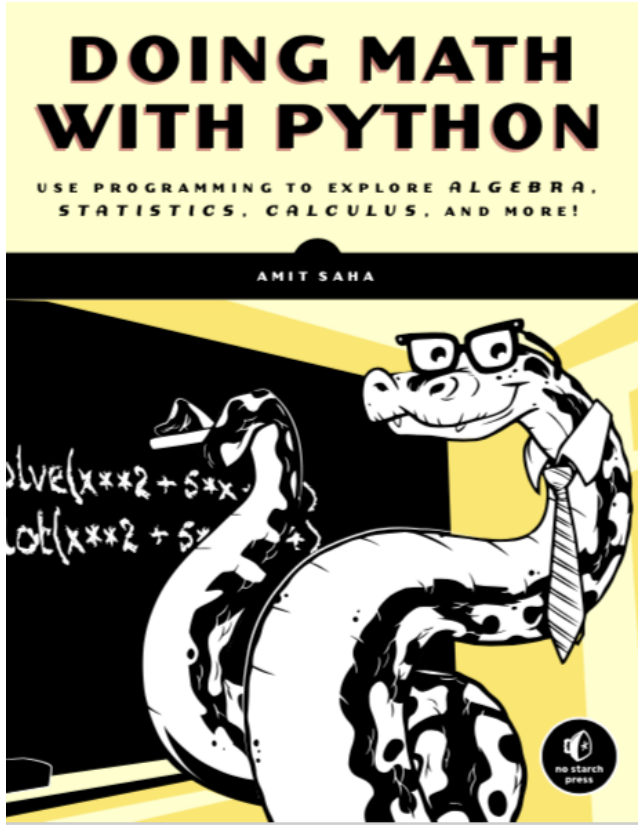
# Discrete Structures: CMPSC 102

Oliver BONHAM-CARTER

Fall 2018
Week 14

# Where We Are?




## Saha, Chapter 5: Playing with sets and probability

- Using Sets with Sympy
- Containing probabilities in sets.

# Using Sympy

## Clone the GitHub Repository

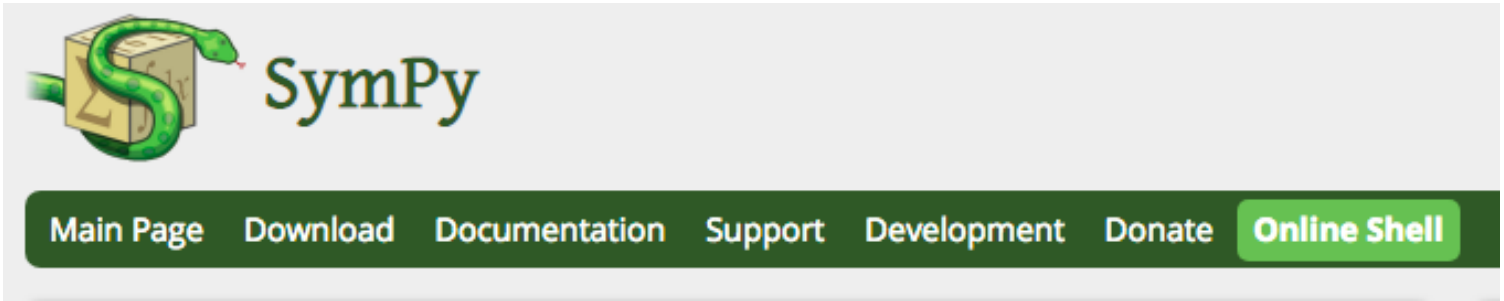
```
git clone git://github.com/sympy/sympy.git
```

## Install locally

```
python3 setup.py install
```

## Or use the Interactive shell online

```
https://live.sympy.org/
```



# Remember *Sets*

- Sets have no order and all members are unique.
- Sets are also symbolically manipulated in SymPy

## Sets, The old way

```
set([1,2,2,2,2,2,3]) == set([3,2,1])
```

Note: If the libraries do not exist, try using python (version 2) or the SymPy website's interactive interpretor, `https://live.sympy.org/`

## Sets, Working with Sympy

```
from sympy import FiniteSet
FiniteSet(1,2,2,3,3,3,3) == FiniteSet(1,3,2)
```

# Construction and Membership



## Let's build a bigger set

```
from sympy import FiniteSet
from fractions import Fraction
s = FiniteSet(1, 1.5, Fraction(1, 5))
print(s) #{1/5, 1, 1.5}
```

## What's in the set

```
print(" Number of elements :",len(s))
for i in s: print(i)
1 in s # does this value exist in the set
Fraction(1,5) in s
```

# Converting Tuples to FiniteSets



## Converting a list to a FiniteSet

```
m = [1,2,3] # list
s = FiniteSet(*m)
print(s) #{1, 2, 3}
type(s) #<class 'sympy.sets.sets.FiniteSet'>
```

```
m = [1, 2, 3, 2] # list
s = FiniteSet(*m)
type(s) # <class 'sympy.sets.sets.FiniteSet'>
```

## Iterating through set

```
s = FiniteSet(1, 2, 3)
for member in s:
  print(member)
```

# Unions and Intersections



## Union of sets

```
from sympy import FiniteSet
s = FiniteSet(1, 2, 3)
t = FiniteSet(2, 4, 6)
u = FiniteSet(3, 5, 7)
s.union(t).union(u)
```

## Intersection of sets

```
s = FiniteSet(1, 2, 3)
t = FiniteSet(2, 4, 6)
u = FiniteSet(3, 5, 7)
s.intersect(t).intersect(u) #EmptySet() why?
```

# Empty Sets

## The loneliest set ever...



$$\emptyset = \{\}$$

The empty set contains the *element* of nothing

- In set theory, *nothing* is actually *something* to note: here we imply that there are no members in the set
- The empty set contains nothing, and is denoted by the symbol: $\emptyset$

### Creating an empty set

```
from sympy import FiniteSet
s = FiniteSet()
print(s) #EmptySet()
```

# Finite and Infinite

$$A = \{-\infty, \cdots, \infty\}$$

The set of all real numbers, $A_i \in A$

## Defining sets

- Set of even numbers: E $= \{\cdots, -4, -2, 0, 2, 4, \cdots\}$
- Set of odd numbers: O $= \{\cdots, -3, -1, 1, 3, \cdots\}$
- Set of prime numbers: P $= \{2, 3, 5, 7, 11, 13, 17, \cdots\}$
- Positive multiples of 3 that are less than 10: L $= \{\cdots, 3, 6, 9\}$
- Set of the first five letters: F $= \{a, b, c, d, e\}$

## Members in a set (True? False?)

- Is $0 \in E$ ? Is $5 \in O$ ? Is $13 \in P$ ?
- Is $90 \in E$ ? Is $4 \in P$ ? Is $f \in F$ ?

# Who is in Which Set?



## Members of a set

- $0 \in E$: 0 is a member of $E$
- $4 \notin O$: 4 is not a member of $O$
- $4000 \in E$
- $8 \notin P$
- $59 \notin P$
- $5 \in P$
- $3 \notin F$
- $\diamondsuit \notin F$

# Who is in Which Set?



## Members of a set

- $0 \in E$: 0 is a member of $E$
- $4 \notin O$: 4 is not a member of $O$
- $4000 \in E$

## The set of real numbers between 0 and 1

```
from sympy import Interval
Interval(0, 1).contains(0.5)
0.5 in Interval(0,10)
```

# Proper Subsets

## What is a *Proper Subset*?

- A proper subset of a set $A$ is a subset that cannot be equal to $A$
- If $B$ is a proper subset of $A$, then all elements of $B$ are also in $A$ but $A$ contains at least one element that is not in $B$.

- Ex: Let $A = \{1, 3, 5\}$ then $B = \{1, 5\}$ is a proper subset of $A$. The set $C = \{1, 3, 5\}$ is a subset of $A$, but it is not a proper subset of $A$ since $C = A$. The set $D = \{1, 4\}$ is not even a subset of $A$, since 4 is not an element of $A$.

## The sets of $A$ and $B$

```
A = FiniteSet(1,3,5)
B = FiniteSet(1,5)
for i in B: i in A # is each element in A?
for i in A: i in B # is each also element in B?
len(A) == len(B) # same cardinality?
```

# Proper Subsets

- **Example**: $A = 1, 2, 3, 4, 5$
  - Subsets of $A$: $\{1, 2, 3\}$, $\{3, 4\}$ and $\{1\}$
  - Written: $\{1, 2, 3\} \subset A$,
  - $\{3, 4\} \subset A$,
  - $\{1\} \subset A$
  - Note: $\{1, 6\}$ is not a subset, since it has an element (6) which is not in the parent set.

## Is $B$ a subset of $A$?

```
A = FiniteSet(1, 2, 3, 4, 5)
B = FiniteSet(1, 6) #potential subset?
for i in B: i in A # is each element in A?
for i in A: i in B # is each also element in B?
len(A) == len(B) # same cardinality?
```

# Proper Subsets

- $A = \{1, 2, 3, 4, 5\}$
- $B = \{1, 2, 3\}$

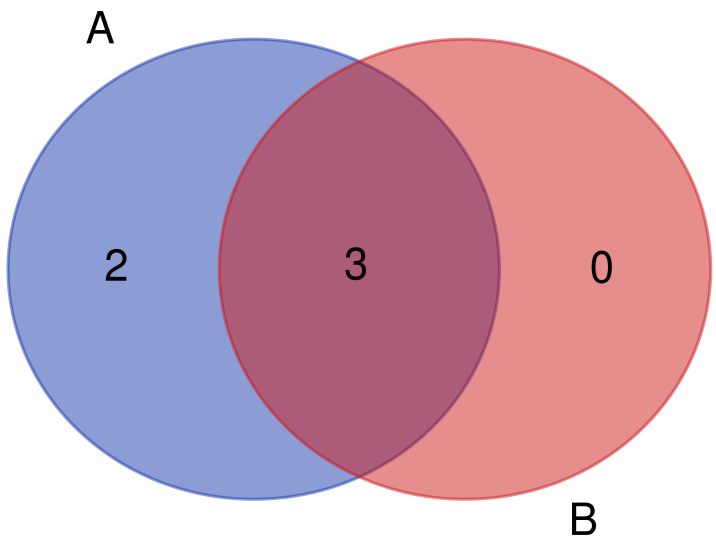



Figure: $B \subset A$ since there are other elements in $A$ that are not in $B$

# Proper Subsets

- **Another Example**: Is $A$ a proper subset of $B$?
- Let $A = \{1, 3, 4\}$ and let $B = \{1, 4, 3, 2\}$?
  - 1 is in $A$, and 1 is in $B$ as well. (good, so far!)
  - 3 is in $A$ and 3 is also in $B$.
  - 4 is in $A$, and 4 is in $B$.
  - We have covered all elements of $A$, and each is in $B$ and so we stop here.
- Yes, $A$ is a proper subset of $B$ since the sets cannot be equal (more in $B$ than in $A$)
- $A \subset B$

# Subsets

- A set, $s$, is a subset of another set, $t$, if all the members of $s$ are also members of $t$.
- For example, the set 1 is a subset of the set 1, 2. You can check whether a set is a subset of another set using the `is_subset()` method:

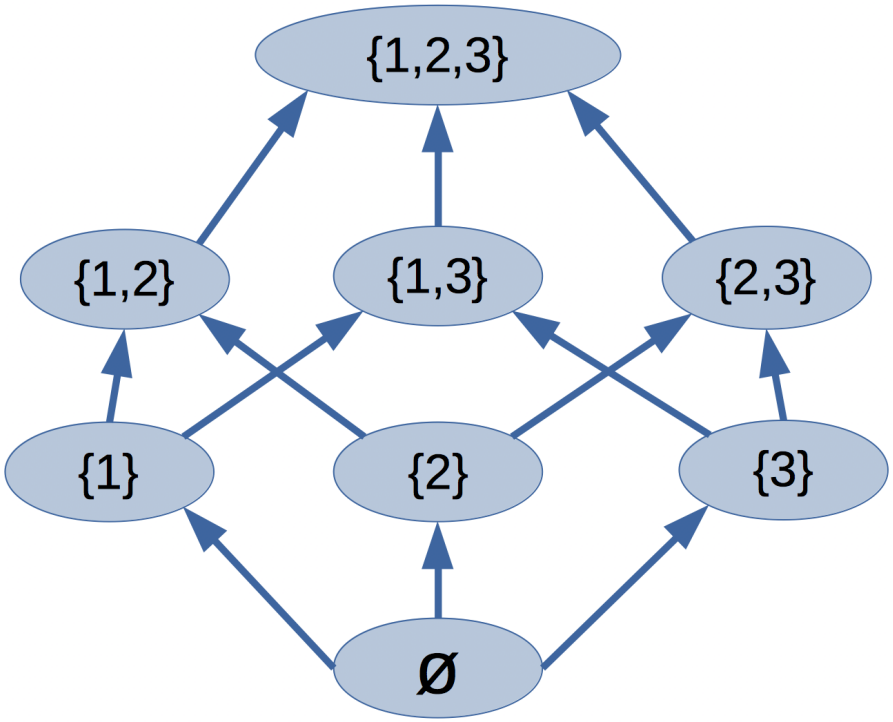
## Subsets

```
s = FiniteSet(1)
t = FiniteSet(1,2)
s.is_subset(t) #True
t.is_subset(s) #False
```

# Subsets Make up Powersets



A Power Set is a set of all the subsets of a set.



# Subsets

A Power Set is a set of all the subsets of a set.



## Putting it together

- The set $\{1, 2, 3\}$:
- Contains The empty set ($\emptyset$) $\{\ \}$ and is a subset
- Contains subsets: $\{1\}$, $\{2\}$ and $\{3\}$
- Contains subsets: $\{1, 2\}$, $\{1, 3\}$ and $\{2, 3\}$
- Contains $\{1, 2, 3\}$ and is a subset of self

## The Subsets of a Powerset

```
s = FiniteSet(1,2,3)
print(s) #{1, 2, 3}
ps = s.powerset()
print(ps)
# {EmptySet(), {1}, {2}, {3},
# {1, 2}, {1, 3}, {2, 3}, {1, 2, 3}}
len(ps) # set cardinality: number of elements
```

# A Simple Application

## Create a coding system



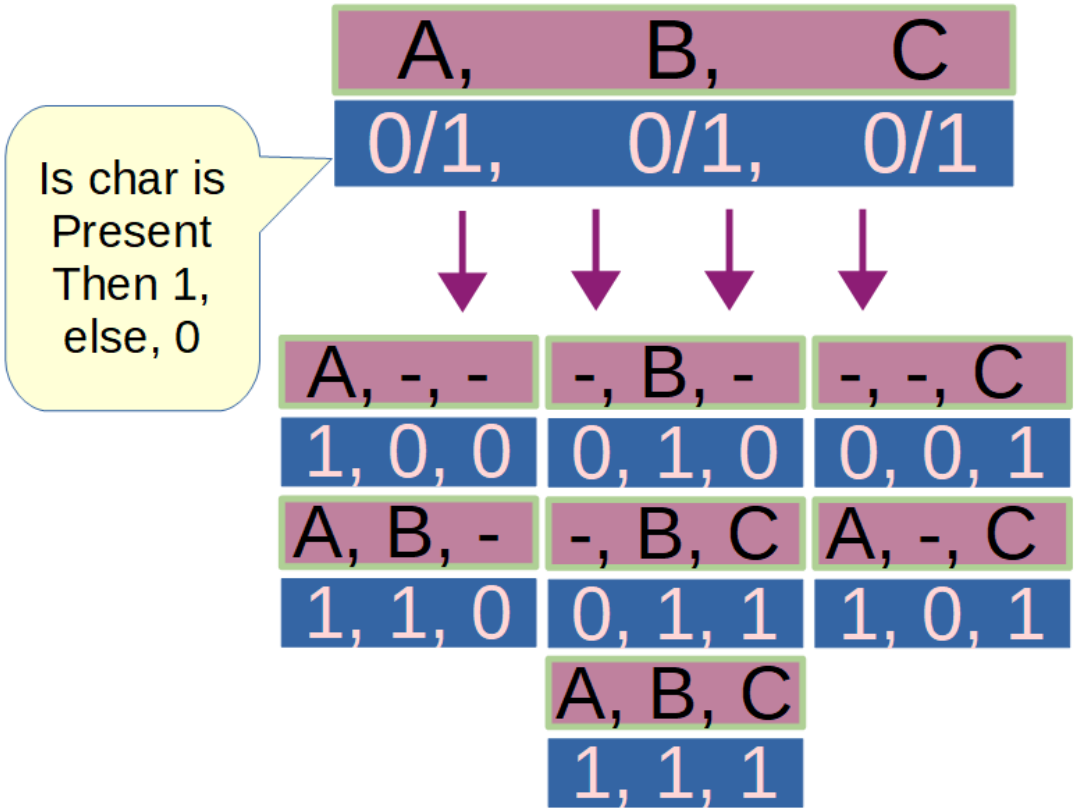
We have three characters that we wish to use to create a coding system to send binary signals over a channel. We want as many unique (binary) codes as possible from these three chars. How many codes can we create and what are these codes?

# A Simple Application

## Use the Powerset



```
b = FiniteSet('a','b','c')
b.powerset()
#{EmptySet(), {a}, {b}, {c}, {a, b}, {a, c},
# {b, c}, {a, b, c}}
```



# A Simple Application

## Use the Powerset

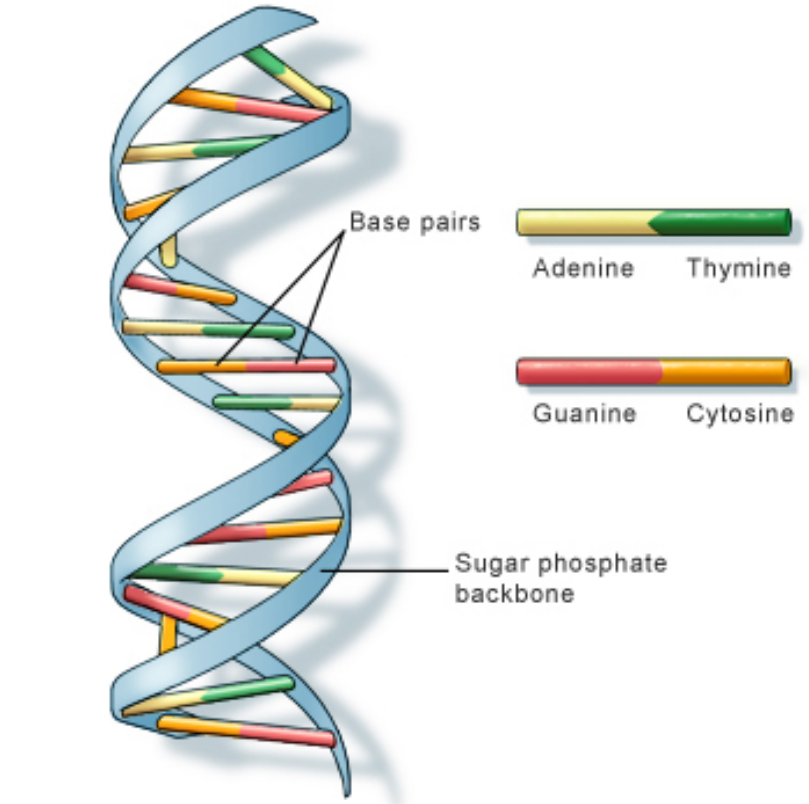


| Subset | Sequence of digits | Binary interpretation | Decimal equivalent |
|---|---|---|---|
| { } | 0, 0, 0 | $000_2$ | $0_{10}$ |
| { a } | 0, 0, 1 | $001_2$ | $1_{10}$ |
| { b } | 0, 1, 0 | $010_2$ | $2_{10}$ |
| { a, b } | 0, 1, 1 | $011_2$ | $3_{10}$ |
| { c } | 1, 0, 0 | $100_2$ | $4_{10}$ |
| { a, c } | 1, 0, 1 | $101_2$ | $5_{10}$ |
| { b, c } | 1, 1, 0 | $110_2$ | $6_{10}$ |
| { a, b, c } | 1, 1, 1 | $111_2$ | $7_{10}$ |

# Another Application

## How many ways to arrange four characters?

We wish to know how many possible *words* we can make from an alphabet of four characters (i.e., a permutation)




U.S. National Library of Medicine

# Another Application

## Use Quadruple Coding! (Only kidding, use the powerset function again)



### powerset stuff

```
d = FiniteSet('a','c','g','t')
dddd = d**4 #Cartesian crossproduct
#{a, c, g, t} x {a, c, g, t}
# x {a, c, g, t} x {a, c, g, t}
len(dddd)
for i in dddd: print(i) #word combinations
```

### Some sample words

(a, a, a, a), (a, a, t, g), (a, a, t, t),
(a, t, t, c), (c, a, a, a), (c, a, t, a),
(c, a, t, c), (g, c, t, a), (c, g, t, a), and etc

# Participation 6

## Search for this repository and push work to it



Place work in:
`cs102-participation-starters/06_part_starter/`
and push it

- In your repository: `mkdir 06_part_starter/`
- Note: Participation checks are given only for **work done while you are in class**.
- Time limit: Push your work by the end of class (12pm) for credit.

- Details on next slide...

# Participation 6

## Explore!



- You are to use the `FiniteSet()` function in a Python program that determines how many telephone numbers may be generated a 7 digit number (i.e., numbers that look like: 555-1234). Then, determine how many numbers are possible when using an area code (i.e., numbers that look like: 814-555-1234).
- Use the interactive interpreter to work with the code or use the python version 2 interpreter on your machine. Save your work in a source file `/06_part_starter/telephone.py`
- Note: Your program is to ask the user how many digits long the program is and then to output a number of possible telephone numbers that can be generated from the length.