Paper coding

Try to fully understand the basic concept before moving on to the next step.

Lack of understanding basic concepts will increase your burden in learning this course, which may make you fail the course.

It may be difficult now, but for successful completion of this course we suggest you to fully understand the concept and move on to the next step.

Q1

Create prime_list that has prime numbers between 2~10 as its elements. Then, use list indexing to the first element of the list and print as shown below.

Conditions for Execution	1st element of prime_list: 2
Time	5 Minutes

Write the entire code and the expected output results in the note.

Q2

Create prime_list that has prime numbers between 1~10 as its elements. Then, use the append method to add 11. Print the results before and after addition as shown below.

Conditions for Execution	Prime numbers : [2,3,5,7] Prime numbers after addition : [2,3,5,7,11]
Time	5 Minutes

Write the entire code and the expected output results in the note.

Q3

For the list1 and list2, use the nested for loop to multiply each element of list1 and list2 and then print the result with the element multiplication result.

Conditions for Execution	Declare list1 and list2 in the first and second rows. Use the nested for loop in the third and fourth row, and use the print loop in the fifth row.
Time	5 Minutes

```
list1= = [3,5,7]
list2 = [2,3,4,5,6]
```

Output example

```
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
7 * 2 = 14
7 * 3 = 21
7 * 4 = 28
7 * 5 = 35
7 * 6 = 42
```

Write the entire code and the expected output results in the note.

| Pair programming

Pair Programming Practice

Guideline, mechanisms & contingency plan

Preparing pair programming involves establishing guidelines and mechanisms to help students pair properly and to keep them paired. For example, students should take turns “driving the mouse.” Effective preparation requires contingency plans in case one partner is absent or decides not to participate for one reason or another. In these cases, it is important to make it clear that the active student will not be punished because the pairing did not work well.

Pairing similar, not necessarily equal, abilities as partners

Pair programming can be effective when students of similar, though not necessarily equal, abilities are paired as partners. Pairing mismatched students often can lead to unbalanced participation. Teachers must emphasize that pair programming is not a “divide-and-conquer” strategy, but rather a true collaborative effort in every endeavor for the entire project. Teachers should avoid pairing very weak students with very strong students.

Motivate students by offering extra incentives

Offering extra incentives can help motivate students to pair, especially with advanced students. Some teachers have found it helpful to require students to pair for only one or two assignments.

Pair Programming Practice

Prevent collaboration cheating

The challenge for the teacher is to find ways to assess individual outcomes, while leveraging the benefits of collaboration. How do you know whether a student learned or cheated? Experts recommend revisiting course design and assessment, as well as explicitly and concretely discussing with the students on behaviors that will be interpreted as cheating. Experts encourage teachers to make assignments meaningful to students and to explain the value of what students will learn by completing them.

Collaborative learning environment

A collaborative learning environment occurs anytime an instructor requires students to work together on learning activities. Collaborative learning environments can involve both formal and informal activities and may or may not include direct assessment. For example, pairs of students work on programming assignments; small groups of students discuss possible answers to a professor’s question during lecture; and students work together outside of class to learn new concepts. Collaborative learning is distinct from projects where students “divide and conquer.” When students divide the work, each is responsible for only part of the problem solving and there are very limited opportunities for working through problems with others. In collaborative environments, students are engaged in intellectual talk with each other.

Pair programming

Q1 There is a list with the string s_list = ['abc', 'bcd', 'bcdefg', 'abba', 'cddc', 'opq’]. Implement the following function to this list.

▪

- Do not use the min function or sort method to print the shortest string from the strings of s_list. (If there are multiple shortest strings, print the string that shows the first as following.)

Output example

```
The shortest string : abc
```

Pair programming

Q2 There is a list with the string s_list = ['abc', 'bcd', 'bcdefg', 'abba', 'cddc', 'opq’]. Implement the following function to this list.

▪

- Do not use the min function or sort method to print the shortest string from the strings of s_list. (If there are multiple shortest strings, print the string that shows the first as following.)

Output example

```
The longest string : bcdefg
```

Pair programming

Q3 There is a list with the string s_list = ['abc', 'bcd', 'bcdefg', 'abba', 'cddc', 'opq']. Implement the following function to this list.

- From the pair programming problem earlier, the length of 'abc', 'bcd', 'opq' are the same as 3. Likewise, if the string lengths are the same, write a program that prints all of the three shortest strings as follows. Use the sort(key=len) function to sort the strings by length and then write a code.

Output example

```
The shortest strings : 'abc', 'bcd', 'opq'
```

Paper coding

Try to fully understand the basic concept before moving on to the next step.

Lack of understanding basic concepts will increase your burden in learning this course, which may make you fail the course.

It may be difficult now, but for successful completion of this course we suggest you to fully understand the concept and move on to the next step.

Q1

Create the capital_dic dictionary with the following string key-value items. Then, use the capital_dic to write results regarding Korea in the following dictionary items.

Program Execution Variable Declaration	key : Korea value : Seoul key : China value : Beijing key : USA value : Washington DC
Tim e	5 Minutes

Output example

```
Seoul
```

Write the entire code and the expected output results in the note.

Q2

Create the fruits_dic dictionary that has elements of the following key-value items. Then, use this dictionary to print the price of each fruit as shown below.

Conditions for Execution	The price of an apple is 5000 KRW. The price of a banana is 4000 KRW. The price of a grape is 5300 KRW. The price of a melon is 6500 KRW.
Tim e	5 Minutes

fruits_dic dictionary

key: apple value: 5000	key: banana value: 4000	key: grape value: 5300	key: melon value: 6500

Write the entire code and the expected output results in the note.

Pair programming

Pair Programming Practice

Guideline, mechanisms & contingency plan

Preparing pair programming involves establishing guidelines and mechanisms to help students pair properly and to keep them paired. For example, students should take turns “driving the mouse.” Effective preparation requires contingency plans in case one partner is absent or decides not to participate for one reason or another. In these cases, it is important to make it clear that the active student will not be punished because the pairing did not work well.

Pairing similar, not necessarily equal, abilities as partners

Pair programming can be effective when students of similar, though not necessarily equal, abilities are paired as partners. Pairing mismatched students often can lead to unbalanced participation. Teachers must emphasize that pair programming is not a “divide-and-conquer” strategy, but rather a true collaborative effort in every endeavor for the entire project. Teachers should avoid pairing very weak students with very strong students.

Motivate students by offering extra incentives

Offering extra incentives can help motivate students to pair, especially with advanced students. Some teachers have found it helpful to require students to pair for only one or two assignments.

Pair Programming Practice

Prevent collaboration cheating

The challenge for the teacher is to find ways to assess individual outcomes, while leveraging the benefits of collaboration. How do you know whether a student learned or cheated? Experts recommend revisiting course design and assessment, as well as explicitly and concretely discussing with the students on behaviors that will be interpreted as cheating. Experts encourage teachers to make assignments meaningful to students and to explain the value of what students will learn by completing them.

Collaborative learning environment

A collaborative learning environment occurs anytime an instructor requires students to work together on learning activities. Collaborative learning environments can involve both formal and informal activities and may or may not include direct assessment. For example, pairs of students work on programming assignments; small groups of students discuss possible answers to a professor’s question during lecture; and students work together outside of class to learn new concepts. Collaborative learning is distinct from projects where students “divide and conquer.” When students divide the work, each is responsible for only part of the problem solving and there are very limited opportunities for working through problems with others. In collaborative environments, students are engaged in intellectual talk with each other.

Pair programming

Q1 Create the fruits_dic dictionary consists of key-value pairs including ('apple', 6000), ('melon', 3000), ('banana', 5000), ('orange', 4000). Then, print all of the key in the fruits_dic as list type and examine if the ‘apple’ and ‘mango’ keys are found in the fruits_dic, and print as follows.

▪

Output example

```
dict_keys(['apple', 'melon', 'banana', 'orange'])
apple is in fruits_dic.
mango is not in fruits_dic.
```

Paper coding

Try to fully understand the basic concept before moving on to the next step.

Lack of understanding basic concepts will increase your burden in learning this course, which may make you fail the course.

It may be difficult now, but for successful completion of this course we suggest you to fully understand the concept and move on to the next step.

Paper coding

Q1 Predict the execution result of the following code and provide handwritten result.

Conditions for Execution	Predict the result of the following program and provide handwritten coding results.
Time	5 Min

Output example

```python
t1 = 'a', 'b', 'c'
t2 = ('a', 'b', 'c')
t3 = ('d', 'e')

print(t1 == t2)

print(t1 > t3)

print(t1 < t3)

print(t2 + t3)

print([ t2 + t3 ])

print(t1)
```

Write the entire code and the expected output results in the note.

Q2.

The following tuple records daily sales of a store for 10 days. Write a code to print how many days had reduced sales compared to previous day. (Hint: compare the values by iterating the elements with the iteration statement.)

Conditions for Execution	Daily sales record: (100, 121, 120, 130, 140, 120, 122, 123, 190, 125) In the past 10 days, 3 days had reduced sales compared to the previous day.
Time	10 Min

Write the entire code and the expected output results in the note.

Pair programming

Pair Programming Practice

Guideline, mechanisms & contingency plan

Preparing pair programming involves establishing guidelines and mechanisms to help students pair properly and to keep them paired. For example, students should take turns "driving the mouse." Effective preparation requires contingency plans in case one partner is absent or decides not to participate for one reason or another. In these cases, it is important to make it clear that the active student will not be punished because the pairing did not work well.

Pairing similar, not necessarily equal, abilities as partners

Pair programming can be effective when students of similar, though not necessarily equal, abilities are paired as partners. Pairing mismatched students often can lead to unbalanced participation. Teachers must emphasize that pair programming is not a "divide-and-conquer" strategy, but rather a true collaborative effort in every endeavor for the entire project. Teachers should avoid pairing very weak students with very strong students.

Motivate students by offering extra incentives

Offering extra incentives can help motivate students to pair, especially with advanced students. Some teachers have found it helpful to require students to pair for only one or two assignments.

Pair Programming Practice

Prevent collaboration cheating

The challenge for the teacher is to find ways to assess individual outcomes, while leveraging the benefits of collaboration. How do you know whether a student learned or cheated? Experts recommend revisiting course design and assessment, as well as explicitly and concretely discussing with the students on behaviors that will be interpreted as cheating. Experts encourage teachers to make assignments meaningful to students and to explain the value of what students will learn by completing them.

Collaborative learning environment

A collaborative learning environment occurs anytime an instructor requires students to work together on learning activities. Collaborative learning environments can involve both formal and informal activities and may or may not include direct assessment. For example, pairs of students work on programming assignments; small groups of students discuss possible answers to a professor's question during lecture; and students work together outside of class to learn new concepts. Collaborative learning is distinct from projects where students "divide and conquer." When students divide the work, each is responsible for only part of the problem solving and there are very limited opportunities for working through problems with others. In collaborative environments, students are engaged in intellectual talk with each other.

Pair programming

Q1. Return the element with the maximum number of occurrences. When there are more than two frequent elements, print the highest number.

Output example

```
Given tuples: (1, 2, 5, 4, 3, 2, 1, 4, 7, 8, 9, 9, 3, 7, 3, 9)
The most frequent element: 9
```

Pair programming

Q2. In the output example below, there are the tuples containing elements, as well as empty tuples, empty strings and empty lists that have no elements.
Write a code to remove these empty tuples, empty strings and empty lists from the given list below.
(However, do not remove (,) tuple because it is considered as having one empty tuple.)

Output example

```
Given tuples: [(), (1,), [], 'abc', (), (), (1,), ('a',), ('a', 'b'), ((),), '']
The most frequent element: [(1,), 'abc', (1,), ('a',), ('a', 'b'), ((),)]
```

Paper coding

Try to fully understand the basic concept before moving on to the next step.

Lack of understanding basic concepts will increase your burden in learning this course, which may make you fail the course.

It may be difficult now, but for successful completion of this course we suggest you to fully understand the concept and move on to the next step.

Q1

Put the two-dimensional arrays [[10, 20], [30, 40], [50, 60] into the variable list_array and output 30. Do the correct indexing.

Conditions for Execution	`30`
Time	5min

Write the entire code and the expected output results in the note.

Q2

Create a 2D list of 4 x 4 size with values ranging from 1 to 16 and print all the elements using the for loop.

Conditions For Execution	`1 2 3 4`
	`5 6 7 8`
	`9 10 11 12`
	`13 14 15 16`

Time	5min

Write the entire code and the expected output results in the note.

Pair programming

Pair Programming Practice

Guideline, mechanisms & contingency plan

Preparing pair programming involves establishing guidelines and mechanisms to help students pair properly and to keep them paired. For example, students should take turns “driving the mouse.” Effective preparation requires contingency plans in case one partner is absent or decides not to participate for one reason or another. In these cases, it is important to make it clear that the active student will not be punished because the pairing did not work well.

Pairing similar, not necessarily equal, abilities as partners

Pair programming can be effective when students of similar, though not necessarily equal, abilities are paired as partners. Pairing mismatched students often can lead to unbalanced participation. Teachers must emphasize that pair programming is not a “divide-and-conquer” strategy, but rather a true collaborative effort in every endeavor for the entire project. Teachers should avoid pairing very weak students with very strong students.

Motivate students by offering extra incentives

Offering extra incentives can help motivate students to pair, especially with advanced students. Some teachers have found it helpful to require students to pair for only one or two assignments.

Pair Programming Practice

Prevent collaboration cheating

The challenge for the teacher is to find ways to assess individual outcomes, while leveraging the benefits of collaboration. How do you know whether a student learned or cheated? Experts recommend revisiting course design and assessment, as well as explicitly and concretely discussing with the students on behaviors that will be interpreted as cheating. Experts encourage teachers to make assignments meaningful to students and to explain the value of what students will learn by completing them.

Collaborative learning environment

A collaborative learning environment occurs anytime an instructor requires students to work together on learning activities. Collaborative learning environments can involve both formal and informal activities and may or may not include direct assessment. For example, pairs of students work on programming assignments; small groups of students discuss possible answers to a professor's question during lecture; and students work together outside of class to learn new concepts. Collaborative learning is distinct from projects where students "divide and conquer." When students divide the work, each is responsible for only part of the problem solving and there are very limited opportunities for working through problems with others. In collaborative environments, students are engaged in intellectual talk with each other.

Pair programming

Q1 Write a program that generates a multidimensional array of n×n size, based on the number of inputs, by receiving two or more n as inputs from users. In this case, the content of the arrangement should be displayed so that the values of 0 and 1 intersect in a checkered pattern.

Output example

```
Enter n: 5
1 0 1 0 1
0 1 0 1 0
1 0 1 0 1
0 1 0 1 0
1 0 1 0 1
```

Paper coding

Try to fully understand the basic concept before moving on to the next step.

Lack of understanding basic concepts will increase your burden in learning this course, which may make you fail the course.

It may be difficult now, but for successful completion of this course we suggest you to fully understand the concept and move on to the next step.

Paper coding

Q1 Let's create a dictionary named person_dic with the following contact information on your phone.
Print this information using the for loop to show the output results below.

Conditions for Execution	`'Last Name': 'Doe', 'First Name': 'David', 'Company': 'Samsung'`
Time	5min

Last Name	
First Name	
Company	

```
Last Name   : Doe
First Name   : David
Company   : Samsung
```

Write the entire code and the expected output results in the note.

Q2

Let's write a program that performs inventory management at a convenience store. To this end, inventory of items sold at convenience stores is stored in the items dictionary as shown in the example below. Write a program that receives the name of the item from users and returns the inventory of the item. Suppose that it is a very small convenience store and the items treated are as following.

Example program execution results.	Enter name of the item: Milk 1
Time	5min

Items

```
items = {"Coffee": 7, "Pen":3, "Paper cup": 2, "Milk": 1, "Coke": 4, "Book":5}
```

Write the entire code and the expected output results in the note.

Pair programming

Pair Programming Practice

Guideline, mechanisms & contingency plan

Preparing pair programming involves establishing guidelines and mechanisms to help students pair properly and to keep them paired. For example, students should take turns "driving the mouse." Effective preparation requires contingency plans in case one partner is absent or decides not to participate for one reason or another. In these cases, it is important to make it clear that the active student will not be punished because the pairing did not work well.

Pairing similar, not necessarily equal, abilities as partners

Pair programming can be effective when students of similar, though not necessarily equal, abilities are paired as partners. Pairing mismatched students often can lead to unbalanced participation. Teachers must emphasize that pair programming is not a "divide-and-conquer" strategy, but rather a true collaborative effort in every endeavor for the entire project. Teachers should avoid pairing very weak students with very strong students.

Motivate students by offering extra incentives

Offering extra incentives can help motivate students to pair, especially with advanced students. Some teachers have found it helpful to require students to pair for only one or two assignments.

Pair Programming Practice

Prevent collaboration cheating

The challenge for the teacher is to find ways to assess individual outcomes, while leveraging the benefits of collaboration. How do you know whether a student learned or cheated? Experts recommend revisiting course design and assessment, as well as explicitly and concretely discussing with the students on behaviors that will be interpreted as cheating. Experts encourage teachers to make assignments meaningful to students and to explain the value of what students will learn by completing them.

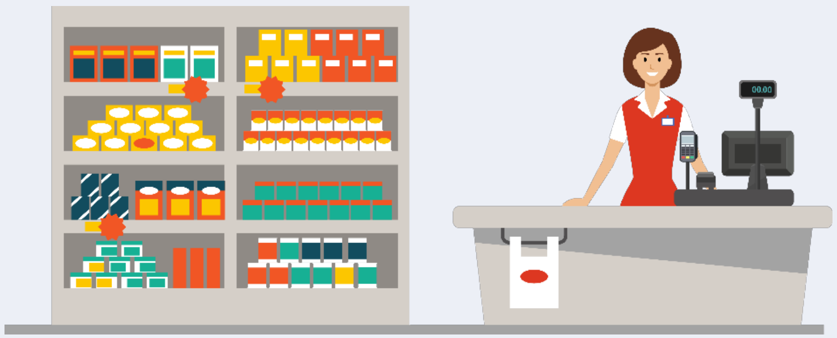
Collaborative learning environment

A collaborative learning environment occurs anytime an instructor requires students to work together on learning activities. Collaborative learning environments can involve both formal and informal activities and may or may not include direct assessment. For example, pairs of students work on programming assignments; small groups of students discuss possible answers to a professor’s question during lecture; and students work together outside of class to learn new concepts. Collaborative learning is distinct from projects where students “divide and conquer.” When students divide the work, each is responsible for only part of the problem solving and there are very limited opportunities for working through problems with others. In collaborative environments, students are engaged in intellectual talk with each other.

Pair programming

Q1 Let's upgrade the program to manage the inventory of convenience stores that we solved in paper coding. In other words, add code to increase or decrease inventory. Also, make simple menus such as inventory inquiry, warehousing, and shipment.

```python
items = {"Coffee": 7, "Pen":3, "Paper cup": 2, "Milk": 1, "Coke": 4, "Book":5}
```



Output example

```
Select menu 1)check stock 2)warehousing 3) release 4) exit :1
[check stock] Enter item: milk
Stock: 1
Select menu 1)check stock 2)warehousing 3) release 4) exit :3
[Release] Enter item and quantity: coke 1
Select menu 1)check stock 2)warehousing 3) release 4) exit :1
[check stock] Enter item: coke
Stock: 3
Select menu 1)check stock 2)warehousing 3) release 4) exit :4
Program exited.
```

Paper coding

Try to fully understand the basic concept before moving on to the next step.

Lack of understanding basic concepts will increase your burden in learning this course, which may make you fail the course.

It may be difficult now, but for successful completion of this course we suggest you to fully understand the concept and move on to the next step.

Q1

A tuple called study_tup, which has three pairs of elements: student ID number, name, and phone number, exists as shown below. Modify the student_tup below to create and print a dictionary of the pair {student ID number : [name, phone number]}.

Condition for Execution	student_tup = (('211101', ‘David Doe', '010-1234-4500'), ('211102', ' John Smith', '010-2230-6540'), ('211103', ' Jane Carter', '010-3232-7788'))
Time	7 min

Output example

```
{'211101' : ['David Doe', '010-1234-4500'] }
{'211102' : ['John Smith', '010-2230-6540'] }
{'211103' : ['Jane Carter', '010-3232-7788'] }
```

Write the entire code and the expected output results in the note.

Q2 Write a bachelor's information program using the student_tup above to receive the student's student ID number as input and print the student's name and phone number.

Example program execution results.	Enter student ID number : 211101 Name : David Doe Phone number : 010-1234-4500
Time	5min

Write the entire code and the expected output results in the note.

Pair programming

Pair Programming Practice

Guideline, mechanisms & contingency plan

Preparing pair programming involves establishing guidelines and mechanisms to help students pair properly and to keep them paired. For example, students should take turns “driving the mouse.” Effective preparation requires contingency plans in case one partner is absent or decides not to participate for one reason or another. In these cases, it is important to make it clear that the active student will not be punished because the pairing did not work well.

Pairing similar, not necessarily equal, abilities as partners

Pair programming can be effective when students of similar, though not necessarily equal, abilities are paired as partners. Pairing mismatched students often can lead to unbalanced participation. Teachers must emphasize that pair programming is not a “divide-and-conquer” strategy, but rather a true collaborative effort in every endeavor for the entire project. Teachers should avoid pairing very weak students with very strong students.

Motivate students by offering extra incentives

Offering extra incentives can help motivate students to pair, especially with advanced students. Some teachers have found it helpful to require students to pair for only one or two assignments.

Pair Programming Practice

Prevent collaboration cheating

The challenge for the teacher is to find ways to assess individual outcomes, while leveraging the benefits of collaboration. How do you know whether a student learned or cheated? Experts recommend revisiting course design and assessment, as well as explicitly and concretely discussing with the students on behaviors that will be interpreted as cheating. Experts encourage teachers to make assignments meaningful to students and to explain the value of what students will learn by completing them.

Collaborative learning environment

A collaborative learning environment occurs anytime an instructor requires students to work together on learning activities. Collaborative learning environments can involve both formal and informal activities and may or may not include direct assessment. For example, pairs of students work on programming assignments; small groups of students discuss possible answers to a professor’s question during lecture; and students work together outside of class to learn new concepts. Collaborative learning is distinct from projects where students “divide and conquer.” When students divide the work, each is responsible for only part of the problem solving and there are very limited opportunities for working through problems with others. In collaborative environments, students are engaged in intellectual talk with each other.

Pair programming

Q1

The student_tuple list with tuples as elements is as shown below. Tuple, which is the element of this tuple consists of a (student ID number, name, phone number). Using this, make a dictionary for (student ID number: name) and print it out. When inquiring by student ID number, make sure that the student ID number, name, and phone number are printed as shown below.

▪

- student_tuple = [('211101', 'David Doe', '010-123-1111'), ('211102', 'John Smith', '010-123-2222'),
  ('211103', 'Jane Carter', '010-123-3333')]

Example Output

```
211101 : David Doe
211102 : John Smith
211103 : Jane Carter
Enter student ID number : 211103
211103 student is Jane Carter and phone number is 010-123-333.
```

Paper coding

Try to fully understand the basic concept before moving on to the next step.

Lack of understanding basic concepts will increase your burden in learning this course, which may make you fail the course.

It may be difficult now, but for successful completion of this course we suggest you to fully understand the concept and move on to the next step.

Q1 Use the set function to generate and print the set s1 from the next list lst.

Program variable conditions	lst = ['apple', 'mango', 'banana'] # A list of 3 fruit information
	s1 = {'apple', 'mango', 'banana'} # Set s1 generated from lst

Time	7min

Output Example

```
s1 = {'banana', 'apple', 'mango'}
```

Write the entire code and the expected output results in the note.

Paper coding

Q2 Write down the computational results for the following two sets. Find the results from 1) to 7).

▪

Program operation conditions	s1 = {10, 20, 30, 40} s2 = {30, 40, 50, 60, 70} 1) s1 | s2 2) s1 & s2 3) s1 – s2 4) s1 ^ s2 5) s1.issubset(s2) 6) s1.issuperset(s2) 7) s1.isdisjoint(s2)
Tim e	5min

Write the entire code and the expected output results in the note.

Pair programming

Pair Programming Practice

Guideline, mechanisms & contingency plan

Preparing pair programming involves establishing guidelines and mechanisms to help students pair properly and to keep them paired. For example, students should take turns "driving the mouse." Effective preparation requires contingency plans in case one partner is absent or decides not to participate for one reason or another. In these cases, it is important to make it clear that the active student will not be punished because the pairing did not work well.

Pairing similar, not necessarily equal, abilities as partners

Pair programming can be effective when students of similar, though not necessarily equal, abilities are paired as partners. Pairing mismatched students often can lead to unbalanced participation. Teachers must emphasize that pair programming is not a "divide-and-conquer" strategy, but rather a true collaborative effort in every endeavor for the entire project. Teachers should avoid pairing very weak students with very strong students.

Motivate students by offering extra incentives

Offering extra incentives can help motivate students to pair, especially with advanced students. Some teachers have found it helpful to require students to pair for only one or two assignments.

Pair Programming Practice

Prevent collaboration cheating

The challenge for the teacher is to find ways to assess individual outcomes, while leveraging the benefits of collaboration. How do you know whether a student learned or cheated? Experts recommend revisiting course design and assessment, as well as explicitly and concretely discussing with the students on behaviors that will be interpreted as cheating. Experts encourage teachers to make assignments meaningful to students and to explain the value of what students will learn by completing them.

Collaborative learning environment

A collaborative learning environment occurs anytime an instructor requires students to work together on learning activities. Collaborative learning environments can involve both formal and informal activities and may or may not include direct assessment. For example, pairs of students work on programming assignments; small groups of students discuss possible answers to a professor’s question during lecture; and students work together outside of class to learn new concepts. Collaborative learning is distinct from projects where students “divide and conquer.” When students divide the work, each is responsible for only part of the problem solving and there are very limited opportunities for working through problems with others. In collaborative environments, students are engaged in intellectual talk with each other.

Pair programming

Q1

There is a list mylist with tuple(m, n) as elements as shown below. If there is tuple with (a,b) value and a, b values entered from the user, print 'There is an element (a, b) in xth'. If there is no (a, b) but (b, a) is there, print 'There is no (a, b) but (b, a) there is at yth. If there is no (a, b) nor (b, a), print 'there is no such element'.

▪

- mylist = [(1, 2), (4, 5), (4, 2), (3, 1), (9, 4)]

Output Example

```
Enter two integers: 1 2
There is (1,2) at the first.
```

```
Enter two integers: 5 4
There is no (5,4) but there is (4,5) at the second.
```

```
Enter two integers: 3 9
There is no (3,9) nor (9,3)
```