**Discussion 1**

The following Pseudocode read in 1o students’ scores and calculate the average.

Initialize total to zero (Must initialize variable before use)

Initialize student\_counter to one (if counter starts from zero, will be 11 students instead of 10)

While student\_counter is less than or equal to ten

Input the next score

Add the score into the total

Add one to student\_counter (update looping control variable)

EndWhile

Set the class average to the total divided by ten

**Discussion 2**

Initialize passes to zero

Initialize failures to zero

Initialize student\_counter to one

While student\_counter is less than or equal to ten

Input the next score

If the student passed

Add one to passes

else

Add one to failures

EndIf

add one to student counter

EndWhile

print the number of passes

print the number of failures

Discussion 3

**num : 1**

**FOR num -> 1 to 20**

**IF num MOD 15 ===0**

**PRINT “FizzBuzz”**

**ELSE IF num MOD 3 ===0**

**PRINT “Fizz”**

**ELSE IF num MOD 5===0**

**PRINT “Buzz”**

**ELSE**

**PRINT num**

**ENDIF**

**ENDFOR**

Discussion 4

**c = 10**

**valid, Python statement**

**7 = a**

**Invalid, LHS cannot be a literal**

**a = d**

**invalid, because d hasn’t been created/defined**

**a = c + 1**

**valid, Python statement (LHS is a variable, RHS is a expression)**

**a + c = c**

**invalid, because LHS cannot be an expression, should be a variable**

**3 + a**

**valid, Python expression**

**7up = 10**

**invalid, Python naming convention: cannot start a variable name with digits**

**must start with a leter or underscore character (\_);**

**import = 1003**

**invalid, import is a reserved word (known as keyword) in Python**

**b = math.pi \* c**

**invalid, we need to first “import math” for math module**

**int = 500**

**valid, int is not a keyword in python, although it can be used to cast non-integer data type to integer. Not recommend to be used as name of variable to avoid ambiguity. (similarly for float, input etc)**

**a \*\* 3**

**valid, Python expression- \*\* is an arithmetic operator for exponential**

**a,b,c = c,1,a**

**valid, Python statement for multiple assignment**

**b,c,a = a,b**

**invalid, different number of elements on the two sides: LHS and RHS**

**c = b = a = 7**

**valid, Python allows chained assignment even though b is just created in the middle**

**print( A )**

**invalid, A not defined (Python: case sensitive)**

**print( "b\*b + a\*a = c\*c" )**

**valid, Python statement**

**print( ‘A’ )**

**valid, Python treats single quotes to be the same as double quotes**

**print( "c" = 1 )**

**invalid, because we cannot assign 1 to c and then do print in a single Python statement**

**Discussion 5**

A

Start

Declare Integer boy, girl, total

Declare Float percent\_boy, percent\_girl

Input boy

Input girl

total = boy + girl

A

percent\_boy = boy / total

percent\_girl = girl / total

Display “Boy:”, percent\_boy, “%:”

Display “girl:”, percent\_girl, “%”

End

# Get the number of boys.

boys = int(input("Enter the number of boys: "))

# Get the number of girls.

girls = int(input("Enter the number of girls: "))

# Calculate the total number of students.

total = boys + girls

##output version 1: round function

percent\_boys = round(boys/total\*100)

percent\_girls = round(girls/total\*100)

# Print the percentage of boys.

print("Boys:",str(percent\_boys) + "%")

# Print the percentage of boys.

print("Girls:",str(percent\_girls) + "%")

##output version 2: format function using %

# Calculate the percentage of boys.

percent\_boys = boys / total

# Calculate the percentage of girls.

percent\_girls = girls / total

# Print the percentage of boys.

print("Boys:", format(percent\_boys, ".0%"))

# Print the percentage of girls.

print("Girls:", format(percent\_girls, ".0%"))

**Discussion 6**

**Simple solution without considering break-even (equal distance) situations**

Set current\_location = Hotel

Mark all attractions as unvisited

route = [Hotel]

WHILE there are unvisited attractions:

nearest\_distance = very large number

nearest\_attraction = NONE

FOR each unvisited attraction:

distance = distance\_matrix[current\_location][attraction]

IF distance < nearest\_distance:

nearest\_distance = distance

nearest\_attraction = attraction

END FOR

Move to nearest\_attraction

Mark nearest\_attraction as visited

Add nearest\_attraction to route

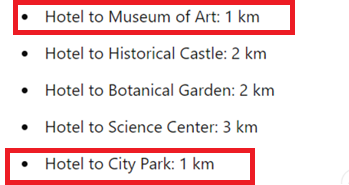
current\_location = nearest\_attraction

END WHILE

Add Hotel to route

Display route

END



You might ask the students to think further: Hotel → Museum of Art and Hotel → City Park are both 1 km away. If aiming for the best route, the students would try both options, complete the rest of the path, and compare total distances. However, since the instruction specifies a ‘good enough route’ and ‘no need to find the absolute best route,’ ask the students to devise a solution that skips checking all possible tie-break cases.

Pseudocode – Nearest Neighbor with Simple Tie-Break Rule

Set current\_location = Hotel

Mark all attractions as unvisited

route = [Hotel]

WHILE there are unvisited attractions:

nearest\_distance = very large number

nearest\_attractions\_list = empty list

FOR each unvisited attraction:

distance = distance\_matrix[current\_location][attraction]

IF distance < nearest\_distance:

nearest\_distance = distance

nearest\_attractions\_list = [attraction] // reset list with this attraction

ELSE IF distance == nearest\_distance:

Add attraction to nearest\_attractions\_list

END FOR

// Tie-breaking rule for equal distances

IF nearest\_attractions\_list has more than one attraction:

Pick one based on a simple rule:

- Choose first in list (or)

- Choose based on alphabetical order (or)

- Choose lowest attraction ID number

nearest\_attraction = chosen one

ELSE:

nearest\_attraction = nearest\_attractions\_list[0]

Move to nearest\_attraction

Mark nearest\_attraction as visited

Add nearest\_attraction to route

current\_location = nearest\_attraction

END WHILE

Add Hotel to route

Display route

END

*This way:*

* *Students* ***still get a valid route****.*
* *They* ***don’t need to try every possible tie-break branch****.*
* *The rule is* ***deterministic*** *(always produces the same choice for the same data).*