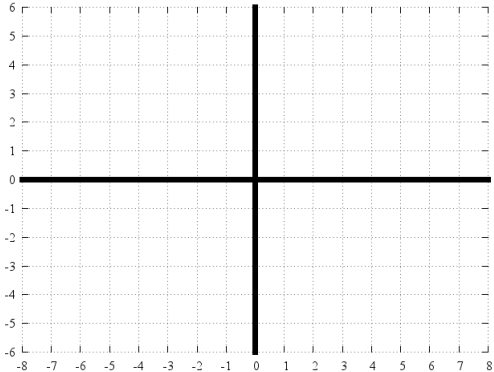
Playing with $\mathbb{C}$: Reflecting about the real/imaginary axis

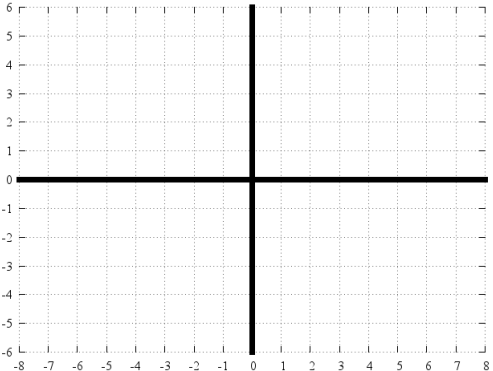
Reflecting about the real axis: Use complex conjugate $z \mapsto \bar{z}$



Playing with $\mathbb{C}$: Multiplying complex numbers by a positive real number

$f(z) = 0.5\,z$

Multiply each complex number by 0.5



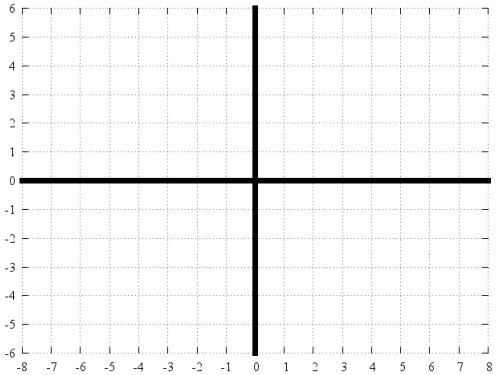
Arrow in same direction but half the length.

Scaling

Playing with $\mathbb{C}$: Multiplying complex numbers by a negative number

$f(z) = (-1)\,z$

Multiply each complex number by -1



Arrow in opposite direction

Rotation by 180 degrees

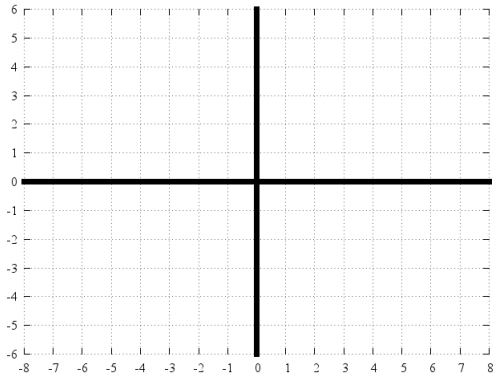
Playing with $\mathbb{C}$: Multiplying by $\mathbf{i}$: rotation by 90 degrees

How to rotate counterclockwise by 90°?

Need $x + y\,\mathbf{i} \mapsto -y + x\,\mathbf{i}$

Use $\mathbf{i}(x + y\mathbf{i}) = x\,\mathbf{i} + y\,\mathbf{i}^2 = x\,\mathbf{i} - y$

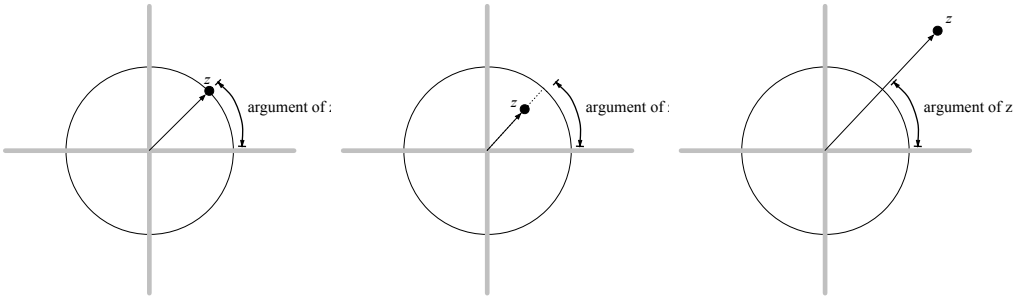
$f(z) = \mathbf{i}\,z$



Playing with $\mathbb{C}$: The unit circle in the complex plane: *argument* and angle

What about rotating by another angle?

Definition: *Argument* of z is the angle in radians between z arrow and $1 + 0\mathbf{i}$ arrow.



Rotating a complex number z means *increasing its argument*.

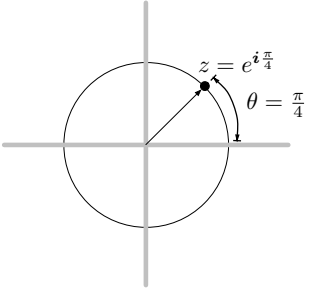
Playing with $\mathbb{C}$: Euler's formula

"He calculated just as men breathe, as eagles sustain themselves in the air."
Said of Leonhard Euler

Euler's formula: For any real number θ,

$$e^{\theta \mathbf{i}}$$

is the point z on the unit circle with argument θ.



$e = 2.718281828...$

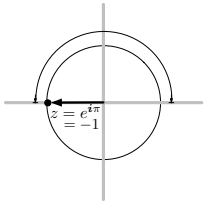
Playing with $\mathbb{C}$: Euler's formula

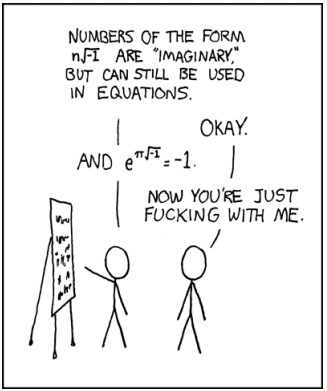
Euler's formula: For any real number θ,

$$e^{\theta \mathbf{i}}$$

is the point z on the unit circle with argument θ.

Plug in $\theta = \pi$....





Playing with $\mathbb{C}$: Euler's formula

Plot

$$e^{0\cdot\frac{2\pi\mathbf{i}}{20}},\ e^{1\cdot\frac{2\pi\mathbf{i}}{20}}, e^{2\cdot\frac{2\pi\mathbf{i}}{20}}, e^{3\cdot\frac{2\pi\mathbf{i}}{20}}, \ldots, e^{19\cdot\frac{2\pi\mathbf{i}}{20}}$$

Playing with $\mathbb{C}$: Rotation by τ radians

Back to question of rotation by any angle τ.

- Every complex number can be written in the form $z = re^{\theta \mathbf{i}}$
  - r is the absolute value of z
  - θ is the argument of z
- Need to increase the argument of z
- Use exponentiation law $e^a \cdot e^b = e^{a+b}$
- $re^{\theta \mathbf{i}} \cdot e^{\tau \mathbf{i}} = re^{\theta \mathbf{i} + \tau \mathbf{i}} = re^{(\theta+\tau)\mathbf{i}}$
- $f(z) = z \cdot e^{\tau \mathbf{i}}$ does rotation by angle τ.

Playing with $\mathbb{C}$: Rotation by τ radians

Rotation by $3\pi/4$

Rotation of Complex Numbers activity

You want a function that rotates by angle $\pi/4$.

1. For what number τ is the function $z \mapsto e^{\tau \mathbf{i}} z$?
2. For input $z = 3e^{(\pi/3)\mathbf{i}}$, what is the output of the rotation function?
3. Draw a diagram of the complex plane showing
   - the circle of radius 1, and
   - the input and output points.

Transformations of complex numbers

- **Translation:** Move a point a specific distance in a specific direction done by adding a specific complex number $z \mapsto z + z_0$
- **Reflection:** Reflect about real axis:take the complex conjugate $z \mapsto \bar{z}$
- **Scaling:** Increase coordinates by a factor Multiply by a positive real number $z \mapsto az$
- **Rotatiion:** Rotate about the origin Multiply by a complex number with absolute value 1 $z \mapsto e^{\theta \mathbf{i}} z$

Probability distribution function

A special kind of function is a *probability distribution function*.
It is used to specify relative likelihood of different outcomes of a single experiment.
It assigns a *probability* (a nonnegative number) to each possible outcome.
The probabilities of all the possible outcomes must sum to 1.

Example:

Probability distribution function for drawing a letter at beginning of the board game Scrabble:

A	9	B	2	C	2	D	4	E	12	F	2	G	3	H	2
I	9	J	1	K	1	L	4	M	2	N	6	O	8	P	2
Q	1	R	6	S	4	T	6	U	4	V	2	W	2	X	1
Y	2	Z	1												

Since the total number of tiles is 98, the probability of drawing an E is 12/98, the probability of drawing an A is 9/98, etc. In Python:

```
{'A':9/98, 'B':2/98, 'C':2/98, 'D':4/98, 'E':12/98, 'F':2/98,
 'G':3/98, 'H':2/98, 'I':9/98, 'J':1/98, 'K':1/98,  'L':1/98,
 'M':2/98, 'N':6/98, 'O':8/98, 'P':2/98, 'Q':1/98,  'R':6/98,
```

Probability distribution function: Uniform distributions

Often the probability distribution is a *uniform distribution*. That means it assigns the same probability to each outcome.

To
model rolling a die, the possible outcomes are 1, 2, 3, 4, 5, and 6, and the probabilities are

$$\Pr(1) = \Pr(2) = \Pr(3) = \Pr(4) = \Pr(5) = \Pr(6) = 1/6 \quad .$$

In Python,

```
>>> Pr = {1:1/6, 2:1/6, 3:1/6, 4:1/6, 5:1/6, 6:1/6}
```

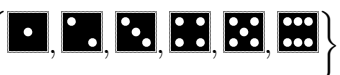
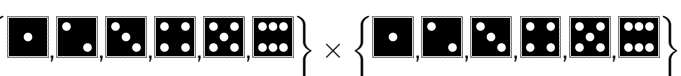
To model the flipping of two coins, the possible outcomes are HH, HT, TH, TT and the probability of all outcomes is $1/4$.
In Python,

```
>>> Pr = {('H', 'H'):1/4, ('H', 'T'):1/4,
          ('T', 'H'):1/4, ('T', 'T'):1/4}
```

Outcome space

The set of all possible outcomes is called the outcome set Ω. .

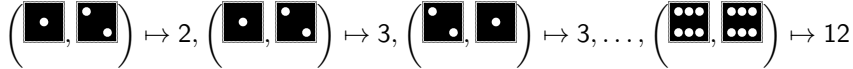
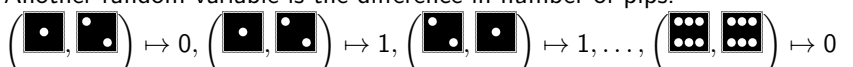
Examples:

- **Drawing a letter in Scrabble:** $\Omega = \{\text{blank}, A, B, C, \ldots, Z\}$
- **Flipping a coin:** $\Omega = \{H, T\}$
- **Flipping two coins:** $\Omega = \{H, T\} \times \{H, T\}$
- **Rolling a die:** $\Omega =$ 
- **Rolling two dice:** $\Omega =$ 

Random variables

A random variable is a function whose domain is the outcome set Ω.

Examples:

- **Drawing a letter in Scrabble:** One random variable is the score per tile: blank $\mapsto 0, A \mapsto 1, B \mapsto 3, C \mapsto 3, D \mapsto 2, E \mapsto 1, D \mapsto 4, \ldots, Z \mapsto 10$
- **Rolling two dice:**
  - One random variable is the total number of pips:
    
    $\mapsto 2, \ldots, \mapsto 12$
  - Another random variable is the difference in number of pips:
    
    $\mapsto 0, \ldots, \mapsto 0$

Probability distribution for a random variable

- If we know probability distribution for the outcome space Ω,
- then we can infer probability distribution for the random variable.

Examples:

- **Number of pips on two dice:** What is the probability of getting 4? Well, how can that happen?
  - Outcomes that result in value being 4: 
  - Each of these outcomes has probability $\frac{1}{36}$.
  - Total probability of value being 4: $\frac{1}{36} + \frac{1}{36} + \frac{1}{36} = \frac{3}{36}$

Playing with $GF(2)$: Encryption

Alice wants to arrange with Bob to communicate one bit p (the *plaintext*).
To ensure privacy, they use a cryptosystem:

- Alice and Bob agree beforehand on a secret key k.
- Alice encrypts the plaintext p using the key k, obtaining the cyphertext c according to the table

p	k	c
0	0	0
0	1	1
1	0	1
1	1	0

Q: Can Bob uniquely decrypt the cyphertext?
A: Yes: for any value of k and any value of c, there is just one consistent value for p.

An eavesdropper, Eve, observes the value of c (but does not know the key k).
Question: Does Eve learn anything about the value of p?
Simple answer: No:

- if $c = 0$, Eve doesn't know if $p = 0$ or $p = 1$ (both are consistent with $c = 0$).
- if $c = 1$, Eve doesn't know if $p = 0$ or $p = 1$ (both are consistent with $c = 1$).

More sophisticated answer: It depends on how the secret key k is chosen.
Suppose k is chosen by flipping a coin:

Probability is $\frac{1}{2}$ that $k = 0$

Playing with $GF(2)$: One-to-one and onto function and perfect secrecy

p	k	c
0	0	0
0	1	1
1	0	1
1	1	0

What is it about this cryptosystem that leads to perfect secrecy? Why does Eve learn nothing from eavesdropping?

Define $f_0 : GF(2) \longrightarrow GF(2)$ by

$f_0(k) =$encryption of $p = 0$ with key k

According to the first two rows of the table,

$$f_0(0) = 0 \text{ and } f_0(1) = 1$$

This function is one-to-one and onto.

When key k is chosen uniformly at random

$$\text{Prob}[k = 0] = \tfrac{1}{2}, \text{Prob}[k = 1] = \tfrac{1}{2}$$

the probability distribution of the output $f_0(k) = p$ is also uniform:

$$\text{Prob}[f_0(k) = 0] = \tfrac{1}{2}, \text{Prob}[f_0(k) = 1] = \tfrac{1}{2}$$

Define $f_1 : GF(2) \longrightarrow GF(2)$ by

$f_1(k) =$encryption of $p = 1$ with key k

According to the last two rows of the table,

$$f_1(0) = 1 \text{ and } f_1(1) = 0$$

This function is one-to-one and onto.

When key k is chosen uniformly at random

$$\text{Prob}[k = 0] = \tfrac{1}{2}, \text{Prob}[k = 1] = \tfrac{1}{2}$$

the probability distribution of the output $f_1(k) = p$ is also uniform:

$$\text{Prob}[f_1(k) = 1] = \tfrac{1}{2}, \text{Prob}[f_1(k) = 0] = \tfrac{1}{2}$$

The probability distribution of the cyphertext does not depend on the plaintext!

Perfect secrecy

Idea is the basis for cryptosystem: the **one-time pad**.

If each bit is encrypted with its own one-bit key, the cryptosystem is unbreakable

p	k	c
0	0	0
0	1	1
1	0	1
1	1	0

In the 1940's the Soviets started re-using bits of key that had already been used.

Unfortunately for them, this was discovered by the US Army's Signal Intelligence Service in the top-secret VENONA project.

This led to a tiny but historically significant portion of the Soviet traffic being cracked, including intelligence on

- spies such as Julius Rosenberg and Donald Maclean, and
- Soviet espionage on US technology including nuclear weapons.

The public only learned of VENONA when it was declassified in 1995.

[2] The Vector

The Vector: William Rowan Hamilton

By age 5, Latin, Greek, and Hebrew
By age 10, twelve languages including Persian, Arabic, Hindustani and Sanskrit.

William Rowan Hamilton, the inventor of the theory of quaternions... and the plaque on Brougham Bridge, Dublin, commemorating Hamilton's act of vandalism.

$$i^2 = j^2 = k^2 = ijk = -1$$

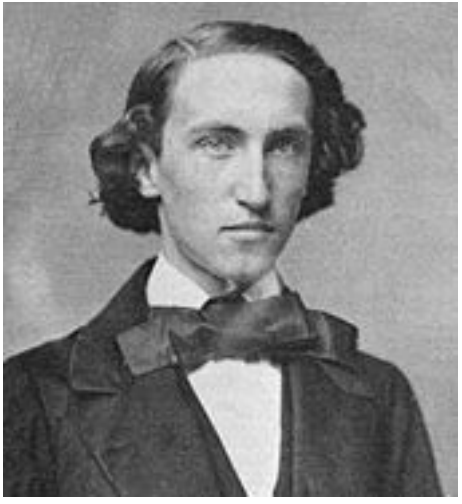
And here there dawned on me the notion that we must admit, in some sense, a fourth dimension of space for the purpose of calculating with triples ... An electric circuit seemed to close, and a spark flashed forth.

The Vector: Josiah Willard Gibbs

Started at Yale at 15
Got Ph.D. at Yale at 24
(1st engineering doctorate in US)
Tutored at Yale
Spent three years in Europe
Returned to be professor at Yale

Developed *vector analysis* as an alternative to quaternions.

His unpublished notes were passed around for twenty years.



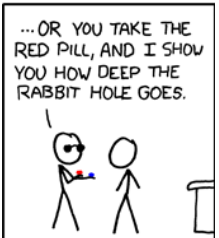
"Professor Willard Gibbs must be ranked as one of the retarders of ... progress in virtue of his pamphlet on *Vector Analysis*; a sort of hermaphrodite monster."
(Peter Guthrie Tait, a partisan of quaternions)

What is a vector?

- This is a 4-vector over $\mathbb{R}$:

$$[3.14159, 2.718281828, -1.0, 2.0]$$

- We will often use Python's lists to represent vectors.
- Set of all 4-vectors over $\mathbb{R}$ is written $\mathbb{R}^4$.
- This notation might remind you of the notation $\mathbb{R}^D$: the set of functions from D to $\mathbb{R}$.



Vectors are functions

Think of our 4-vector $[3.14159, 2.718281828, -1.0, 2.0]$ as the function

$$0 \mapsto 3.14159, \quad 1 \mapsto 2.718281828, \quad 2 \mapsto -1.0, \quad 3 \mapsto 2.0$$

$\mathbb{F}^d$ is notation for set of functions from $\{0, 1, 2, \ldots, d-1\}$ to $\mathbb{F}$.

Example: $GF(2)^5$ is set of 5-element bit sequences, e.g. [0,0,0,0,0], [0,0,0,0,1], ...

Let WORDS = set of all English words
In information retrieval, a document is represented ("bag of words" model) by a function $f : WORDS \longrightarrow \mathbb{R}$ specifying, for each word, how many times it appears in the document.
We would refer to such a function as a WORDS-vector over $\mathbb{R}$

Definition: For a field $\mathbb{F}$ and a set D, a *D-vector over $\mathbb{F}$* is a function from D to $\mathbb{F}$. The set of such functions is written $\mathbb{F}^D$

For example, $\mathbb{R}^{WORDS}$

Representation of vectors using Python dictionaries

We often use Python's dictionaries to represent such functions, e.g.

$$\{0\texttt{:3.14159, }1\texttt{:2.718281828, }2\texttt{:-1.0, }3\texttt{:2.0}\}$$

What about representing a WORDS-vector over $\mathbb{R}$?

For any single document, most words are *not* represented. They should be mapped to zero.

Our convention for representing vectors by dictionaries: we are allowed to omit key-value pairs when value is zero.

Example: "The rain in Spain falls mainly on the plain" would be represented by the dictionary

```
{'on': 1, 'Spain': 1, 'in': 1, 'plain': 1, 'the': 2,
 'mainly': 1, 'rain': 1, 'falls': 1}
```

Sparsity

A vector most of whose values are zero is called a *sparse* vector.
If no more than k of the entries are nonzero, we say the vector is *k-sparse*.

A k-sparse vector can be represented using space proportional to k.

Example: when we represent a corpus of documents by WORD-vectors, the storage required is proportional to the total number of words in all documents.

Most signals acquired via physical sensors (images, sound, ...) are not exactly sparse.

Later we study *lossy compression*: making them sparse while preserving perceptual similarity.

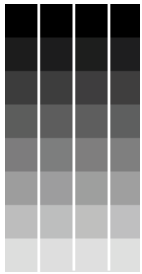
What can we represent with a vector?

- Document (for information retrieval)
- Binary string (for cryptography/information theory)
- Collection of attributes
  - Senate voting record
  - demographic record of a consumer
  - characteristics of cancer cells
- State of a system
  - Population distribution in the world
  - number of copies of a virus in a computer network
  - state of a pseudorandom generator
  - state of *Lights Out*
- Probability distribution, e.g. {`1:1/6, 2:1/6, 3:1/6, 4:1/6, 5:1/6, 6:1/6`}

What can we represent with a vector?

- Image

```
{(0,0): 0,    (0,1): 0,    (0,2): 0,    (0,3): 0,
 (1,0): 32,   (1,1): 32,   (1,2): 32,   (1,3): 32,
 (2,0): 64,   (2,1): 64,   (2,2): 64,   (2,3): 64,
 (3,0): 96,   (3,1): 96,   (3,2): 96,   (3,3): 96,
 (4,0): 128,  (4,1): 128,  (4,2): 128,  (4,3): 128,
 (5,0): 160,  (5,1): 160,  (5,2): 160,  (5,3): 160,
 (6,0): 192,  (6,1): 192,  (6,2): 192,  (6,3): 192,
 (7,0): 224,  (7,1): 224,  (7,2): 224,  (7,3): 224 }
```



What can we represent with a vector?

- Points
  - Can interpret the 2-vector $[x, y]$ as a point in the plane.

  - Can interpret 3-vectors as points in space, and so on.

Representing points

- List of 2-element lists:

```
>>> L = [[2, 2], [3, 2], [1.75, 1], [2, 1], [2.25, 1], [2.5, 1],
 [2.75, 1], [3, 1], [3.25, 1]]
```

- Use the `plot` module to plot these 2-vectors.

```
>>> plot(L, 5)
```

Vector addition: Translation and vector addition

With complex numbers, translation achieved by adding a complex number, e.g. $f(z) = z + (1 + 2\mathbf{i})$

Let's do the same thing with vectors...

Definition of vector addition:

$$[u_1, u_2, \ldots, u_n] + [v_1, v_2, \ldots, v_n] = [u_1 + v_1, u_2 + v_2, \ldots, u_n + v_n]$$

For 2-vectors represented in Python as 2-element lists, addition procedure is

```
def add2(v,w): return [v[0]+w[0], v[1]+w[1]]
```

Vector addition: Translation and vector addition

Quiz: Suppose we represent n-vectors by n-element lists. Write a procedure

`addn(v, w)`

to compute the sum of two vectors so represented.

Answer:

```
def addn(v, w): return [v[i]+w[i] for i in range(len(v))]
```