Introduction to Python - North Salem High School

David Moste

Unit	Intro to Programming
Lesson	**Topic:** Creating Functions
Methods + Reasoning	**Pair Programming** The goal of the pair programming in this lesson is to give students the opportunity to both work with a new format for code (writing functions) and deal with a new abstract concept. As the driver in pair programming, students will be able to grapple with the syntax and formatting of writing and calling functions. As the navigator in pari programming, students will be able to wrestle with some of the abstract concepts involved in creating and calling functions. **Scaffolding** Scaffolding will be used to remove some of the unnecessary abstraction from the task at hand. Since the real goal of this lesson is to get students comfortable with writing and calling functions, scaffolded code will be provided which will eliminate the need for students to think about some of the outside concepts that aren't central to the lesson objectives.
Objective	Students will be able to: ● Write a function to simplify their code in a meaningful way.
Accompanying Documents	Scaffolded Base Staircase Solution

Lesson Plan	
Warm Up	**5 mins**
Prompt: Explain the roles involved in pair programming. Include an example of why each role is important. **Discuss:** Provide students time to discuss their ideas with their neighbors before sharing with the class. This can be done fairly quickly.	

Lesson Body	**35 mins**
Staircase - Pair Programming + Scaffolding	**35 mins**

Pair Programming + Scaffolding: At this point in the year, students should be relatively familiar with pair programming. However, since it is still early, it is important to remind students of the roles they will be fulfilling (driver and navigator) as well as the importance of each role. The teacher should remind the students that each role has a particular purpose in getting the code written **AS WELL AS** a particular purpose in helping them learn and master the content. Navigator should be grappling with the abstraction of the algorithm while drivers should be focused on the syntax and proper writing of the code.

The students will be provided with some base code that will hopefully eliminate some of the unrelated abstraction that could get in the way of the lesson objective. This base code can be found here. A solution can be found here.

Students will work in repl.it for this activity. They will have access to a built-in object called Arrow() as well as a built-in object called Board().

The goal of this activity is to get students to both gain practice building and designing functions as well as to see the value. It is relatively easy to build a step, but much harder to code every line of a staircase. This exercise should show students how functions can help reduce the complexity of a program.

Scaffolded Base

```
from Arrow import *

arrow = Arrow()

## Build a function that creates a single step.
## A step looks like the following:
## ----
## |
## |
def step():
  ## Replace with step() code
  pass

## Build a function that creates a staircase
## A staircase looks like the following:
##         ----
##         |
##     ----|
##     |
## ----|
## |
## |
def staircase():
  ## Replace with staircase() code
  pass

## CHALLENGE
## Build a function that creates an entire diamond.
## This should be four sides placed together.
def diamond():
  ## Replace with diamond() code
  pass

def main():
  arrow.printRoute()

main()
```

Staircase Solution

```
from Arrow import *

arrow = Arrow()

## Build a function that creates a single step.
## A step looks like the following:
## ----
## |
## |
def step():
  ## Replace with step() code
  arrow.forward()
  arrow.forward()
  arrow.right()
  arrow.forward()
  arrow.forward()
  arrow.forward()
  arrow.forward()

## Build a function that creates a staircase
## A staircase looks like the following:
##          ----
##          |
##      ----|
##      |
## ----|
## |
## |
def staircase():
  ## Replace with staircase() code
  step()
  arrow.left()
  step()
  arrow.left()
  step()

## CHALLENGE
## Build a function that creates an entire diamond.
## This should be four sides placed together.
def diamond():
  ## Replace with diamond() code
  pass

def main():
  arrow.left()
  staircase()
  arrow.printRoute()

main()
```

Arrow.py

```
from Board import *

class Arrow:
  def __init__(self):
    self.x_loc = 0
    self.y_loc = 24
    self.direction = "→"
    self.board = Board()
    self.board.addCharacter(self.direction, self.x_loc, self.y_loc)

  def printRoute(self):
    self.board.printBoard()

  def forward(self):
    if self.direction == "→":
      self.board.addCharacter("-", self.x_loc, self.y_loc)
      self.x_loc += 1
      self.board.addCharacter(self.direction, self.x_loc, self.y_loc)
    elif self. direction== "←":
      self.board.addCharacter("-", self.x_loc, self.y_loc)
      self.x_loc -= 1
      self.board.addCharacter(self.direction, self.x_loc, self.y_loc)
    elif self. direction== "↓":
      self.board.addCharacter("|", self.x_loc, self.y_loc)
      self.y_loc += 1
      self.board.addCharacter(self.direction, self.x_loc, self.y_loc)
    elif self. direction== "↑":
      self.board.addCharacter("|", self.x_loc, self.y_loc)
      self.y_loc -= 1
      self.board.addCharacter(self.direction, self.x_loc, self.y_loc)

  def left(self):
    if self.direction == "↓":
      self.direction = "→"
    elif self.direction ==  "←":
      self.direction = "↓"
    elif self.direction == "↑":
      self.direction = "←"
    elif self.direction == "→":
      self.direction = "↑"

    self.board.addCharacter(self.direction, self.x_loc, self.y_loc)

  def right(self):
    self.left()
    self.left()
    self.left()
```

Board.py

```
class Board:
  def __init__(self):
    self.board = []
    for i in range(0, 50):
      self.board.append([])
      for j in range(0, 50):
        self.board[i].append(" ")

  def printBoard(self):
    for i in range(0, 50):
      line = ""
      for j in range(0, 50):
        line += self.board[i][j]
      print(line)

  def addCharacter(self, character, x_loc, y_loc):
    self.board[y_loc][x_loc] = character
```