Basic Algebra

Functions, polynomials, coordinate systems, complex numbers

Yordan Darakchiev

Technical Trainer

Software University

SoftUni

Software University

https://softuni.bg

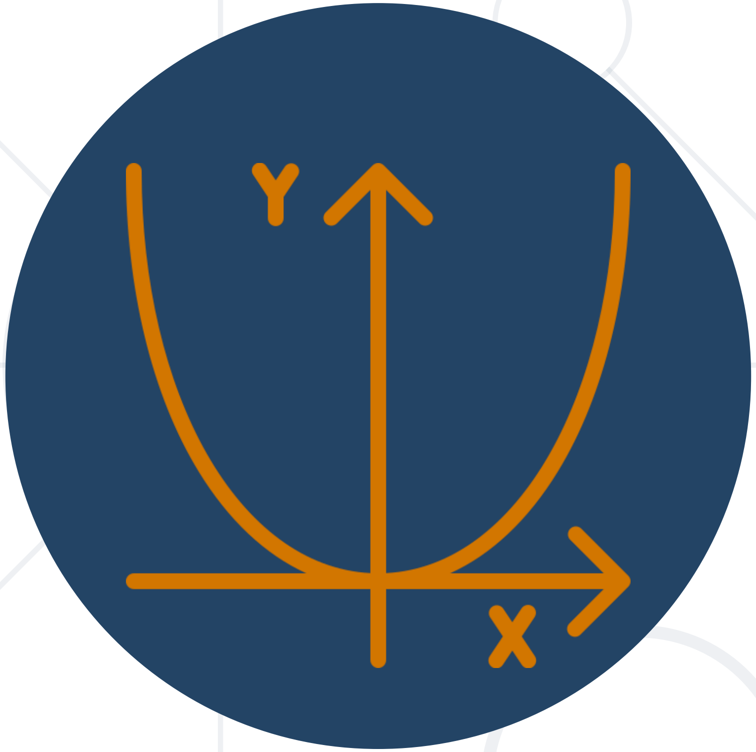
Have a Question?

Software University

sli.do

#MathForDevs

Table of Contents



Polynomials

Polynomials

- We already looked at linear and quadratic polynomials
- **Term** (monomial): $2x^2$
  - Coefficient (number), variable, power (number ≥ 0)
- **Polynomial**: sum of monomials
  - $2x^4 + 3x^2 - 0{,}5x + 2{,}72$
  - **Degree**: the highest degree of the variable (with coefficient $\neq 0$)

Operations on Polynomials

- Defined the same way as with numbers
- Addition and subtraction
  - $(2x^2 + 5x - 8) + (3x^4 - 2) = 3x^4 + 2x^2 + 5x - 10$
- Multiplication and division
  - $(2x^2 + 5x - 8)(3x^4 - 2) =$

    $= 6x^6 + 15x^5 - 24x^4 - 4x^2 - 10x + 16$

Polynomials in Python

- numpy has a module for working with polynomials
  - Includes the "general" polynomials,
    as well as a few special cases
    - Chebyshev, Legandre, Hermit
- Storing polynomials
  - As a list
  - There are two ways, depending on which function we use
    - Preferred way: **index = power, value = coefficient**

Polynomials in Python (2)

- Preferred way

```
from numpy.polynomial import Polynomial

print(Polynomial([1, 2, 3]))  # Represents 1 + 2x + 3x^2
p1 = Polynomial([-8, 5, 2])
p2 = Polynomial([-2, 0, 0, 0, 3])
print(p1 + p2)
# -10.0 + 5.0 x + 2.0 x**2 + 0.0 x**3 + 3.0 x**4
print(p1 * p2)
# 16.0 - 10.0 x - 4.0 x**2 + 0.0 x**3 - 24.0 x**4 + 15.0 x**5 + 6.0 x**6
```

- Old way (still supported)

```
print(np.poly1d([1, 2, 3])) # Represents x^2 + 2x + 3
print(np.polyadd([2, 5, -8], [3, 0, 0, 0, -2])) # [  3   0   2   5 -10]
print(np.polymul([2, 5, -8], [3, 0, 0, 0, -2])) # [  6  15 -24   0  -4 -10  16]
```

Sets

Set notation and basic operations

Set

- An **unordered collection** of things
  - Usually, numbers
  - No repetitions
- Set notation: $\{x \in \mathbb{R} \mid x \geq 0\}$
  - "The set of numbers x, which are a subset of the real numbers, which are greater than or equal to zero"
  - Left: example element
  - Right: conditions to satisfy

Sets in Python

- Direct creation
  - Set notation: curly braces

```python
names = {"Alice", "Bob", "Charlie", "David", "Bob"}
# {'Charlie', 'Alice', 'Bob', 'David’}
```

- Set comprehension
  - Very similar to the math notation
  - Like list comprehensions

```python
positive_x = {x for x in range(-5, 5) if x >= 0}
# {0, 1, 2, 3, 4}
```

Operations on Sets

- Cardinality (number of elements) $|S|$
- Checking whether an element is in the set $x \in S$
- Checking whether a set is subset of another set $S_1 \subseteq S_2$
- Union $S_1 \cup S_2$, intersection $S_1 \cap S_2$, difference $S_1 \setminus S_2$

```
set1 = {1, 2, 3, 4}
set2 = {3, 4, 5, 10, 3, 5, 10, 3, 3}
print(len(set2))  # 4
print(1 in set1)  # True
print(10 not in set1)  # True
print({1, 2}.issubset(set1))  # True
print(set1.union(set2))  # {1, 2, 3, 4, 5, 10}
print(set1.difference(set2))  # {1, 2}
print(set2.difference(set1))  # {10, 5}
print(set1.symmetric_difference(set2))  # {1, 2, 5, 10}
```

Operations on Sets (2)

- De Morgan's laws
  - Provide a link between logic and set theory
- Cartesian product: $A \times B = \{(a, b) \mid a \in A, b \in B\}$
  - Note that the elements of the new set are **tuples**

```
set1 = {1, 2, 3}
set2 = {-1, 0, 1, 2}

print({(a, b) for a in set1 for b in set2})
# {(2, -1), (1, 2), (3, -1), ...}
```

  - Commonly used to denote pairs of numbers (coordinates)

$$\mathbb{R} \times \mathbb{R} \equiv \mathbb{R}^2 = \{(x, y) \mid x \in \mathbb{R}, y \in \mathbb{R}\}$$

Functions

Mappings from one thing to another

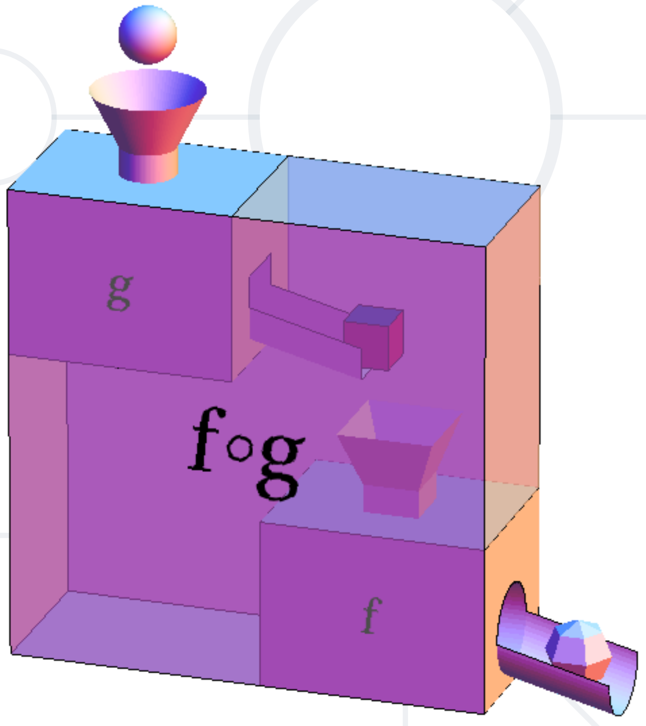
Function

- A relation between
  - A set of inputs X (**domain**)
  - … and a set of outputs Y (**codomain**)
- **One input produces exactly one output**
- The inputs don't need to be numbers
- Functions don't know how to compute the output, they're just mappings
  - In programming, we write **procedures**
- Math notation: $f : X \to Y$
  - Commonly abbreviated as $y = f(x)$

Function Composition

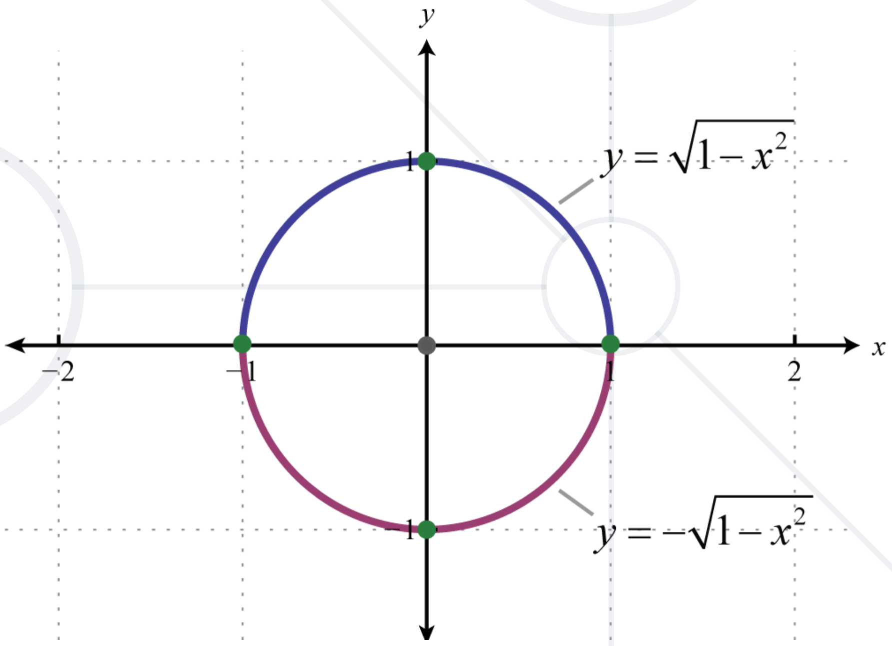
- Also called pipelining in most languages
- Takes two functions and applies them in order
  - **Innermost to outermost**
  - Math notation: $f \circ g = f(g(x))$
  - Can be generalized to more functions
- Note that the order matters

$$f(x) = 2x + 3, \;\; g(x) = x^2$$
$$(f \circ g)(x) = f(g(x)) = f(x^2) = 2x^2 + 3$$
$$(g \circ f)(x) = g(f(x)) = g(2x + 3) = (2x + 3)^2$$



Challenge: Graphing a circle

- Let's try to graph the unit circle
  - Equation: $x^2 + y^2 = 1$
- This cannot be represented as one function
  - There are multiple values of y (e.g., for $x = 0{,}5$)
  - But we want to represent the circle as one object
  - First try: draw two separate half-circles

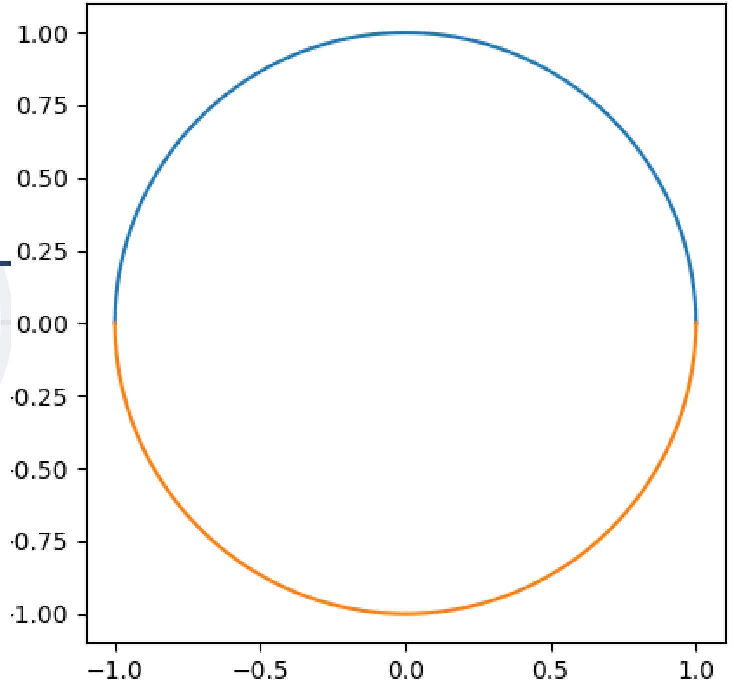


Graphing a Circle: First Try

Software University

```
def plot_function(f, x_min = -10, x_max = 10, n_values = 2000):
  plt.gca().set_aspect("equal")
  x = np.linspace(x_min, x_max, n_values)
  y = f(x)
  plt.plot(x, y)

plot_function(lambda x: np.sqrt(1 - x**2), -1, 1)
plot_function(lambda x: -np.sqrt(1 - x**2), -1, 1)
plt.show()
```

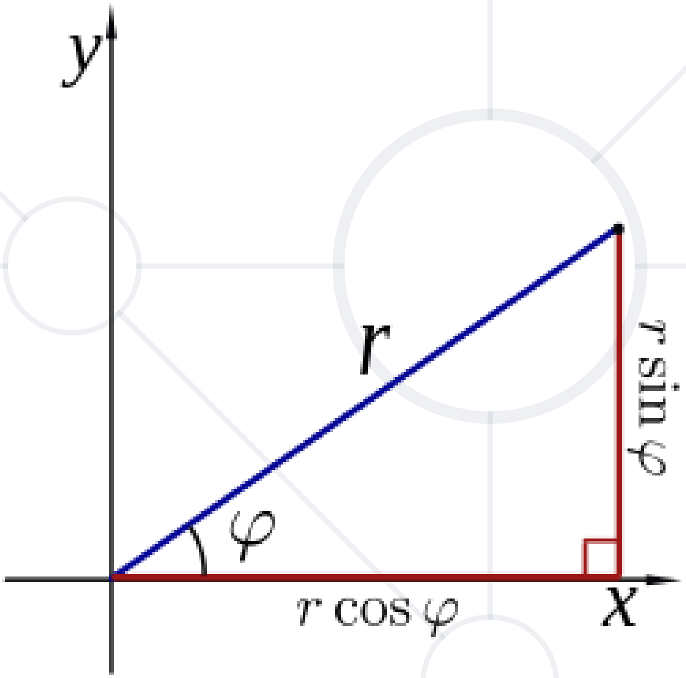


- This works but looks rather ugly and feels like a cheat

Graphing a Circle: Second Try

- Let's change our viewpoint
- Polar coordinates (r, φ)

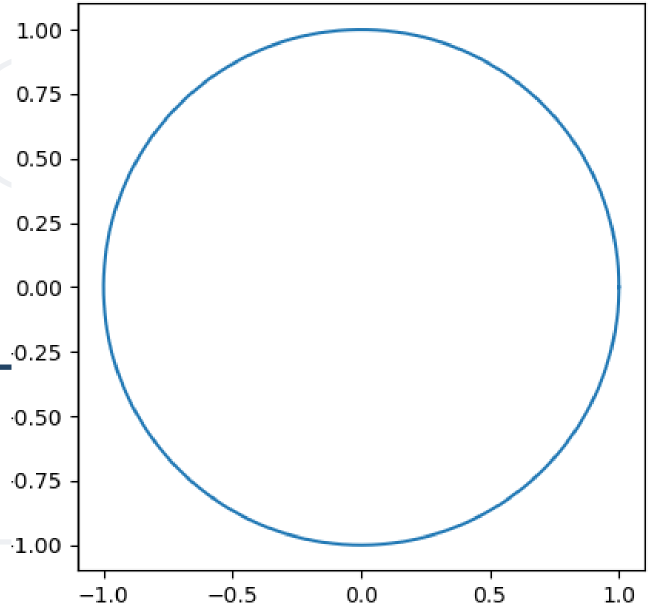
$$x^2 + y^2 = 1$$
$$(r\cos\varphi)^2 + (r\sin\varphi)^2 = 1$$
$$r^2\cos^2\varphi + r^2\sin^2\varphi = 1$$
$$r^2(\cos^2\varphi + \sin^2\varphi) = 1$$
$$r^2 = 1,\ r \geq 0 \Rightarrow r = 1$$



- The equation became very, very simple
  - Doesn't even depend on φ

Graphing a Circle: Second Try

```python
r = 1 # Radius
phi = np.linspace(0, 2 * np.pi, 1000) # Angle (full circle)
x = r * np.cos(phi)
y = r * np.sin(phi)
plt.plot(x, y)
plt.gca().set_aspect("equal")
plt.show()
```



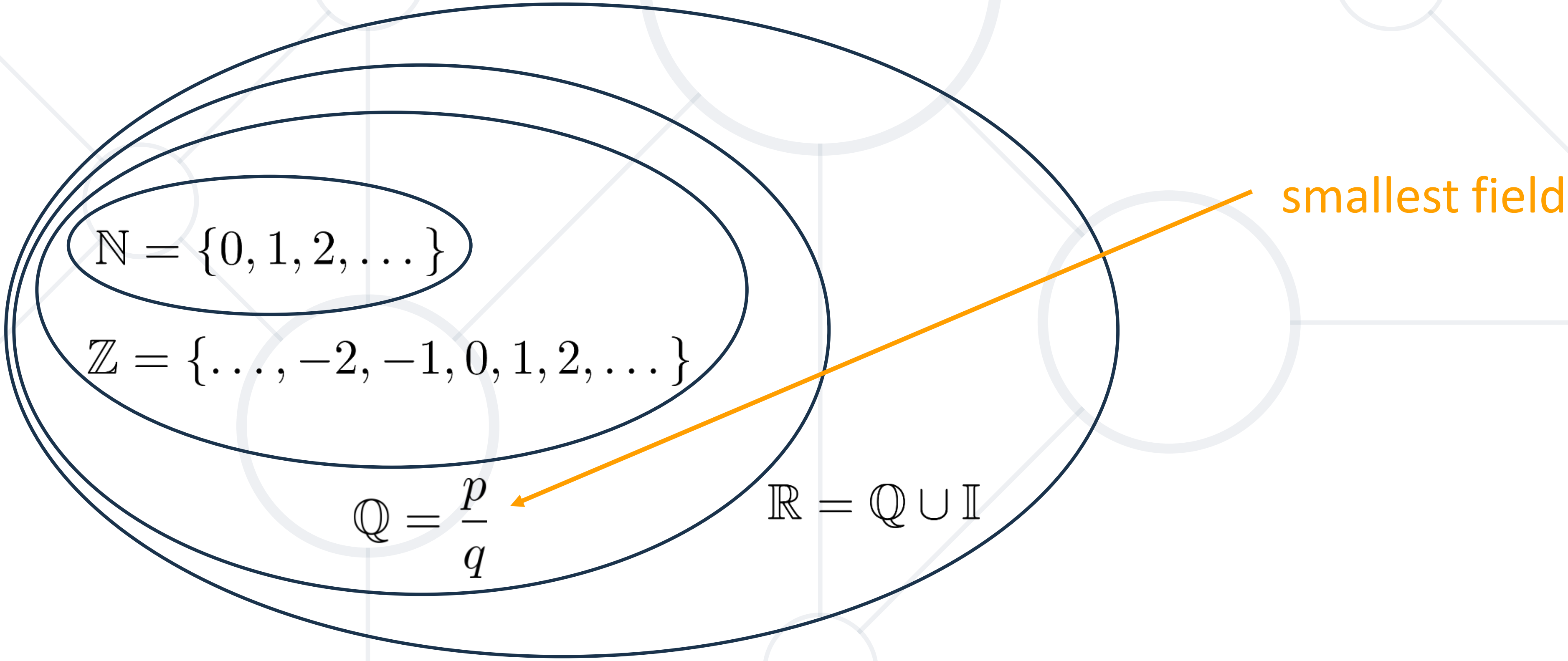
- This introduced parametric curves
  - Given by $x(r, \varphi),\ y(r, \varphi)$ rather than $y(x)$
  - *What other shapes could we represent like this?*

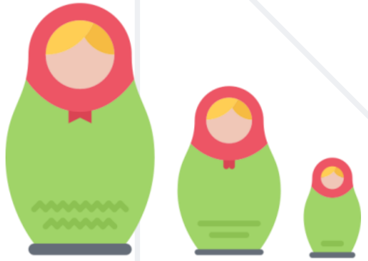
Complex Numbers

Not as complex as they seem

Number Fields

- Field
  - A collection of values with operations "plus" and "times"
  - Algebra is so abstract we can redefine these operations (stay tuned)





Complex Numbers

- Pairs of real numbers: $(a, b) : a, b \in \mathbb{R}$
  - Also $a + bi$
  - "imaginary unit" i: the positive solution of $x^2 = -1$

    $$\operatorname{Re}(a + bi) = a$$

    $$\operatorname{Im}(a + bi) = b$$
- In Python: `j` instead of `i`
-
```
z = 3 + 2j
print(z) # (3+2j)
print(z.real) # 3
print(z.imag) # 2
```
- Addition and multiplication

```
print((3 + 2j) + (8 - 3j)) # (11-1j)
print((3 + 2j) * (8 - 3j)) # (30+7j)
```

Geometry of Complex Numbers

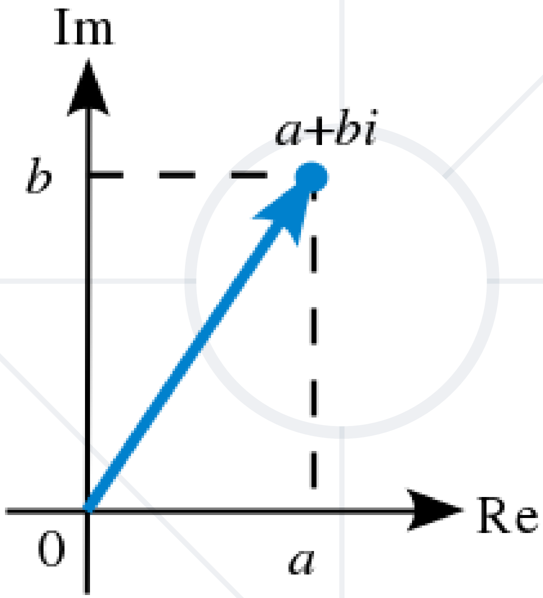
- Tuple ⇒ 2D points

$$z = a + bi; (a, b) \in \mathbb{R}^2$$

- We could also apply the polar transformation

$$r = |z| = \sqrt{a^2 + b^2} \qquad a = r\cos\varphi$$

$$\tan\varphi = \frac{b}{a} \qquad b = r\sin\varphi$$



- Polar form
  - $|z|$ – modulus, φ – argument
  - $z = a + bi = r(\cos\varphi + i\sin\varphi)$

Fundamental Theorem of Algebra

Putting it all together

Fundamental Theorem of Algebra

- Every non-zero, single-variable, degree-n polynomial with complex coefficients has, counted with multiplicity, exactly n complex roots
- Every algebraic equation has as many roots as its power

```
coefs = [
    [-4, -3, 1], # [-1.  4.]
    [-4, 0, 1],  # [-2.  2.]
    [1, 2, 1],   # [-1.00000001 -0.99999999]
    [5, 4, 1]    # [-2.-1.j -2.+1.j]
]

for c in coefs:
  print(Polynomial(c).roots())
```

Some Thoughts on Abstraction

Taking abstraction to the `max()`

Galois Field

- Since algebra is abstract, we can define our own fields
- Galois field: $GF(2)$
  - Elements $\{0, 1\}$
    - There are also extensions with more elements
  - Addition: equivalent to XOR
  - Multiplication: as usual
- Usage: in cryptography

+	0	1
0	0	1
1	1	0

*	0	1
0	0	0
1	0	1

Vectors

- **Vector**: a line segment with a direction
- We saw that 2D vectors and 2D points have a **one-to-one correspondence**
  - Point ⇔ radius-vector ⇔ pair of coordinates
- Can we think of a vector as a mapping?
  - $[2, 3, 5] \Leftrightarrow 0 \rightarrow 2, 1 \rightarrow 3,\ \ 2 \rightarrow -5$
- What does this mean?
  - … we'll find out next time
- What does this imply about fields?

Summary

- Polynomials
  - Single variable, coefficients, powers
- Sets
  - Elements, properties
- Functions
  - Functions in math and programming, function composition
- Coordinates
  - Coordinate systems as a "viewpoint"
- Complex numbers
- Fundamental theorem of algebra

Software University

Questions?

SoftUni

- Software University
- SoftUni Creative
- SoftUni Digital
- SoftUni Foundation
- SoftUni Kids
- СофтУни БУДИТЕЛ
- Finance Academy

SoftUni Diamond Partners

SmartIT

Coca-Cola HBC Bulgaria

SUPER HOSTING .BG

createX

Postbank

top EMPLOYER България Bulgaria 2024

FOR A BETTER WORLD OF WORK

DRAFT KINGS

THE CROWN IS YOURS

VIVACOM

INDEAVR

Serving the high achievers

AMBITIONED

PHAR VISION

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity

License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://about.softuni.bg/
- © Software University – https://softuni.bg