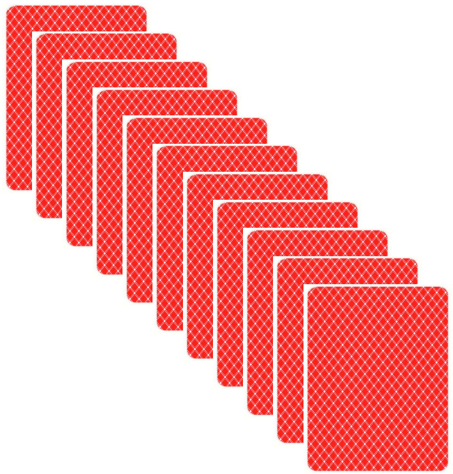
15-110 PRINCIPLES OF COMPUTING – S21

LECTURE 4:
ABSTRACTION

TEACHER:
GIANNI A. DI CARO

جامعة كارنيجي ميلون في قطر
Carnegie Mellon University Qatar

The *sorting* problem was cumbersome to solve just using *words*



- You are given a set of cards (covered) as show in the figure
- Cards are uniquely numbered from 1 to 100, but cards aren’t necessarily placed in the 1-100 order!

➢ You must **sort** the cards in the 1 → 100 order

1. Pick up first card from deck
2. Add the card to sorted pile
3. Pick up first card from deck
4. If card value greater than top card on sorted pile
    1. Then add card on top of sorted pile
5. Instead, if card value is lower that bottom card on sorted pile
    1. Then add card to the bottom of sorted pile
6. If neither 4 or 5 conditions are satisfied, *insert* card in sorted pile
7. Repeat 3-6 until no cards in card deck

Insert:

1. If distance from bottom is less than distance from top, start from bottom
2. Otherwise, start from top
3. Inspect the first two cards from start position
4. If card value is lower than first and higher than second, insert card after first
5. Otherwise, set first card as new start position
6. Repeat 3-5

Let's start moving from natural language to formal language (math)

Can we use less English and more math-like formalism?

$$y = 5$$
$$x = y + 1$$

- **Variables**

$$x = (1, 3, 5, 7, 11, 13, 17)$$
$$x_0 = 1, x_3 = 7$$

- **Indices, lists/vectors**

$$x = \sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

- **Parametric functions**

Variables

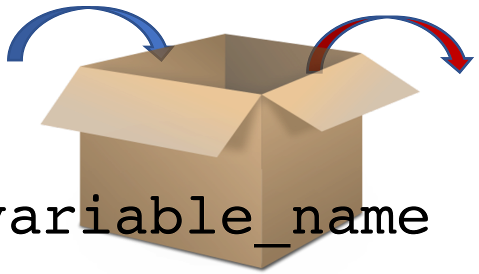
- In math we commonly use **named** parameters and variables to refer to symbols that will take values that we don't know yet

$$y = ax^2 + bx + c$$

$$x = -b \pm \frac{\sqrt{b^2 - 4ac}}{2a}$$

- **Variables:** provide a way to name *information and* access and modify the information by using the name

- A named *container* of information



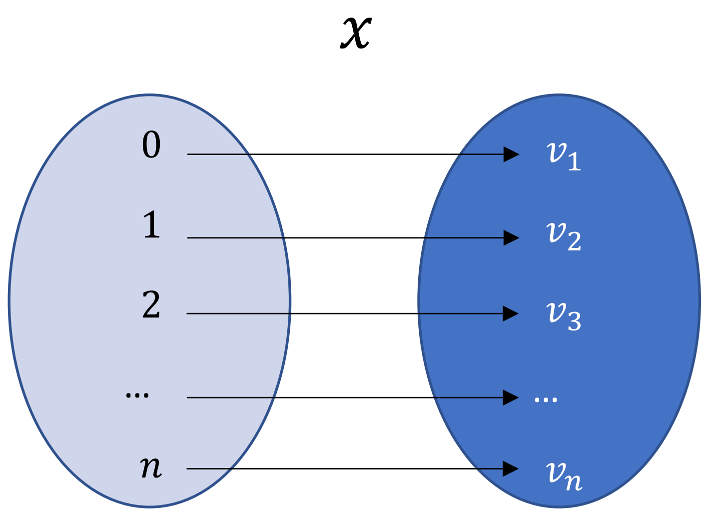
➢ What can we do with a variable (e.g., x)?

✓ **Assign** its value $x = 2$

✓ **Read / use** its value $y = x + 2$

✓ **Modify** its value $x = 4.5$

Indices / vectors / lists



$$x = (1, 3, 5, 7, 11, 13, 17)$$
$$x_0 \to 1,\ x_3 \to 7$$

$$x_4 = -2$$

- ✓ We have established a **1-1 correspondence** between indices (sequence of integers) and values of the variable

- ✓ The variable x is a **vector**, a variable holding a **list of values** (*multi-variate* is the proper mathematical term).
- ➢ Indices give us the freedom to flexibly **access/address the different values**

Parametric functions

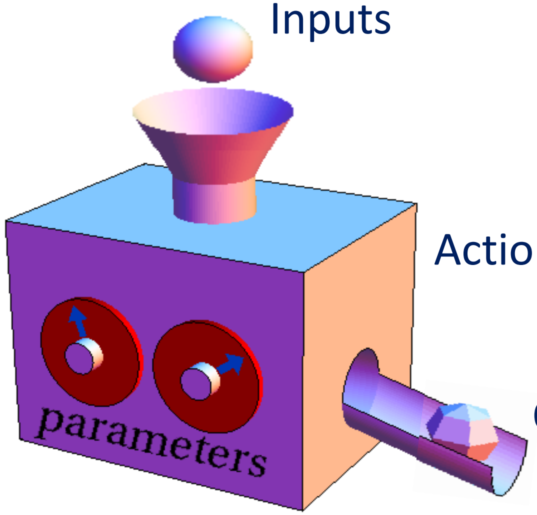
$$\sum_{i=1}^{n} i$$

- Sum of the first n integers
- n is a **parameter**, can take any integer value!

Let's give it a **name** (a *function variable*):

$$sum\_first\_n(n)$$

Name / Label



- We have packed in the name sum_first_n a **procedure** that does something specific
- We have made it **parametric**: it will work for different inputs n

Abstraction

- Define variables
- Use indices to handle multi-valued variables
- Pack procedures / strategies and make then parametric

Examples of **abstraction** from the specific instance / problems

Using abstraction:

✓ We can be more **compact and general** writing algorithms

✓ We can **reuse** things / procedures for **different inputs**

✓ We can **reuse** things /procedures for **different problems requiring the same strategy**

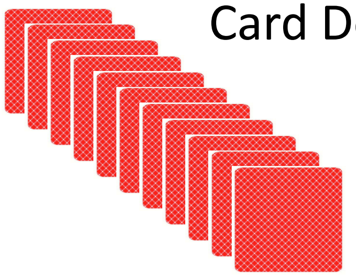
Let’s rewrite the algorithm for the simple search for *max* value

Our solution from previous lectures

1. Pick up the first card from the *deck* pile (n cards)
2. Record down the number v and remove the card from the deck (put it in *done* pile)
3. Assign the number v to max value

4. Pick up the next card from the deck
5. Look at the number, v, and remove the card from the deck
6. If the number is higher that current max value: max value becomes v

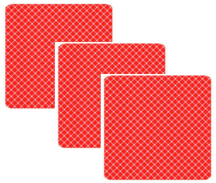
7. Repeat 4-6 $n-1$ times (i.e., until no more cards in deck)
8. Output max value
9. Stop



Card Deck

Where / how can we use variables and indices?

Max value: YY



Done pile

Variables and indices at work: increasing the index variable

1. Pick up the first card from the *deck* pile (n cards)
2. Record down the number v and remove the card from the deck (put it in *done* pile)
3. Assign the number v to max value

4. Pick up the next card from the deck
5. Look at the number, v, and remove the card from the deck
6. If the number is higher that current max value: max value becomes v

7. Repeat 4-6 $n-1$ times (i.e., until no more cards in deck)
8. Output max value
9. Stop

1. Define a **variable** to hold the number of cards: n, e.g., $n = 52$
2. Label the cards values with a set of **indices**: $c_0, c_1, c_2, \cdots, c_{n-1}$
3. Define a **variable** max to hold the best value so far
4. $max = c_{n-1}$
5. Card **index variable**, initialized to 0: $i = 0$
6. Check if $i < n$:
7. if yes: Check if $c_{i+1} > max$
8. if yes: $max = c_{i+1}$; $i = i + 1$; go back to step 6
9. if no: $i = i + 1$; go back to step 6
10. If no: highest card is max

What is the value of i at step 10?

Variables and indices at work: decreasing the index variable

1. Define a **variable** to hold the number of cards: n, e.g., $n = 52$
2. Label the cards values with a set of **indices**: $c_0, \mathrm{c}_1, \mathrm{c}_2, \cdots, c_{n-1}$
3. Define a **variable** max to hold the best value so far
4. $max = c_{n-1}$
5. Card **index variable**, initialized to 0: $i = n - 1$
6. Check if $i > 0$:
7. if yes: Check if $c_{i-1} > max$
8. if yes: $max = c_{i-1}$; $i = i - 1$; go back to step 6
9. if no: $i = i - 1$; go back to step 6
10. If no: highest card is max

Could we start form n?
What else should we modify in that case?

Variables and indices at work: using a *Repeat for* directive

1. Define a **variable** to hold the number of cards: n, e.g., $n = 52$
2. Label the cards values with a set of **indices**: $c_0, c_1, c_2, \cdots, c_{n-1}$
3. Define a **variable** max to hold the best value so far
4. $max = c_{n-1}$
5. **Repeat for** $i = 0, 1, \cdots, n-1$
6. Check if $c_i > max$:
7. if yes: $max = c_i$;
8. highest card is max

What the **Repeat for** directive does?

Same problem: Power of abstraction!

Muffin:	5 QAR	500
Croissant:	7 QAR	450
Chips:	10 QAR	700
Hamburger:	8 QAR	800
Chocolate:	2 QAR	300
Fruit salad:	6 QAR	200

Choose the snack with the lowest calories (preference).
You only have 9 QAR (budget constraint)

Sceptre:	4kg	10$
Shoes:	1kg	1$
Helmet	1kg	2$
Armour:	12kg	4$
Dagger:	2kg	2$

Choose your item in a game! Your want the most valuable item (preference). You can only carry up to 7 kg (weight constraint).

Same abstract problem:

- Choose one item among n possible choices
- Each item uses resources r (money, weight) and has a quality q (calories, worthiness)
- You only have R, limited resources available (e.g., 9 QAR, 7 kg)
- You aim to choose the item with the best quality while respecting the limitation in resources