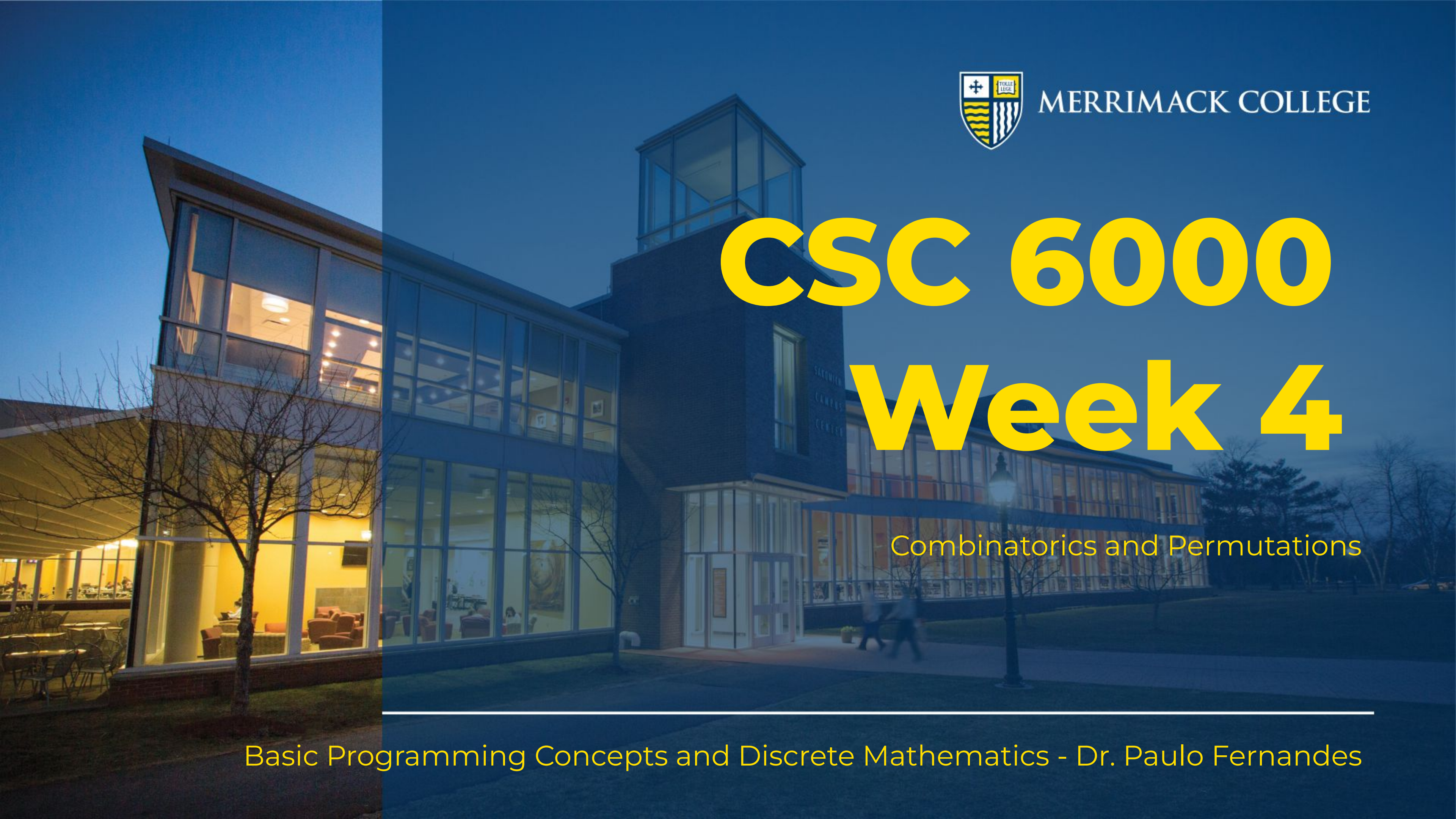


MERRIMACK COLLEGE

CSC 6000 Week 4

Combinatorics and Permutations

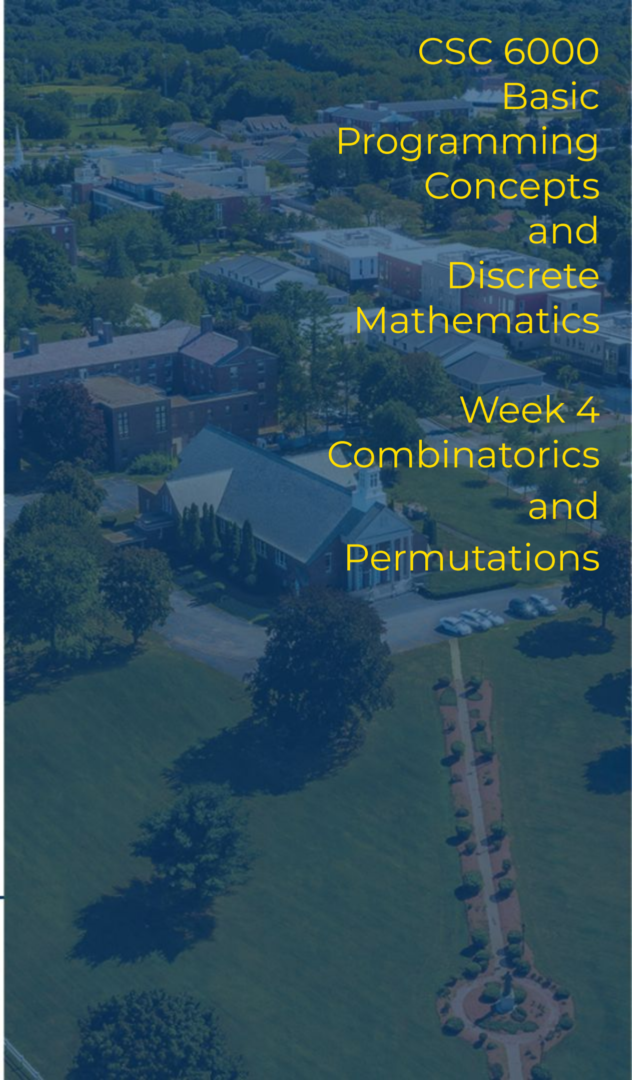
Basic Programming Concepts and Discrete Mathematics - Dr. Paulo Fernandes

Presentation Agenda

Week 4

- Combinatorics
    a. Permutations, Combinations, and Partitions
        i. A few Python tricks to use
- Permutations
    a. Permutations
    b. Permutations of Multisets
- Partial Permutations - a.k.a. Arrangements
    a. Arrangements
    b. Arrangements of Multisets
- This Week's tasks

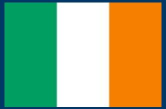
MERRIMACK COLLEGE

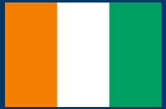

CSC 6000 Basic Programming Concepts and Discrete Mathematics

Week 4 Combinatorics and Permutations

Combinatorics

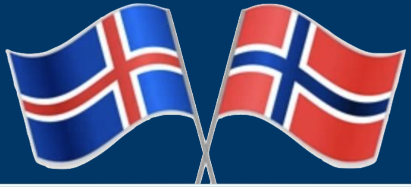
What is the difference between the flags of Ireland and Ivory Coast?





Just a permutation!

… and what about Norway and Iceland flags?



Counting is natural - we count the days, the weeks (this is the fourth by the way), months, years, choices, possibilities, etc.

Combinatorics is an analysis tool to assess possible outcomes, but also to define structures.

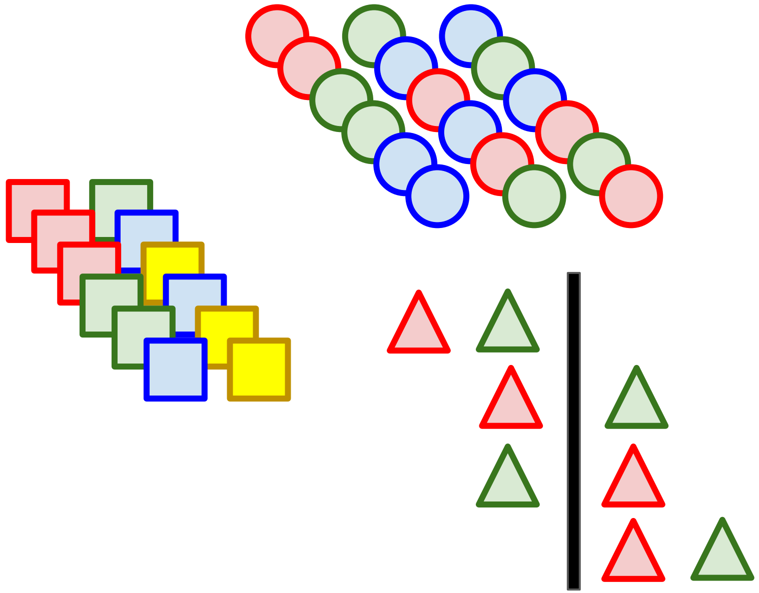
The main kinds of combinatorics are Enumerative and are applied to compute:

- permutations,
- combinations, and
- partitions.

Enumerative Combinatorics

The usual problems solved by enumerative combinatorics are:

- Permutations
  - How many ways to sort the elements of a set
- Combinations
  - How many subsets of a given size from a set
- Partitions
  - How many ways to split the elements of a set



MERRIMACK COLLEGE

Wikipedia: Combinatorics.

Enumerative Combinatorics

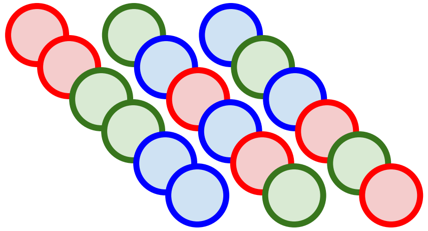
Permutations

- How many ways to sort the elements of a set

We will see today how to compute these in a few variations.

From a practical point of view, these are the most common basic problems in combinatorics. Permutations can be applied to answer questions like:

- Three athletes, **Joe**, **Jack**, and **Jeff** are competing in a final round. How many outcomes are possible for the podium?



MERRIMACK COLLEGE

Wikipedia: Permutation.

Enumerative Combinatorics

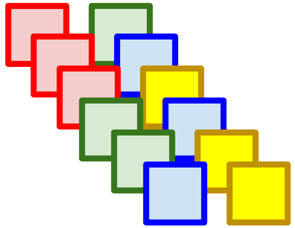
Combinations

- How many subsets of a given size from a set

We will see next week how to compute these in a few variations.

From a practical point of view, these are the oldest problems in combinatorics. Combinations can be applied to answer questions like:

- Four state teams, **MA**, **NY**, **FL**, and **CA** are competing in a pool. How many matches are needed for all teams to play against each other?



MERRIMACK COLLEGE

Wikipedia: Combination.

Enumerative Combinatorics

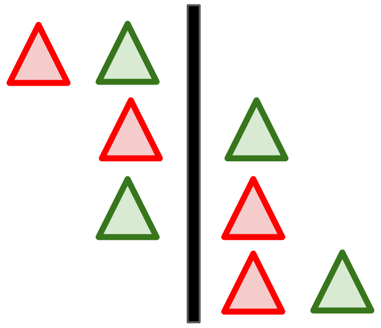
Partitions

- How many ways to split the elements of a set

We will see next week how to compute it in a few variations.

This is one of the more common problems in combinatorics. Partitions can be applied to answer questions like:

- Two sisters, **Jane** and **Joan**, are entering into a competition where the contenders are split into two different pools. In how many ways the sisters can be placed in the two available pools?



MERRIMACK COLLEGE

Wikipedia: Partition of a set.

A few Python tricks

Functions and Formatting the output

In the following explanations we will use a few Python functions and format the output of strings.

Those concepts were already used before, but now let's see it a little bit more in-depth.

Python **functions** are very simple and they employ a rather straightforward mechanism for parameter input and output.

Formatting strings can be useful for other purposes, but essentially, they are meant to produce good looking output with minimal effort.

Python Functions

Definition by an Identifier

- The same rules for variable identifiers are employed for function identifiers.
- All functions are defined by the reserved word **def** followed by the identifier, followed by parentheses encompassing the arguments (sometimes with default values), then a colon.

```
def average(a, b):
    return (a+b)/2

print(average(4, 8))        6.0

def ratio(c, d):
    if (d == 0):
        return c
    else:
        return c / d

print(ratio(3, 4))          0.75
print(ratio(3, 0))          3

def ratio(c, d=1):
    return c / d

print(ratio(3, 4))          0.75
print(ratio(3))             3.0
```

```
print(ratio(3))
```

👎

```
print(ratio(3, 0))
```

👎

MERRIMACK COLLEGE

Video: Python Functions.

Python Functions

Try it for 23, 59, 121, 189871, 191161, 9876542103, and 31381059607.

Function Example

- Every number can be decomposed into its prime number components;
- This example function, *decompose*, finds the prime numbers that decompose an input number ***n*** into its prime number components;
- Basically, this function tries dividing a number by all numbers ***i*** from 2 to itself (***range(2, n+1)***) and if ***n*** is divisible by ***i***, it appends ***i*** to the answer (***primes***) and divides ***n*** by ***i***;
    - The function could stop when ***n*** becomes 1.

```python
def decompose(n):
  """Decompose into primes"""
  primes = []
  for i in range(2, n + 1):
    while n % i == 0:
      primes.append(i)
      n //= i
  return primes
```

```
print(decompose(33))        [3, 11]
print(decompose(12))        [2, 2, 3]
print(decompose(105))       [3, 5, 7]
print(decompose(31))        [31]
```

MERRIMACK COLLEGE

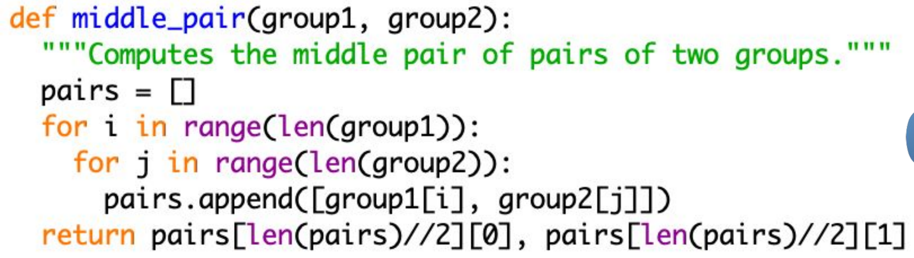
Text: Python Functions.

Python Functions

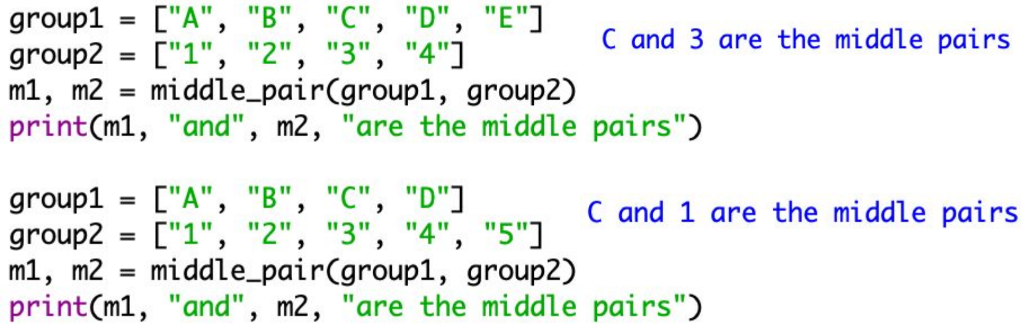
Try it for [2, 4, 6, 8] and [3, 9, 27], then for [4, 2, 1] and [1, 2, 3, 4, 5].

Output Parameters

- The explicit output of a function is done with the command ***return*** that is capable of handling as many expressions as you want;
- Multiple outputs of a function are handled as a multiple assignment command.



```python
def middle_pair(group1, group2):
  """Computes the middle pair of pairs of two groups."""
  pairs = []
  for i in range(len(group1)):
    for j in range(len(group2)):
      pairs.append([group1[i], group2[j]])
  return pairs[len(pairs)//2][0], pairs[len(pairs)//2][1]
```



```python
group1 = ["A", "B", "C", "D", "E"]
group2 = ["1", "2", "3", "4"]
m1, m2 = middle_pair(group1, group2)
print(m1, "and", m2, "are the middle pairs")
```

```
C and 3 are the middle pairs
```

```python
group1 = ["A", "B", "C", "D"]
group2 = ["1", "2", "3", "4", "5"]
m1, m2 = middle_pair(group1, group2)
print(m1, "and", m2, "are the middle pairs")
```

```
C and 1 are the middle pairs
```

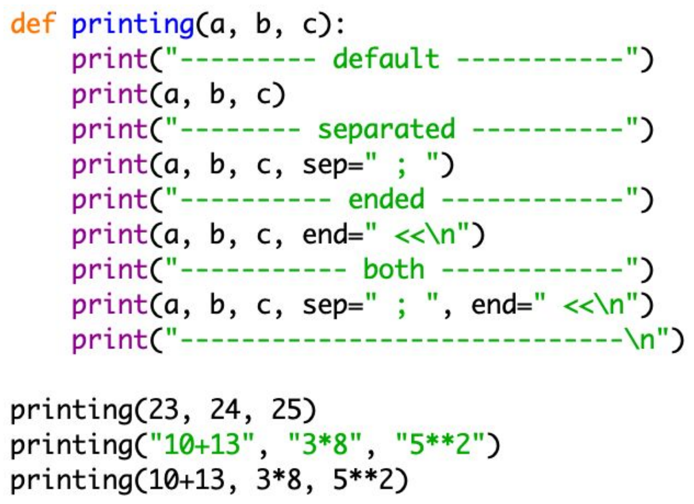
MERRIMACK COLLEGE

Text: Python Functions.

Formatting Output

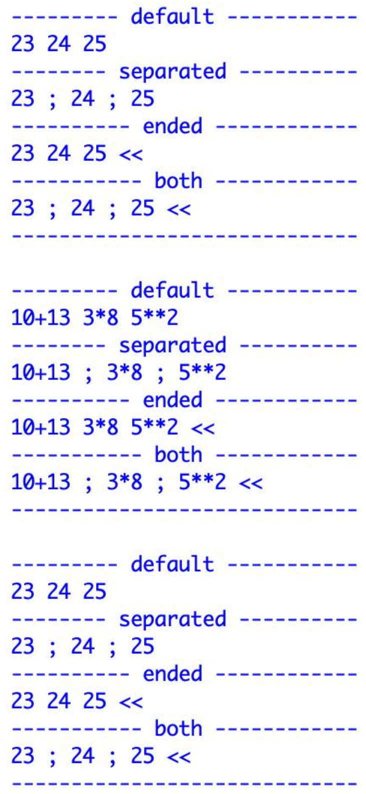
Using print function

- Here, every print command prints all arguments separated by a separator string, and ended by an ending string;
- You can use the default values of sep (a blank space) and end (a line break), or define your own.



```python
def printing(a, b, c):
    print("--------- default -----------")
    print(a, b, c)
    print("-------- separated ----------")
    print(a, b, c, sep=" ; ")
    print("---------- ended ------------")
    print(a, b, c, end=" <<\n")
    print("----------- both ------------")
    print(a, b, c, sep=" ; ", end=" <<\n")
    print("-----------------------------\n")

printing(23, 24, 25)
printing("10+13", "3*8", "5**2")
printing(10+13, 3*8, 5**2)
```



```
--------- default -----------
23 24 25
-------- separated ----------
23 ; 24 ; 25
---------- ended ------------
23 24 25 <<
----------- both ------------
23 ; 24 ; 25 <<
-----------------------------

--------- default -----------
10+13 3*8 5**2
-------- separated ----------
10+13 ; 3*8 ; 5**2
---------- ended ------------
10+13 3*8 5**2 <<
----------- both ------------
10+13 ; 3*8 ; 5**2 <<
-----------------------------

--------- default -----------
23 24 25
-------- separated ----------
23 ; 24 ; 25
---------- ended ------------
23 24 25 <<
----------- both ------------
23 ; 24 ; 25 <<
-----------------------------
```

```
sep = ' '        end = '\n'
```

MERRIMACK COLLEGE

Text: Python print() Function.

Formatting Output

The minimum place holders are just ***{ }***
You can refer to them using a keyword ***{name:}*** or
using a positional index starting from zero ***{0:}***
After the colon you can give formatting info ***{:<5}***

String formatting

- Strings can be filled with placeholders indicated by curly braces;
- These curly braces can be keyworded and perform several string conversions for strings and numbers.

```
n, y, today = "Paulo", 1998, 2023
print("My name is {name:>7}, I finished my PhD in {year:,} - {dif:^4} years ago".format(name=n, year=y, dif=today-y))
n, y, today = "Michael", 1983, 2023
print("My name is {name:>7}, I finished my PhD in {year:,} - {dif:^4} years ago".format(name=n, year=y, dif=today-y))
```

```
My name is   Paulo, I finished my PhD in 1,998 -  25  years ago
My name is Michael, I finished my PhD in 1,983 -  40  years ago
```

MERRIMACK COLLEGE

Video: Format specifiers in Python are awesome.

Formatting Output

The built-in function ***round***(*value*, *digits*) is a good option to trim long floating point numbers.

String formatting

- Strings can be filled with place holders indicated by curly braces;
- These curly braces can be indexed and perform several string conversions for numbers.

```
primes = [2, 3, 5, 7, 11, 13]
for p in primes:
    print("The fraction 1/{0:<2} is about {2:.3f}, but the full precision is {1:}".format(p, 1/p, round(1/p,5)))
```

```
The fraction 1/2  is about 0.500, but the full precision is 0.5
The fraction 1/3  is about 0.333, but the full precision is 0.3333333333333333
The fraction 1/5  is about 0.200, but the full precision is 0.2
The fraction 1/7  is about 0.143, but the full precision is 0.14285714285714285
The fraction 1/11 is about 0.091, but the full precision is 0.09090909090909091
The fraction 1/13 is about 0.077, but the full precision is 0.07692307692307693
```

Text: Python String format() Method.

Permutations

More and more I'm aware that the permutations are not unlimited.

Russel Hoban

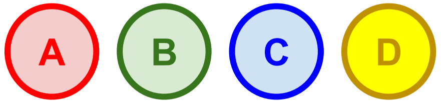
Given a set of elements, in how many ways can it be sorted?

And what would these permutations be?

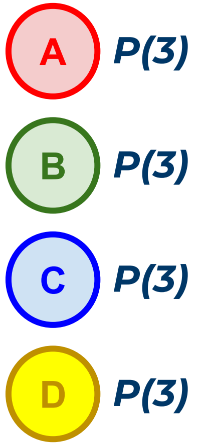
As we will see, permutations can be computed considering elements which are unique - or not:

- **Permutations**
- Permutations of multisets

Basic Permutations



- Let's assume a set of ***n*** elements, for example, for ***n* = 4**, let's consider the set ***{ A , B , C , D }***
- How many ways can we sort these ***n*** elements? Let's call it ***P(n)***
  - There will be some permutations starting with the first element (***A***), some with the second (***B***), and so on... (***n*** permutations)
  - ... and for all of then the ***n-1*** elements can be permuted among themselves
  - Repeating it for f(n-1) we find n-1 permutations of n-2
  - Finally, we know the one single element has only one possible permutation.



$$P(4) = 4 * P(3)$$
$$P(3) = 3 * P(2)$$
$$P(2) = 2 * P(1)$$

$$P(n) = n * P(n-1)$$

$$P(1) = 1$$

$$P(4) = 4 * 3 * 2 * 1 = 4! = 24$$

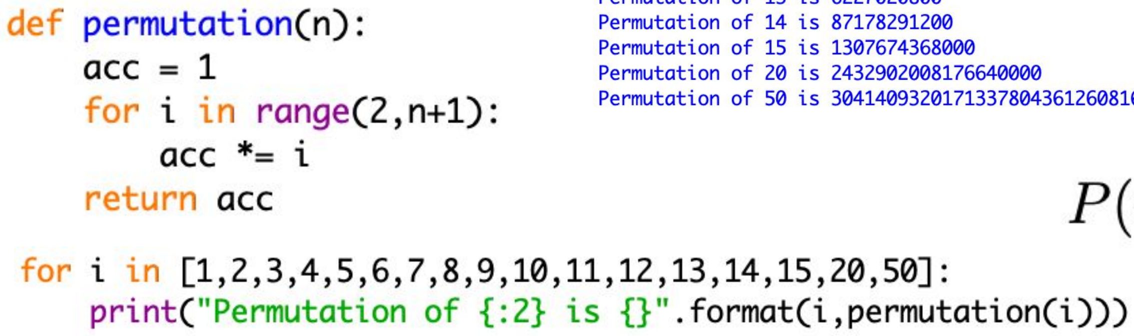
$$\boxed{P(n) = n!}$$

Wikipedia: Permutation.

Computing Permutations

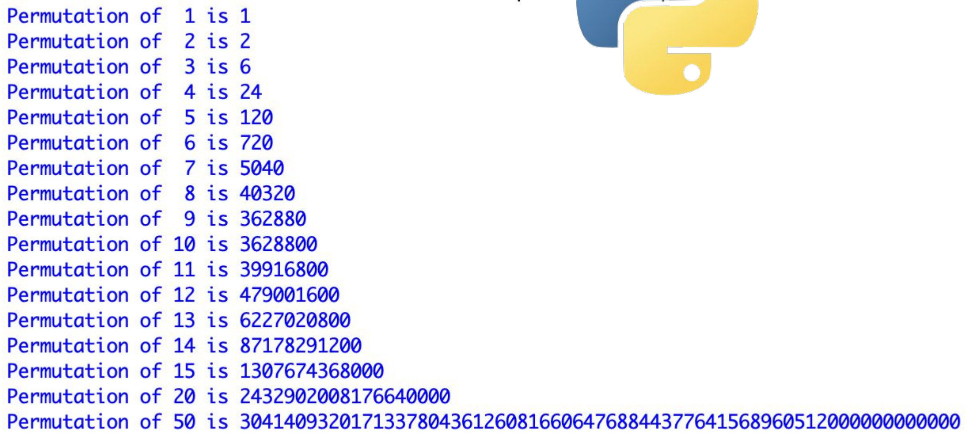
Counting Basic Permutations

- Since ***P(n) = n!***
- The actual number of permutation can be defined by the function:



```python
def permutation(n):
    acc = 1
    for i in range(2,n+1):
        acc *= i
    return acc

for i in [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,20,50]:
    print("Permutation of {:2} is {}".format(i,permutation(i)))
```



```
Permutation of  1 is 1
Permutation of  2 is 2
Permutation of  3 is 6
Permutation of  4 is 24
Permutation of  5 is 120
Permutation of  6 is 720
Permutation of  7 is 5040
Permutation of  8 is 40320
Permutation of  9 is 362880
Permutation of 10 is 3628800
Permutation of 11 is 39916800
Permutation of 12 is 479001600
Permutation of 13 is 6227020800
Permutation of 14 is 87178291200
Permutation of 15 is 1307674368000
Permutation of 20 is 2432902008176640000
Permutation of 50 is 30414093201713378043612608166064768844377641568960512000000000000
```

$$P(n) = n!$$

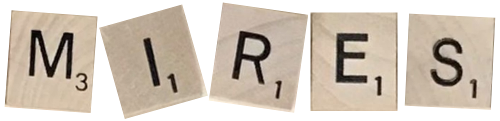
MERRIMACK COLLEGE

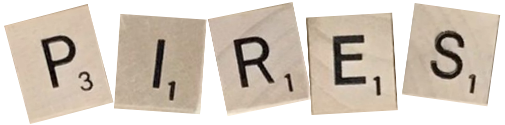
Try it!

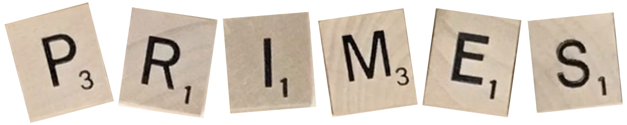
Applying Permutations

Playing Scrabble

- While playing Scrabble, you receive 6 different letters, for example:
  - "E", "I", "M", "P", "R", and "S"
- Ignoring the English words that actually exist, what would be the hypothetical maximum number of six- or five-letter words that could be made with the 6 letters you received?







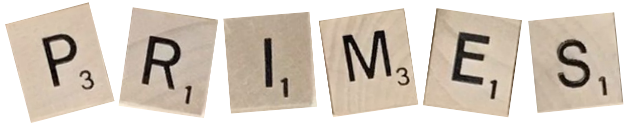
Answer in the next slide - don't look at it before thinking it through!

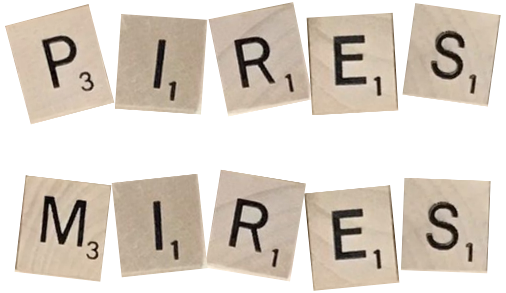
MERRIMACK COLLEGE

Applying Permutations

Playing Scrabble

- Ignoring the English words that actually exist, what would be the hypothetical maximum number of six- or five-letter words that could be made with the 6 letters you received?
- P(6) = 720
- P(5) = 120
  - You have 6 possible sets of 5 letters, each one taking one letter out, thus:
    - 6 times P(5)





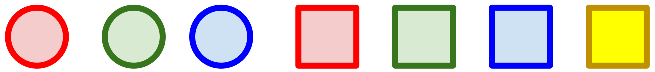
MERRIMACK COLLEGE

Adding up all possibilities: 720 + (120 * 6) = **1440**

Combining Permutations

If you have two groups (actually two sets) named **A** and **B**, containing ***n*** and ***m*** elements respectively, how many permutations would you have for these ***n+m*** elements while always keeping the elements of **A** before the elements of **B**?

For example: ***n = 3*** and ***m = 4***



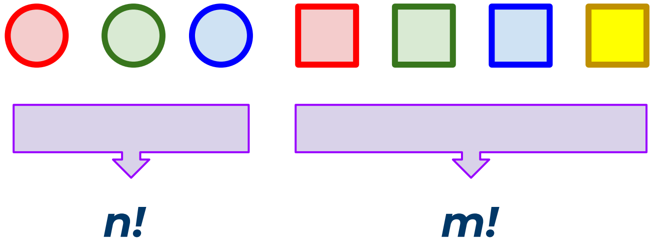
If you could mix them all it would be: ***(n+m)!***

7! = 5040

But each one apart would be

n! * m!

3! * 4! = 144



Merrimack College

Calculator: Permutations (and more).

Permutations

More and more I'm aware that the permutations are not unlimited.

Russel Hoban

Given a set of elements, in how many ways can it be sorted?

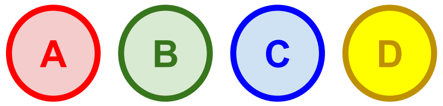
And what would these permutations be?

As we will see, permutations can be computed considering elements which are unique - or not:

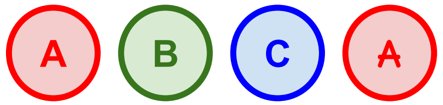
- Permutations
- **Permutations of multisets**

Permutations of Multisets

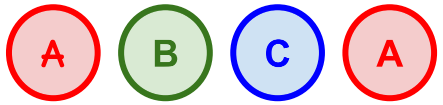
- Let's assume a set of ***n*** elements, but now with ***m*** equal elements in it.
  - For example, if ***n = 4*** and ***m = 2***, we could have the set ***{ A , B , C , A }***
- If they were all different, the number of permutations would be ***n!***, but since the ***m*** equal elements could be freely interchanged, these permutations of ***m*** elements can be taken out of the total number of combinations, thus the total number of permutations of ***n*** elements, where ***m*** are equal is denoted as ***P(n / m)*** and computed as:
  - For the example ***n = 4*** and ***m = 2*** with the set ***{ A , B , C , A }*** it will be:
    - ***P (4 / 2) = 4! / 2! = 24 / 2 = 12***



P(4) = 4! = 24



All combinations starting and ending with an A are the same



P(n / m) = n! / m!

MERRIMACK COLLEGE

a.k.a. Permutation with repetition.

Permutations of Multisets

- Generalizing the previous formula we might consider that the permutation of a set of ***n*** elements where there are ***j*** of distinct elements each kind with m_1, m_2, … m_j.
  - For example if you take the set of letters composing the word MONTANA you have ***n=7*** elements **{M,O,N,T,A,N,A}**, being ***k=5*** distinct letters $m_1 = 1$ for **M**, $m_2 = 1$ for **O**, $m_3 = 2$ for **N**, $m_4 = 1$ for **T**, and $m_5 = 2$ for **A**.
- As such the number of different permutations will be the factorial of ***n*** divided by the product of the factorials of m_1 to m_j:

$$P\left(\frac{n}{m_1\, m_2 \ldots m_j}\right) = \frac{n!}{m_1!\, m_2! \ldots m_j!}$$

The case with no repetition can be computed as k = n, therefore:

P(n) = n! / (1! 1! … 1!)

{M,O,N,T,A,N,A}

P(7 / 1 1 2 1 2) =

7! / (1! 1! 2! 1! 2!) =

5040 / 4 = 1260

MERRIMACK COLLEGE

Video: Permutations of a Multiset.

Permutations of Multisets

What is the number of permutations for the set of the letters of the word ***ABRACADABRAR*** ?

- ***n = 12***

$$P\left(\frac{n}{m_1\, m_2 \ldots m_j}\right) = \frac{n!}{m_1!\, m_2! \ldots m_j!}$$

- for A: $\boldsymbol{m_1 = 5}$
- for B: $\boldsymbol{m_2 = 2}$
- for R: $\boldsymbol{m_3 = 3}$
- for C: $\boldsymbol{m_4 = 1}$
- for D: $\boldsymbol{m_5 = 1}$

Answer: 332 640

Important

Some authors refer to *permutations with repetition* when you state as ***n*** the number of distinct elements and then the combination of those elements is computed.

For example with ***n=2*** being the set of distinct elements ***{A, B}*** there are these ***4*** of those so-called *permutations with repetition*:

{A,B}, ***{B,A}***, ***{A,A}*** and ***{B,B}***

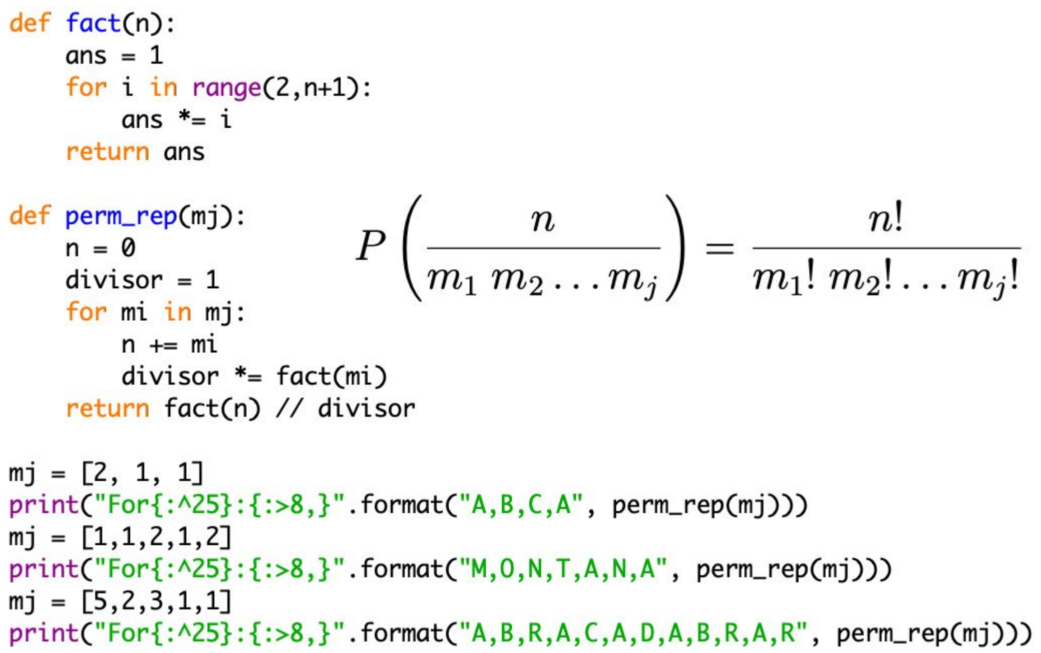
However, those are not really permutations, as they're more accurately defined as ***words over an alphabet***.

MERRIMACK COLLEGE

a.k.a. Permutation with repeats.

Computing Permutations of Multisets

Given a original set with the number of elements of each kind (***m_i*** with ***i*** going from ***1*** to ***j***), it is possible to compute the permutation with repetition of n (sum of all ***m_i*** is equal to ***n***).

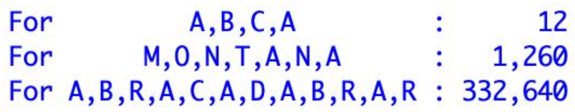


```
def fact(n):
    ans = 1
    for i in range(2,n+1):
        ans *= i
    return ans

def perm_rep(mj):
    n = 0
    divisor = 1
    for mi in mj:
        n += mi
        divisor *= fact(mi)
    return fact(n) // divisor

mj = [2, 1, 1]
print("For{:^25}:{:>8,}".format("A,B,C,A", perm_rep(mj)))
mj = [1,1,2,1,2]
print("For{:^25}:{:>8,}".format("M,O,N,T,A,N,A", perm_rep(mj)))
mj = [5,2,3,1,1]
print("For{:^25}:{:>8,}".format("A,B,R,A,C,A,D,A,B,R,A,R", perm_rep(mj)))
```

$$P\left(\frac{n}{m_1\, m_2 \ldots m_j}\right) = \frac{n!}{m_1!\, m_2! \ldots m_j!}$$



```
For         A,B,C,A         :      12
For      M,O,N,T,A,N,A      :   1,260
For A,B,R,A,C,A,D,A,B,R,A,R : 332,640
```

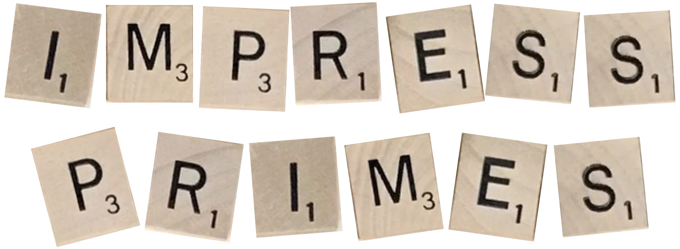
MERRIMACK COLLEGE

Try it!

Applying Permutations of Multisets

Playing Scrabble … again

- Now you receive 7 letters with one double, for example:
  - "E", "I", "M", "P", "R", "S", and "S"
- Ignoring the English words that actually exist, what would be the hypothetical maximum number of seven- or six-letter words that could be made with the 7 letters (1 double) you received?

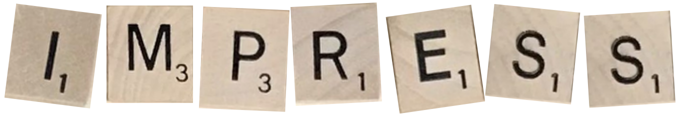


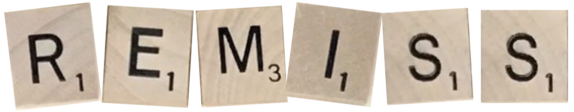
Answer in the next slide - don't look at it before thinking it through!

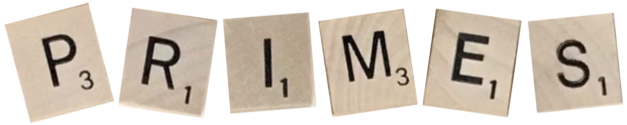
Applying Permutations of Multisets

Playing Scrabble … again

- Ignoring the English words that actually exist, what would be the hypothetical maximum number of seven- or six-letter words that could be made with the 7 letters (1 double) you received?
- P(7 / 2) = 2520
- P(6 / 2) = 360 (if you take one of the 5 non-doubled letters)
- P(6) = 720 (if you take one of the doubled letters)







MERRIMACK COLLEGE

Adding up all possibilities 2520 + (360 * 5) + 720 = **5040**

Partial Permutations

The mind has its arrangement; it proceeds from principles to demonstrations. The heart has a different mode of proceeding.

Blaise Pascal

Given a set of elements, in how many ways can it be sorted, but not necessarily use all of its elements?

This is called a partial permutation or an arrangement.

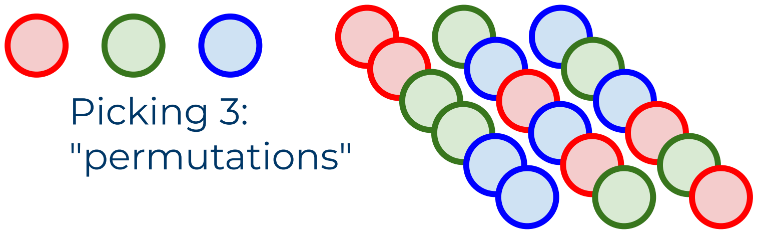
As we will see these arrangements are important to many applications, specifically we will see how to use:

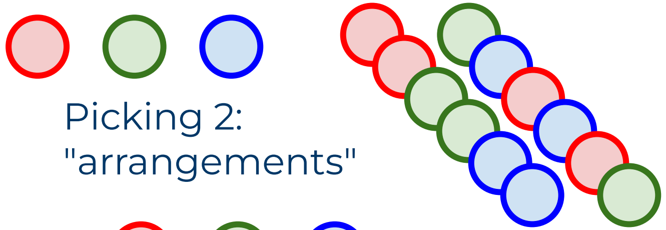
- **Arrangements**;
- Arrangements of Multisets.

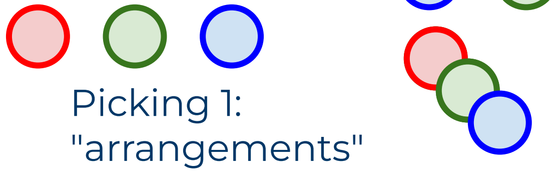
Arrangements, a.k.a. Partial Permutations

The permutations we have seen consider ways of sorting ***n*** elements of a set in all possible orders. In other words, they determine how many distinct lists can be produced with the same size ***n*** as the original set.

Now we will see the same idea, but producing lists with a specific size ***k*** that is smaller than or equal to ***n***, thus, a partial permutation. Since this can also be described as different ways to arrange the elements of the original set in lists of size ***k***, this idea is often referred as *arrangements*. We call it: arrangement of ***k*** elements out of ***n***.







MERRIMACK COLLEGE

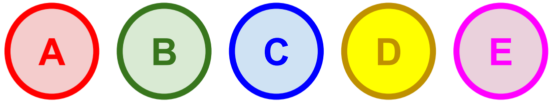
Video: A Closer look into **k** permutations of **n** elements.

From Permutations to Partial Permutations

- All possible permutations for a set of ***n*** elements are ***n!***
- If we limit it to just the first ***k*** elements we are leaving out ***n-k*** elements and their permutations
    - the number of permutations of ***n-k*** elements is given by ***(n-k)!***
        - Thus, the arrangements of ***k*** elements out of ***n*** will be denoted by ***A(k, n)*** and computed as:

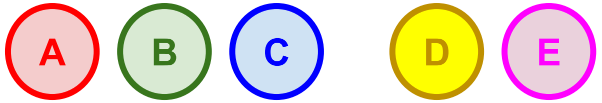
$$A(k,n) = \frac{n!}{(n-k)!}$$

a.k.a. Arrangements



P(5) = 5! = 120

Now considering not 5, but just 3 elements:



A(3,5) = 5! / 2! = 60

Calculator: K-Permutations.

Computing Arrangements

- To compute the arrangements:

```python
def fact(n):
    acc = 1
    for i in range(2,n+1):
        acc *= i
    return acc

def arrangement(k, n):
    return fact(n) // fact(n-k)

def arrang(k, n):
    ans = 1
    for i in range(n, n-k, -1):
        ans *= i
    return ans

for n in range(3, 8):
    for k in range(1,n+1):
        print(k, "{} elements out of {} is {}".format(k, n, arrang(k, n)))
    print("-----")
```

$$A(k,n) = \frac{n!}{(n-k)!}$$

$$A(3,7) = \frac{7!}{(7-3)!} = \frac{7!}{4!} = \frac{7\times6\times5\times4\times3\times2\times1}{4\times3\times2\times1} = 7\times6\times5$$

```
1 1 elements out of 3 is 3
2 2 elements out of 3 is 6
3 3 elements out of 3 is 6
-----
1 1 elements out of 4 is 4
2 2 elements out of 4 is 12
3 3 elements out of 4 is 24
4 4 elements out of 4 is 24
-----
1 1 elements out of 5 is 5
2 2 elements out of 5 is 20
3 3 elements out of 5 is 60
4 4 elements out of 5 is 120
5 5 elements out of 5 is 120
-----
1 1 elements out of 6 is 6
2 2 elements out of 6 is 30
3 3 elements out of 6 is 120
4 4 elements out of 6 is 360
5 5 elements out of 6 is 720
6 6 elements out of 6 is 720
-----
1 1 elements out of 7 is 7
2 2 elements out of 7 is 42
3 3 elements out of 7 is 210
4 4 elements out of 7 is 840
5 5 elements out of 7 is 2520
6 6 elements out of 7 is 5040
7 7 elements out of 7 is 5040
-----
```

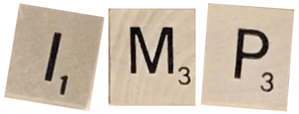
MERRIMACK COLLEGE

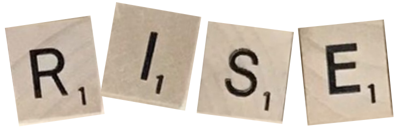
Try it!

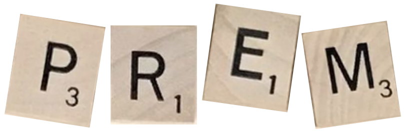
Applying Arrangements

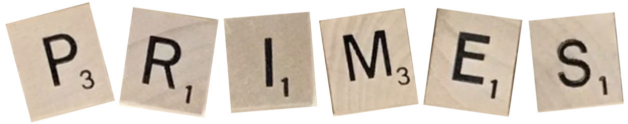
Playing Scrabble ... one more time ...

- While playing Scrabble, you receive 6 different letters, for example:
  - "E", "I", "M", "P", "R", and "S"
- Ignoring the English words that actually exist, what would be the hypothetical maximum number of *at least* three-letter words that could be made with the 6 letters you received?









Answer in the next slide - don’t look at it before thinking it through!

Applying Arrangements

Playing Scrabble … one more time …

- Ignoring the English words that actually exist, what would be the hypothetical maximum number of *at least* three-letter words that could be made with the 6 letters you received?

- P(6) = 720
- A(5,6) = 720
- A(4,6) = 360
- A(3,6) = 120

Adding up all possibilities 720 + 720 + 360 + 120 = ***1920***

Partial Permutations

The mind has its arrangement; it proceeds from principles to demonstrations. The heart has a different mode of proceeding.

Blaise Pascal

Given a set of elements, in how many ways can it be sorted, but not necessarily use all of its elements?

This is called a partial permutation or an arrangement.

As we will see these arrangements are important to many applications, specifically we will see how to use:

- Arrangements;
- **Arrangements of Multisets**.

Arrangements of Multisets

- The permutations of ***n*** elements is equal to ***n*** factorial;

$$P(n) = n!$$

- If you have permutation of n elements, where you have multisets ***m*** $\boldsymbol{m_i}$ of equal elements, you need to divide by the factorial of the subsets sizes;

$$P\left(\frac{n}{m_1\, m_2 \ldots m_j}\right) = \frac{n!}{m_1!\, m_2! \ldots m_j!}$$

- If you have arrangements of ***k*** elements out of ***n*** you need to divide by the factorial of ***n-k***;

$$A(k, n) = \frac{n!}{(n-k)!}$$

- Therefore, if you have arrangements of ***k*** elements out of ***n***, where you have subsets of equal elements $\boldsymbol{m_i}$, what do you imagine the formula will be?

$$A\left(k, \frac{n}{m_1\, m_2 \ldots m_j}\right) = ?$$

MERRIMACK COLLEGE

Before going to the next slide, please think it through!

Arrangements of Multisets

- The number of arrangements of ***k*** elements out of ***n***, where you have subsets of equal elements ***m***$_j$ can be computed, similarly to the permutations of multisets, by taking out the permutation of equal elements, i.e.:

$$A\left(k, \frac{n}{m_1\, m_2 \ldots m_j}\right) = \frac{A(k,n)}{m_1!\, m_2! \ldots m_j!}$$

- Expanding it:

$$A\left(k, \frac{n}{m_1\, m_2 \ldots m_j}\right) = \frac{n!}{(n-k)!m_1!\, m_2! \ldots m_j!}$$

Video: Permutations with Repetition.

Applying Arrangements of Multisets

Playing Scrabble … one more time …

- While playing Scrabble you receive 7 letters with one double, for example:
    - "E", "I", "M", "P", "R", "S", and "S"
- Ignoring the English words that actually exist, what would be the hypothetical maximum number of at least three-letter words that could be made with the 7 letters (one doubled) you received?

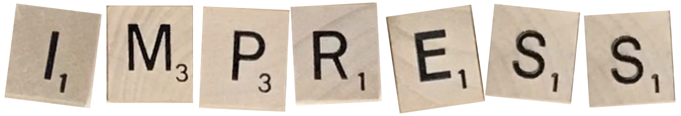
Answer in the next slide - don’t look at it before thinking it through!

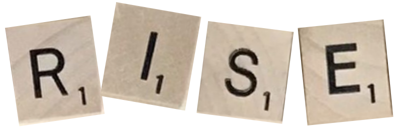
Applying Arrangements

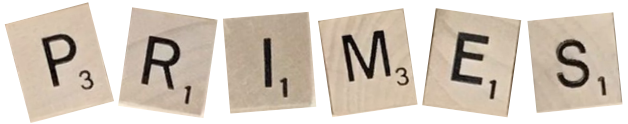
Playing Scrabble … one more time …

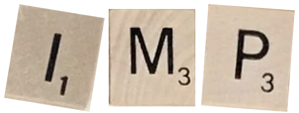
- Ignoring the English words that actually exist, what would be the hypothetical maximum number of at least three-letter words that could be made with the 7 letters (one doubled) you received?

- P(7 / 2) = 2520
- A(6, 7 / 2) = 2520
- A(5, 7 / 2) = 1260
- A(4, 7 / 2) = 420
- A(3, 7 / 2) = 105









Adding up all possibilities:
2520 + 2520 + 1260 + 420 + 105 = ***6825***

This Week's tasks

- Post discussion D#4
- Coding Project P#4
- Quiz Q#4

Tasks

- Post in the discussion one true and one false application of permutations.
- Coding Project #4, computing the arrangements of multisets.
- Quiz #4 about this week topics.

Post Discussion - Week 4 - D#4

Double work here!

Post in the discussion two practical problems:

- One that you could correctly solve using permutations;
- Another that cannot be correctly solved using permutations as seen today.

However, do not say which one is which, as it will be the job of your colleagues to analyze the problems in order to say which one could be solved with permutations, and why. Additionally, you have to check out your colleagues' replies to see if they got it right or not!

Your task:

- Post your discussion in the message board by this Monday;
- Reply to your colleagues' posts in the message board by next Thursday.

This task counts towards the Discussions grade.

Fourth Coding Project - Week 4 - P#4

- Write a Python program to compute Arrangements of Multisets that:
  - Asks the user a number of subsets (no smaller than 3, no greater than 8);
    - ***j***
  - Asks the user the size of each subset (from 1 to 5);
    - m_i with ***i*** going from ***1*** to ***j***
  - Asks the user the total number of the arrangement (a number smaller than the sum of sizes of the subsets - ***n***)
    - ***k***
  - Computes and prints the number of arrangements of ***k*** elements out of ***n***, considering the subsets of size m_i

Example in the next slide

Your task:

- Go to Canvas, and submit your Python file (.py) within the deadline:
  - The deadline for this assignment is next Thursday.

MERRIMACK COLLEGE

This assignment counts towards the Projects grade.

Fourth Coding Project - Week 4 - P#4

- For example:
  - If the user enters the value of ***j*** as ***4***;
  - If the user enters $\boldsymbol{m_1}$ as ***3***, $\boldsymbol{m_2}$ as ***1***, $\boldsymbol{m_3}$ as ***2***, $\boldsymbol{m_4}$ as ***5***;
  - If the user enters ***k*** as ***6***;
  - Then your program should compute the number of arrangements of ***k*** elements out of ***n***, considering the subsets of size $\boldsymbol{m_i}$ which would be:
    - 11! / (11-6)! * 3! * 1! * 2! * 5!   = ***231*** arrangements
  - Then it should print:
    - "***Given your inputs, the number of arrangements is 231***"

Your task:

- Go to Canvas, and submit your Python file (.py) within the deadline:
  - The deadline for this assignment is next Thursday.

MERRIMACK COLLEGE

This assignment counts towards the Projects grade.

Fourth Quiz - Week 4 - Q#4

- The fourth quiz in this course covers the topics of Week 4;
- The quiz will be available this Saturday, and it is composed of 10 questions;
- The quiz should be taken on Canvas (Module 4), and it is not timed:
  - You can take as long as you want to answer it;
- The quiz is open book, open notes, and you can even use any language Interpreter to answer it;
- Yet, the quiz is evaluated and you are allowed to submit it only once.

Your task:

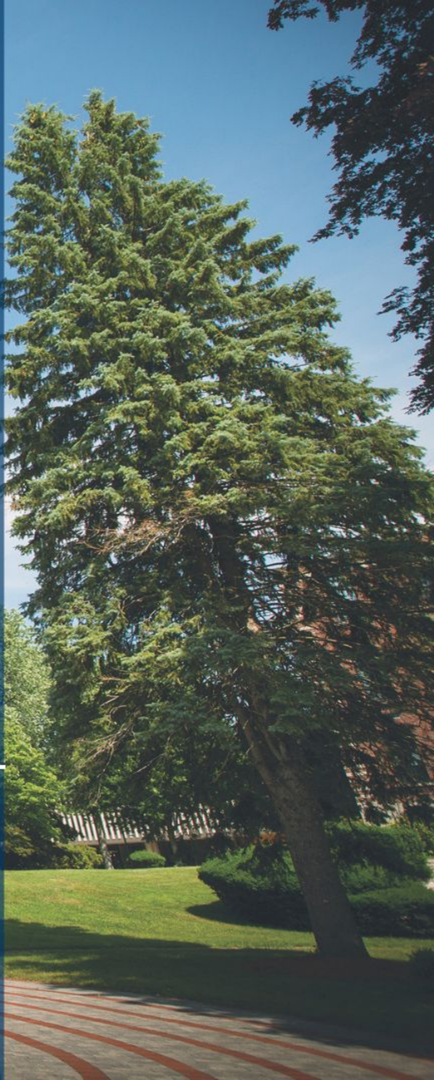
- Go to Canvas, answer the quiz and submit it within the deadline:
  - The deadline for the quiz is Next Thursday.

This quiz counts towards the Quizzes grade.

We are in Week 4 of CSC 6000

- Post discussion D#4 by Monday, reply to your colleagues by next Thursday;
- Do quiz Q#4 (available Saturday) by next Thursday;
- Develop coding project P#4 by next Thursday.

Next Week - Combinatorics and Binomial Coefficients



MERRIMACK COLLEGE