COMP1021
Introduction to Computer Science

Creating Turtle Objects

David Rossiter and Gibson Lam

Outcomes

- After completing this presentation, you are expected to be able to:
  1. Explain the concept of an object
  2. Create new turtle objects
  3. Execute functions in a turtle object
  4. Access information about a turtle object

The Turtle

```
import turtle
turtle.forward(100)
turtle.left(120)
turtle.forward(100)
turtle.left(120)
turtle.forward(100)
turtle.left(120)
turtle.done()
```

- So far, we have used lots of code like this
- One turtle was used
- The turtle is actually a *turtle object*
- We will discuss objects at a deeper level later in the course
- In this presentation we do an introduction

- There's always one turtle object, which we sometimes call the *default turtle* – that's what we have been using since the start of the course

A Turtle Object

- Below you can see the basic idea of a turtle object
- Inside an object there are variables and functions

Variables	e.g.	*x position of the turtle*
		y position of the turtle
		angle of the turtle
		pen colour of the turtle
		. . .
Functions	e.g.	`forward()`
		`backward()`
		`left()`
		`right()`
		`...`

- When you talk about objects, variables are often called 'attributes' or 'properties' and functions are called 'methods'

- Every turtle object has this same structure, including any new turtle objects you create

Creating a New Turtle Object

- This is how you create a new turtle object:

```
newTurtle = turtle.Turtle()
```

- After the above code `newTurtle` is a new turtle
- After you create the new turtle object you can use all the techniques you know about e.g.

```
newTurtle.forward(100)
newTurtle.left(90)
newTurtle.color("red")
```

and so on

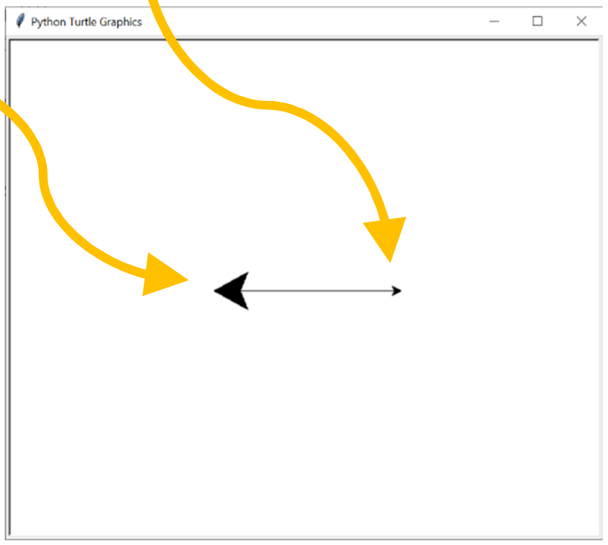
Simple Example

```
import turtle
turtle.forward(100)
```

} Use the default turtle

```
t=turtle.Turtle()
t.shapesize(4, 4)
t.left(180)
t.forward(100)
```

} Create a new turtle, make it bigger, rotate it 180 degrees, move it forward



You can see 2 turtles in the turtle window

```
turtle.done()
```

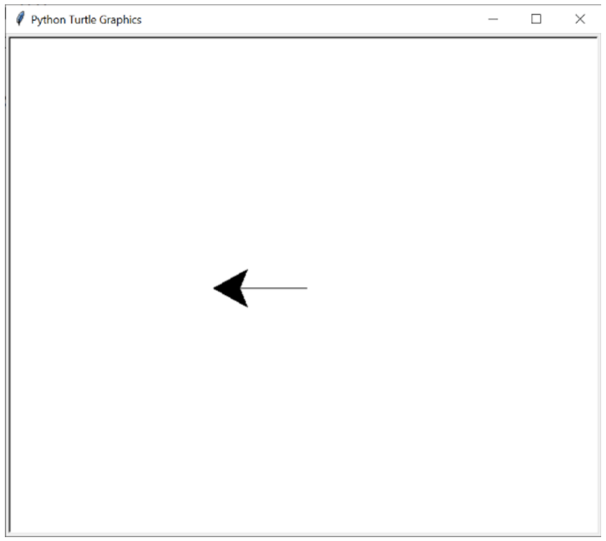
} Need this at the end

- Don’t forget that turtles always start in the middle of the screen

Hiding the Default Turtle

```
import turtle

turtle.hideturtle()
```
} Sometimes you only want the newly created turtle(s) – you can hide the default turtle

```
t=turtle.Turtle()

t.shapesize(4, 4)

t.left(180)

t.forward(100)
```
} Create a new turtle, make it bigger, rotate it 180 degrees, move it forward



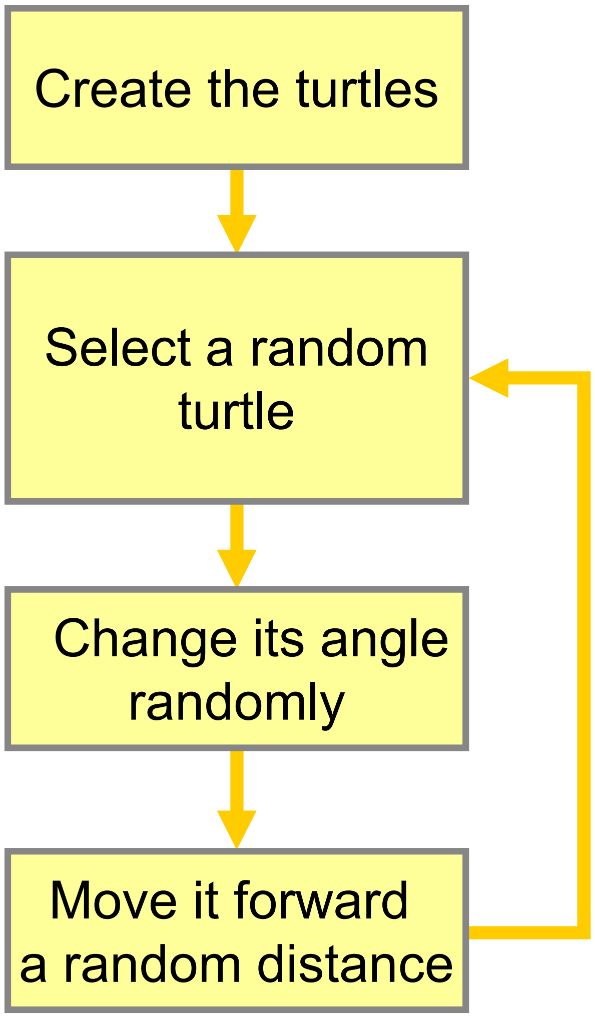
You can see 1 turtle in the turtle window

```
turtle.done()
```
} Need this at the end

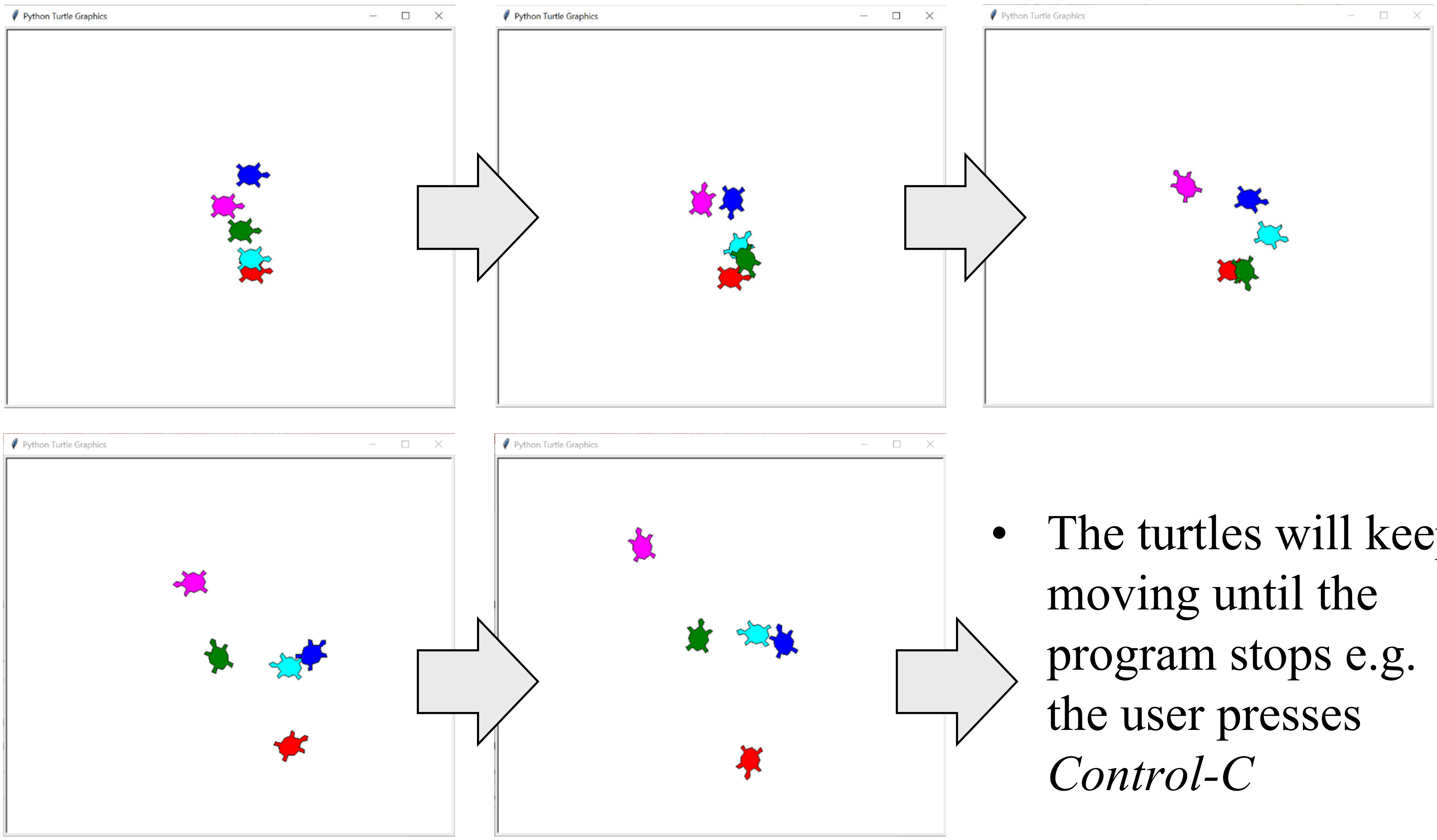
- Don't forget that turtles always start in the middle of the screen

A Demonstration Using 5 Turtles

- Now we will build a demonstration which creates and uses 5 turtles
  - Most of the properties of the 5 turtles are the same, except for the colour
- After making the turtles, an infinite loop is used:
  - One of the turtles is randomly selected
  - That turtle has its angle changed randomly
  - That turtle is moved forward a random distance



Running The Program



- The turtles will keep moving until the program stops e.g. the user presses *Control-C*

Using a List

- To better manage the turtles we store them in a list, to make a list of turtle objects
- We start with an empty list:

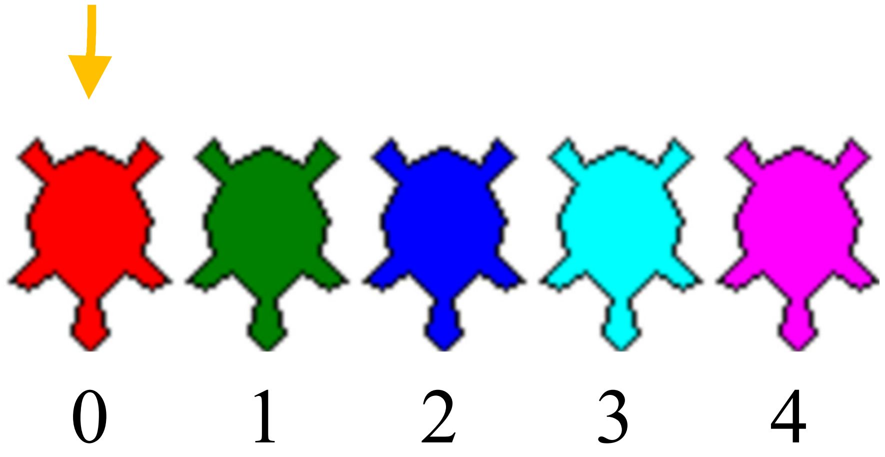
```
allTurtles = []
```

- Then, after we create each new turtle, we add it to the list of turtles using `append`

```
newTurtle = turtle.Turtle()
allTurtles.append(newTurtle)
```

Accessing Turtles in a List

- As you know, we can retrieve something from a list by using the index e.g. *name_of_list*`[2]`
- This is true whatever is in the list, even a turtle
- For example, to access the first item in the list we can use `allTurtles[0]`



Creating 1 Turtle Object

```
# Create a new turtle, set its parameters, add it to the list
def createOneTurtle(thisColor):
    thisTurtle = turtle.Turtle()    # Make a new turtle object
    thisTurtle.fillcolor(thisColor)# Set the color of the turtle
    thisTurtle.shape("turtle")      # Make it look like a turtle
    thisTurtle.shapesize(2, 2)      # Make the turtle twice as big
    thisTurtle.up()                 # Do this so no line drawn
    thisTurtle.goto(random.randint(-80, 80),
                    random.randint(-80, 80)) # Starting position
    allTurtles.append(thisTurtle)  # Append the turtle to the list
```

- This function creates one turtle object
- The turtle is added to a list, so it can be easily accessed and managed later

- See the next slide for examples of how the function is used

Creating all the Turtle Objects

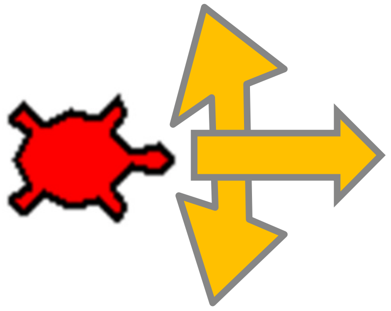
```
def createOneTurtle(thisColor):
    . . . see previous slide . . .

# The main part of the program
allTurtles = [] # An empty list
createOneTurtle("red")
createOneTurtle("green")
createOneTurtle("blue")
createOneTurtle("cyan")
createOneTurtle("magenta")
. . .
```

Create 5 turtles, each with a different colour

- For our demonstration program we create 5 turtle objects, using the function shown on the previous slide

Changing a Turtle Object

- To make the turtles look alive we repeatedly select a random turtle, change it to a random angle, and move it a random distance



```
def changeOneTurtle():
    index = random.randint(0, 4)
    allTurtles[index].left(
        random.randint(-90, 90) ) # Change angle
    allTurtles[index].forward(
        random.randint(10, 15) )  # Move forward

# The main part of the program
. . .
while True:
    changeOneTurtle() # Repeatedly change a turtle
```

Getting Info From a Turtle Object

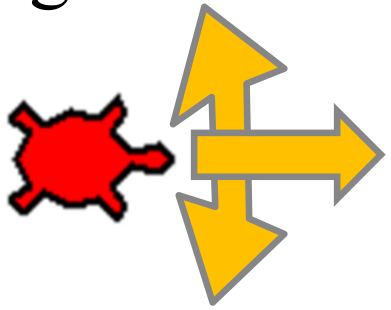
- If you have a turtle object called e.g. `thisTurtle` then you can get information from it like this:

`result = thisTurtle.xcor()`	Get the x position value
`result = thisTurtle.ycor()`	Get the y position value
`result = thisTurtle.position()`	Get both x and y
`result = thisTurtle.heading()`	Get the turtle angle
`result = thisTurtle.fillcolor()`	Get the fill color
`result = thisTurtle.speed()`	Get the speed
`result = thisTurtle.shape()`	Get the shape

… other information can also be extracted from a turtle object …

Showing Turtle Information

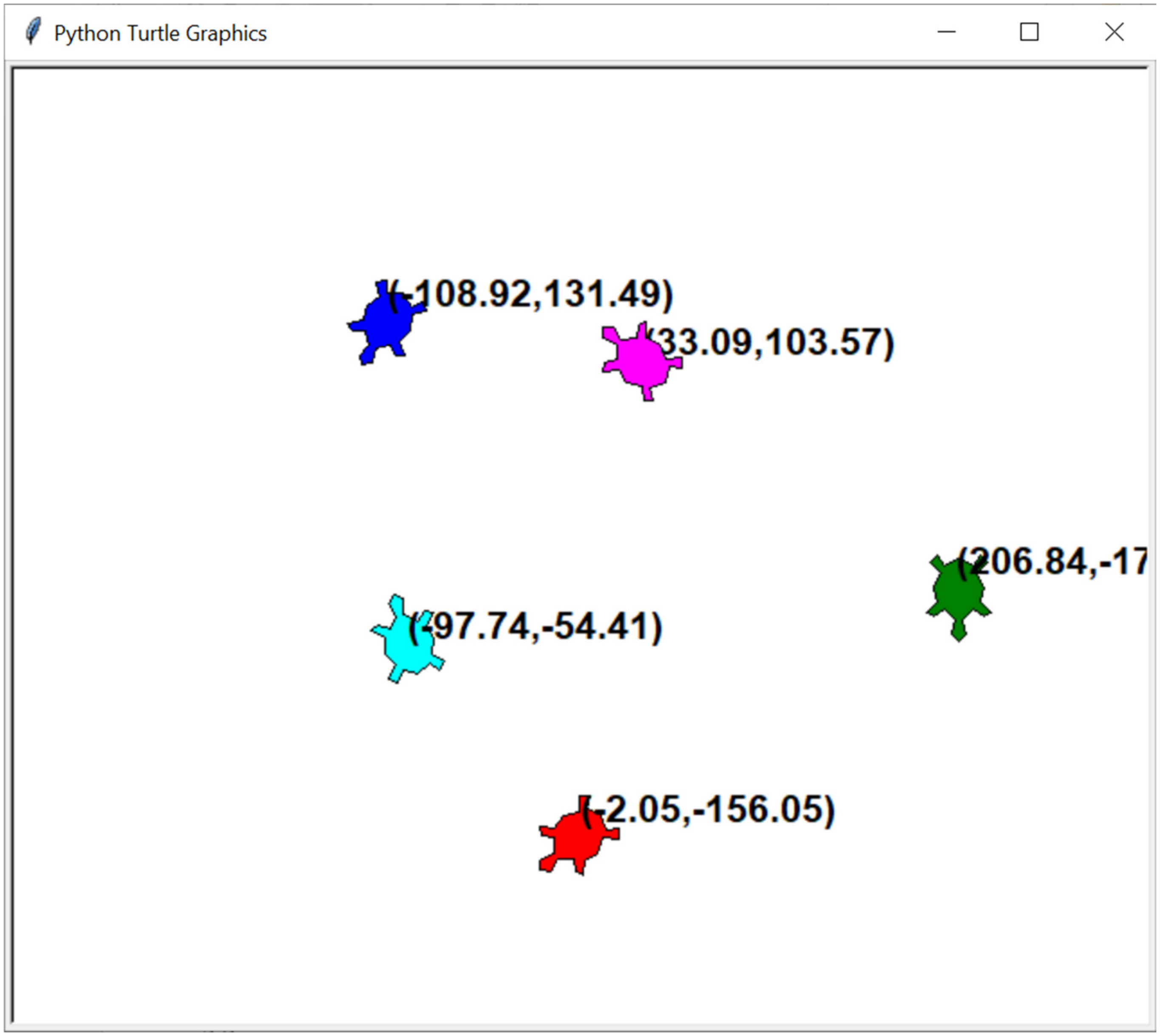
- Let's extend the previous example so that the position of the turtle is shown after it is changed:



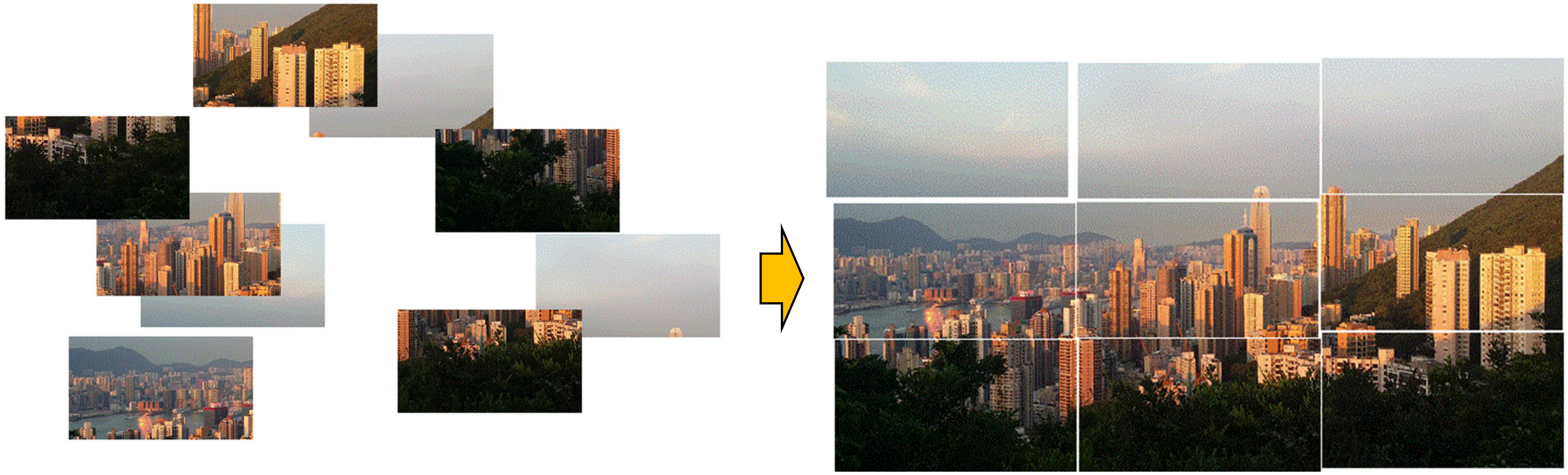
```
def changeOneTurtle():
    index = random.randint(0, 4)
    allTurtles[index].left(
        random.randint(-90, 90) ) # Change angle
    allTurtles[index].forward(
        random.randint(10, 15) )  # Move forward
    allTurtles[index].clear() # Clear previous text
    allTurtles[index].write( \
      str( allTurtles[index].position() ), \
      font=("Arial", 16, "bold") ) # Show position
```

These 2 lines of code are added

Example Program Display



Another Example - Jigsaw



- Each turtle appears as a small image
- Click and drag the turtles to assemble the jigsaw

Getting Ready

```
import turtle
import random

totalRows = 3
totalColumns = 3

# Main part of the program
allTurtles=[] # We will store all the turtles here


createJigsaw() # Create jigsaw pieces


# Keep checking if anything is dragged,
# if so, execute the appropriate function
turtle.done()
```

$\}$ *This function code is shown on the next slide*

```
for row in range(totalRows):                def createJigsaw():
    for column in range(totalColumns):
        newTurtle = turtle.Turtle() # Make turtle object
        newTurtle.up() # No line when turtle moves
        newTurtle.speed(0) # Fast movement

        x = random.randint( -int(turtle.window_width()/2),
            int(turtle.window_width()/2) )
        y = random.randint(-int(turtle.window_height()/2),
            int(turtle.window_height()/2) )
        newTurtle.goto(x, y) # Move to random position

        thisFilename="image-" + str(row) + "-" +  \
            str(column) + ".gif" # Example "image-2-1.gif"
        turtle.addshape(thisFilename)# Add image to system
        newTurtle.shape(thisFilename) # Apply to turtle

        newTurtle.ondrag(newTurtle.goto) # Move when drag
        allTurtles.append(newTurtle) # Add to list
```