Missing Pieces

1. Write import statements.
2. Write down new function names and inputs.
3. Fill in return values.
4. Fill in body of functions.

```
import turtle
def setUp():
    newTurtle = turtle.Turtle()
    newTurtle.penup()
    return(newTurtle)
def getInput():
    x = int(input('Enter x: '))
    y = int(input('Enter y: '))
    return(x,y)
def markLocation(t,x,y):
    t.goto(x,y)
    t.stamp()
def main():
    tess = setUp()     #Returns a purple turtle with pen up.
    for i in range(5):
        x,y = getInput()        #Asks user for two numbers
        markLocation(tess,x,y).
```

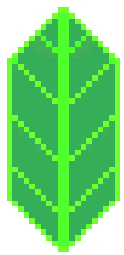
What is Folium ?

- A module for making HTML maps.
- It's a Python interface to the popular `leaflet.js`.
- Outputs `.html` files which you can open in a browser.
- An extra step:

Write code. → *Run program.* → *Open .html in browser.*

Plotting Maps with Folium

Folium



- install folium:
  $ pip install folium

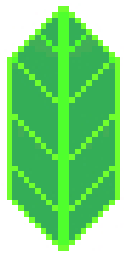
Plotting maps with Folium is easier than you think.
Folium provides the **folium.Map()** class:

- It can take a location parameter in terms of latitude and longitude and generates a map around it.

I.e to plot a map of New York City with latitude and longitude as
40.730610 and -73.935242 respectively:

Plotting Maps with Folium

Folium



- To use:

```
import folium
```

- Create a map:

```
m = folium.Map(location=[40.730610, -73.935242])
```

- Write map to html file to view:

```
m.save(outfile="index.html")
```

Folium

To use:

- `import folium`

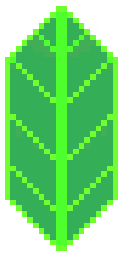
Create a map:

- `m = folium.Map(location=[40.730610, -73.935242])`

Change map design (tiles *(str, default 'OpenStreetMap')* **:**

- `folium.TileLayer('Stamen Terrain').add_to(m)`
- Many options to customize background map ("tiles") – can be passed on Map initialization i,e Map(tiles = `'Stamen Terrain', location=[40.730610, -73.935242]`)

Folium



Plotting Markers on the Map

- Markers are the items used for marking a location on a map
- i.e when you use Google Maps for navigation, your location is marked by a marker and your destination is marked by another marker.
- Folium gives a folium.Marker() class for plotting markers on a map
- Just pass the latitude and longitude of the location, mention the popup and tooltip and add it to the map.

-

- Make markers:

```
newMark = folium.Marker([lat,lon],popup=name)
```

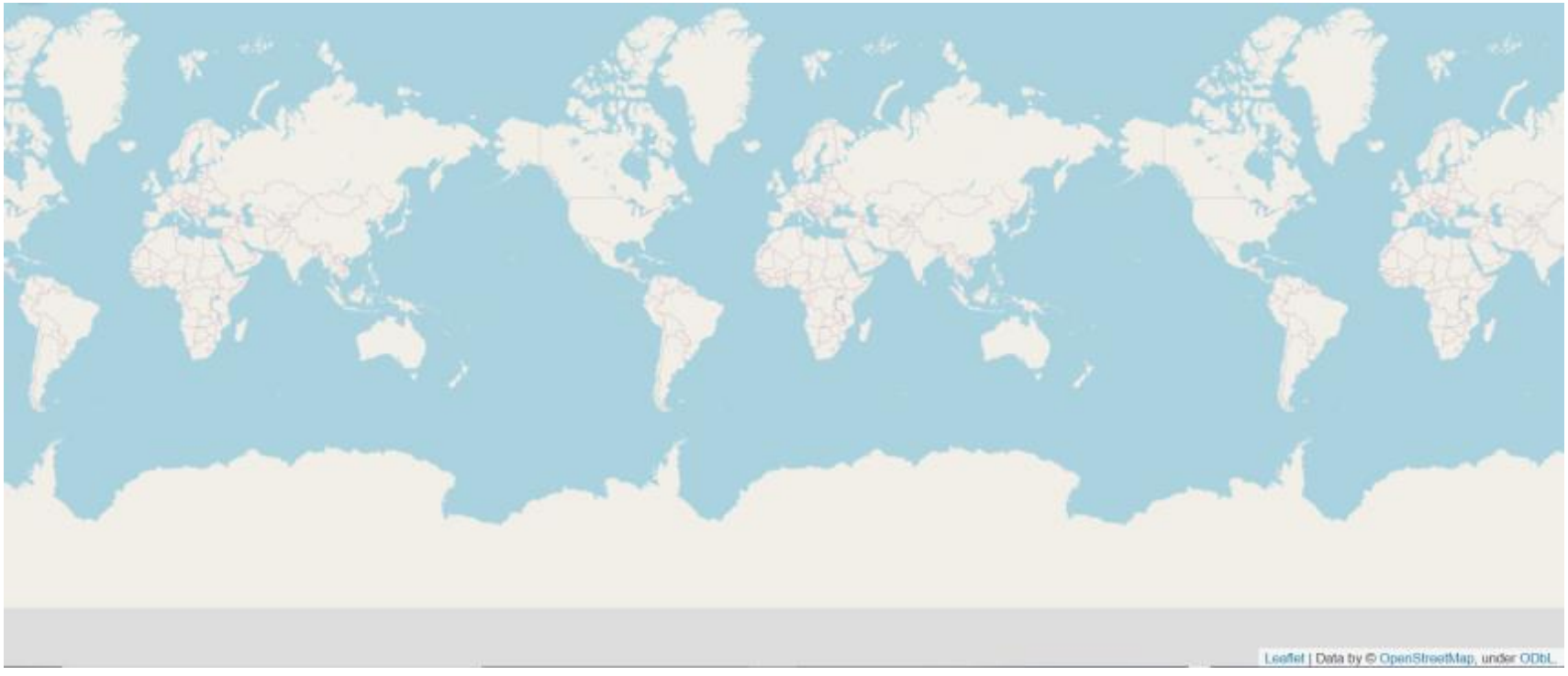
Add to the map:

```
newMark.add to(myMap)
```

Many options to customize background map ("tiles") and markers.

Plotting Maps with Folium

```
folium.Map()
```



In Pairs of Triples

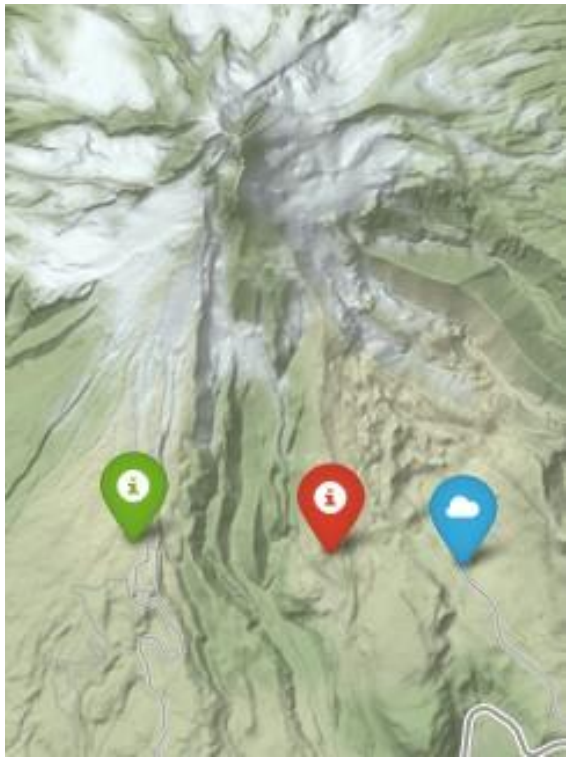
- Predict which each line of code does:

```
m = folium.Map(
    location=[45.372, -121.6972],
    zoom_start=12,
    tiles='Stamen Terrain'
)

folium.Marker(
    location=[45.3288, -121.6625],
    popup='Mt. Hood Meadows',
    icon=folium.Icon(icon='cloud')
).add_to(m)

folium.Marker(
    location=[45.3311, -121.7113],
    popup='Timberline Lodge',
    icon=folium.Icon(color='green')
).add_to(m)

folium.Marker(
    location=[45.3300, -121.6823],
    popup='Some Other Location',
    icon=folium.Icon(color='red', icon='info-sign')
).add_to(m)
```



(example from Folium documentation)

Code Reuse

- Goal: design your code to be reused.
- Example: code to make maps of CUNY locations from CSV files (see lab)
  - Same idea can be used for mapping traffic collisions data.
  - Or recycling bins, or wifi locations, or 311 calls,...
  - Small wrinkle: some call the columns "Latitude", while others use "LATITUDE", "latitude", or "lat".
  - Solution: ask user for column names and pass as parameters.