Project 1: Counting Methods

For this assignment, you can just upload this document with your answers provided after each question below.

* If you worked with a partner, include their name
* Please print and sign the honor pledge somewhere on the document

**For this project, you should not use built-in functions or itertools.**

1. A student is registering for classes for next semester. The courses she can enroll in are calculus (CA), physics (PH), organic chemistry(ORGO), history(HIST), literature(LIT), French(FR), and astronomy(ASTR). She has room for four classes in her schedule, at the following times: 9am, 10am, 11am, 12pm. She wants to count all the possible versions of her schedule. Additionally, the student’s mother is helping her buy textbooks for her classes. Each course listed above has a specific textbook that is required with it. Therefore, a book order for the student will consist of four textbooks. Book orders are automatically listed in alphabetical order by the class subject (e.g., one book order could be astronomy, French, organic chemistry, physics). The student’s mother wants to count all the possible book orders that she could be making.

* 1. Make the beginnings of a carefully ordered list of all possible book orders. (Your list should include enough outcomes so that someone else can see and continue the pattern.)

35 combinations 7!/(4!(7-4)!)

Astr, calc, chem, french

Astr, calc, chem, hist

Astr, calc, chem, lit

Astr, calc, chem, phys

Astr, calc, fren, chem

Astr, calc, fren, hist

* 1. Now make the beginnings of a carefully ordered list of all the possible versions of the student’s *schedule,* assuming she can’t register one class for multiple slots. (Your list should include enough outcomes so that someone else can see and continue the pattern.)

Should have 840 but only doing a couple here

CA, PH, ORGO, HIST

CA, PH, HIST, ORGO

CA, ORGO, PH, HIST

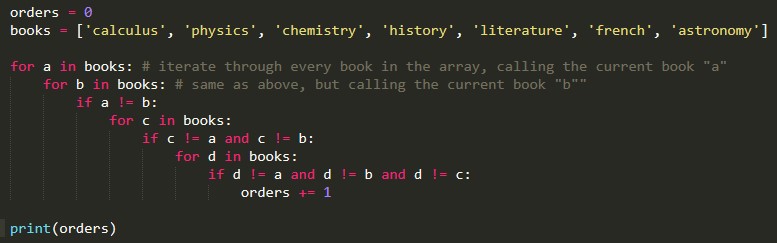
CA, ORGO, HIST, PH

CA, HIST, ORGO, PH

CA, HIST, PH, ORGO

Strategy is to pick one set of 4 from a and shuffle them exhaustively to extract all orders. The 6 above only go up to shuffling the last 3 elements of (CA, PH, ORGO, HIST) but its doable to shuffle all.

* 1. Here is a program written in Python. Does this count the possible book orders, schedules, or neither? Explain by referencing specific pieces of the provided code.



This code counts the number of the total possible schedules. This can be detected by looking at how the if functions work. It only checks that none of the books are identical and iterates the order then. This assumes that each unique permutation of classes is important. Which indicates that the code is counting the number of sequences of classes. We would probably use < or <= if we were trying to calculate the number of book orders as in that case, the sequence of books in the book order is irrelevant.

* 1. Write a program using for loops that *lists* AND *counts* all the possible book orders. **Paste your code below. What are the first 20 outcomes that your program lists? What is the total number of outcomes?** Hint: consider using < or <= instead of !=. (The logic and code for this problem and in problems 2 & 3 should closely resemble the logic and code provided in problem 1c.)

Code:

orders = 0

books = ['astronomy', 'calculus', 'chemistry', 'french', 'history', 'literature', 'physics']

for a in books:

for b in books:

if a < b:

for c in books:

if c > a and c > b:

for d in books:

if d > a and d > b and d > c:

orders += 1

print(orders, a, b, c, d)

print(orders)

The first 20 outcomes are here:

1 astronomy calculus chemistry french

2 astronomy calculus chemistry history

3 astronomy calculus chemistry literature

4 astronomy calculus chemistry physics

5 astronomy calculus french history

6 astronomy calculus french literature

7 astronomy calculus french physics

8 astronomy calculus history literature

9 astronomy calculus history physics

10 astronomy calculus literature physics

11 astronomy chemistry french history

12 astronomy chemistry french literature

13 astronomy chemistry french physics

14 astronomy chemistry history literature

15 astronomy chemistry history physics

16 astronomy chemistry literature physics

17 astronomy french history literature

18 astronomy french history physics

19 astronomy french literature physics

20 astronomy history literature physics

The total number of outcomes are 35.

1. An 8-digit password is required to have three 0’s and five 1’s. You will determine how many unique passwords are possible.

First consider how you might notate possible outcomes in the process of listing them. For example, any of the following can represent the same outcome:

01011101

137 (the digits are the locations of the 0’s)

24568 (the digits are the locations of the 1’s)

Write a program that lists and counts the unique passwords. **Provide your code, all outcomes, and the count**.

The code is here:

orders = 0

books = ['0', '1', '2', '3', '4', '5', '6', '7']

for a in books:

for b in books:

if a < b:

for c in books:

if c > a and c > b:

orders += 1

print(a, b, c)

print(orders)

We only care about the locations of the 0’s as the ones just fall in. So we only need to find the unique combinations of 3 numbers of the 8 locations. We use the array 0-7 to denote the positions of the 0s. Here are all 56 outcomes:

0 1 2

0 1 3

0 1 4

0 1 5

0 1 6

0 1 7

0 2 3

0 2 4

0 2 5

0 2 6

0 2 7

0 3 4

0 3 5

0 3 6

0 3 7

0 4 5

0 4 6

0 4 7

0 5 6

0 5 7

0 6 7

1 2 3

1 2 4

1 2 5

1 2 6

1 2 7

1 3 4

1 3 5

1 3 6

1 3 7

1 4 5

1 4 6

1 4 7

1 5 6

1 5 7

1 6 7

2 3 4

2 3 5

2 3 6

2 3 7

2 4 5

2 4 6

2 4 7

2 5 6

2 5 7

2 6 7

3 4 5

3 4 6

3 4 7

3 5 6

3 5 7

3 6 7

4 5 6

4 5 7

4 6 7

5 6 7

1. An 8-digit password is required to have exactly three 0’s. The other 5 digits can be any number 1-7, but numbers 1-7 may not be repeated.

(a) Consider different options for notating possible outcomes. (Hint: it may be easier to think of a way to notate the *location* of the 0’s, rather than just writing the outcomes as a list of 8 digits) **Then make a list of at least 20 possible passwords.** (List the outcomes in an ordered, intentional way so as not to miss any if you kept going.)

We are essentially moving forward on the previous problem and all we have to do is replace the ones with real numbers. Here I will add just 5 numbers after the locations of 0’s as such:

1 2 3-1 2 3 4 5= 10002345

0 1 2-12345

0 1 3-12345

0 1 4-12345

0 1 5-12345

0 1 6-12345

0 1 7-12345

0 2 3-12345

0 2 4-12345

0 2 5-12345

0 2 6-12345

0 2 7-12345

0 3 4-12345

0 3 5-12345

0 3 6-12345

0 3 7-12345

0 4 5-12345

0 4 6-12345

0 4 7-12345

0 5 6-12345

0 5 7-12345

(b) Write a program that lists all the possible outcomes. **Provide your code and the first 100 outcomes**.

The code is here and gives output in the format described above with the first three digits identifying the position of the 0s and the remaining digits identifying actual digits. The total number of outcomes here is 141120. This was verified by this function mathematically:

Where the first term accounts for the locations of 0’s (8 locations, pick 3) and the latter term accounts for the other numbers in the five spots. This value was verified programmatically as well.

orders = 0

books = ['0', '1', '2', '3', '4', '5', '6', '7']

books1 = ['1', '2', '3', '4', '5', '6', '7']

for a in books:

for b in books:

if a < b:

for c in books:

if c > a and c > b:

for a1 in books1:

for b1 in books1:

if a1 != b1:

for c1 in books1:

if c1 != a1 and c1 != b1:

for d1 in books1:

if d1 != c1 and d1 != b1 and d1 != a1:

for e1 in books1:

if e1 != a1 and e1 != b1 and e1 != c1 and e1 != d1:

orders += 1

print(orders, a, b, c, a1, b1, c1, d1, e1)

print(orders)

The first 100 outcomes:

1 0 1 2 1 2 3 4 5

2 0 1 2 1 2 3 4 6

3 0 1 2 1 2 3 4 7

4 0 1 2 1 2 3 5 4

5 0 1 2 1 2 3 5 6

6 0 1 2 1 2 3 5 7

7 0 1 2 1 2 3 6 4

8 0 1 2 1 2 3 6 5

9 0 1 2 1 2 3 6 7

10 0 1 2 1 2 3 7 4

11 0 1 2 1 2 3 7 5

12 0 1 2 1 2 3 7 6

13 0 1 2 1 2 4 3 5

14 0 1 2 1 2 4 3 6

15 0 1 2 1 2 4 3 7

16 0 1 2 1 2 4 5 3

17 0 1 2 1 2 4 5 6

18 0 1 2 1 2 4 5 7

19 0 1 2 1 2 4 6 3

20 0 1 2 1 2 4 6 5

21 0 1 2 1 2 4 6 7

22 0 1 2 1 2 4 7 3

23 0 1 2 1 2 4 7 5

24 0 1 2 1 2 4 7 6

25 0 1 2 1 2 5 3 4

26 0 1 2 1 2 5 3 6

27 0 1 2 1 2 5 3 7

28 0 1 2 1 2 5 4 3

29 0 1 2 1 2 5 4 6

30 0 1 2 1 2 5 4 7

31 0 1 2 1 2 5 6 3

32 0 1 2 1 2 5 6 4

33 0 1 2 1 2 5 6 7

34 0 1 2 1 2 5 7 3

35 0 1 2 1 2 5 7 4

36 0 1 2 1 2 5 7 6

37 0 1 2 1 2 6 3 4

38 0 1 2 1 2 6 3 5

39 0 1 2 1 2 6 3 7

40 0 1 2 1 2 6 4 3

41 0 1 2 1 2 6 4 5

42 0 1 2 1 2 6 4 7

43 0 1 2 1 2 6 5 3

44 0 1 2 1 2 6 5 4

45 0 1 2 1 2 6 5 7

46 0 1 2 1 2 6 7 3

47 0 1 2 1 2 6 7 4

48 0 1 2 1 2 6 7 5

49 0 1 2 1 2 7 3 4

50 0 1 2 1 2 7 3 5

51 0 1 2 1 2 7 3 6

52 0 1 2 1 2 7 4 3

53 0 1 2 1 2 7 4 5

54 0 1 2 1 2 7 4 6

55 0 1 2 1 2 7 5 3

56 0 1 2 1 2 7 5 4

57 0 1 2 1 2 7 5 6

58 0 1 2 1 2 7 6 3

59 0 1 2 1 2 7 6 4

60 0 1 2 1 2 7 6 5

61 0 1 2 1 3 2 4 5

62 0 1 2 1 3 2 4 6

63 0 1 2 1 3 2 4 7

64 0 1 2 1 3 2 5 4

65 0 1 2 1 3 2 5 6

66 0 1 2 1 3 2 5 7

67 0 1 2 1 3 2 6 4

68 0 1 2 1 3 2 6 5

69 0 1 2 1 3 2 6 7

70 0 1 2 1 3 2 7 4

71 0 1 2 1 3 2 7 5

72 0 1 2 1 3 2 7 6

73 0 1 2 1 3 4 2 5

74 0 1 2 1 3 4 2 6

75 0 1 2 1 3 4 2 7

76 0 1 2 1 3 4 5 2

77 0 1 2 1 3 4 5 6

78 0 1 2 1 3 4 5 7

79 0 1 2 1 3 4 6 2

80 0 1 2 1 3 4 6 5

81 0 1 2 1 3 4 6 7

82 0 1 2 1 3 4 7 2

83 0 1 2 1 3 4 7 5

84 0 1 2 1 3 4 7 6

85 0 1 2 1 3 5 2 4

86 0 1 2 1 3 5 2 6

87 0 1 2 1 3 5 2 7

88 0 1 2 1 3 5 4 2

89 0 1 2 1 3 5 4 6

90 0 1 2 1 3 5 4 7

91 0 1 2 1 3 5 6 2

92 0 1 2 1 3 5 6 4

93 0 1 2 1 3 5 6 7

94 0 1 2 1 3 5 7 2

95 0 1 2 1 3 5 7 4

96 0 1 2 1 3 5 7 6

97 0 1 2 1 3 6 2 4

98 0 1 2 1 3 6 2 5

99 0 1 2 1 3 6 2 7

100 0 1 2 1 3 6 4 2

(c) Suppose you have a password with *m* digits required to have exactly *n* 0’s. The other (m-n) digits can be any number 1-9 but numbers 1-9 may not be repeated. **How many unique passwords exist?** (Your answer should be in terms of m and n.)

So two parts. The first is the locations of the 0s. This is a m C n problem where only the composition matters and order doesn’t. This gives us the equation

And for the next part we have a permutation problem where 9 P (m-n). This gives us the equation:

Combining the two (multiplying them) as we build on the positions of 0’s by inputting the order of real digits we get:

4. A 5-letter password is to be constructed from the following letters: A, B, C, D, and E and the password must contain exactly 3 of the letters. (Obviously at least one of them will need to be used more than once to create a 5-letter password.)

1. Suppose A,B, and C are the letters selected and we use A twice, B twice, and C once. How many different passwords exist in this scenario?

All we get to do is pick the positions of the As, and Bs. So this becomes something along the lines of:

Where the first term accounts for the position selections of the A’s, second term for the B’s, and the final term for the C’s

1. Suppose A,B, and C are the letters selected and there are no other restrictions. How many different passwords exist in this scenario?

This problem is fairly simple. All we need to do is find unique 5 letter sequences that only use 3 unique characters. We can think of this as one of those bike looks with wheels that have a single line up catch phrase like shown below:

In our case, there are 5 rings and each of the rings only has 3 characters or notches. This enables us to think about the problem as a permutation/possibilities problem (I am unsure about the exact phrasing). But all we need to do to get the total number of possible passwords is multiply each of the potential possibilities of number for each position by one another. As such: 3\*3\*3\*3\*3 = 35. This gives us a total number of combinations of 243.

1. Now suppose we don’t put any restrictions on the letters chosen. (It could be A,B, and C, or it could be A, C, and D, or it could be B, D, and E, etc.) How many different passwords exist in this scenario? **5 OPTIONS PICKING 3 LETTERS**

The answer is the same as before if we have the same number of possible characters(3) to select 3 characters from. But that is not the case, This time, we have the entire set of 5 capital letters to select from (5). This is a combination problem where out of 5, we simply pick 3. So all we have to do is multiply the previous answer we found in 4b by 5 C 3:

This yields us the answer of 2430. We choose the combination formula here over the permutation formula because we only care about the letters, we select from the set of 5 letters as a set not as a sequence. The latter part of the expression above accounts for the sequential arrangement of the letters.

*You may solve problem 4 however you wish (with or without a program). If you write a program, you should clearly explain what each section of your code accomplishes. If you don’t use a program, you should show all your work and justify each computation.*