Data Processing with Python

Lecturer: Nguyen Van Phi
Email: phinv@vnu.edu.vn

Institute for Artificial Intelligence, University of Engineering and Technology, VNU Hanoi

TODAY

- Course info
- Why data science?
- What is computation?
- Python basics

COURSE INFO

Grading

approx. 10%	Quiz/ Attendance
approx. 60%	Final - project
approx. 40%	Midterm/ Lab lecture

no deadline extensions

no copy code, submission from other students or online resource

no solution from AI (eg. ChatGPT)

COURSE OBJECTIVE

1. Basic programming with Python

1. Basic Pipeline of data processing

1. Tools and techniques to process data using Python

Why become an AI and Data Science expert?

But if you decide to do it...

1. It’s a lot of fun!
2. You will be at the cutting edge of research and product
3. You will make lots of money doing something you will enjoy.
4. It’s not that hard to start and do!

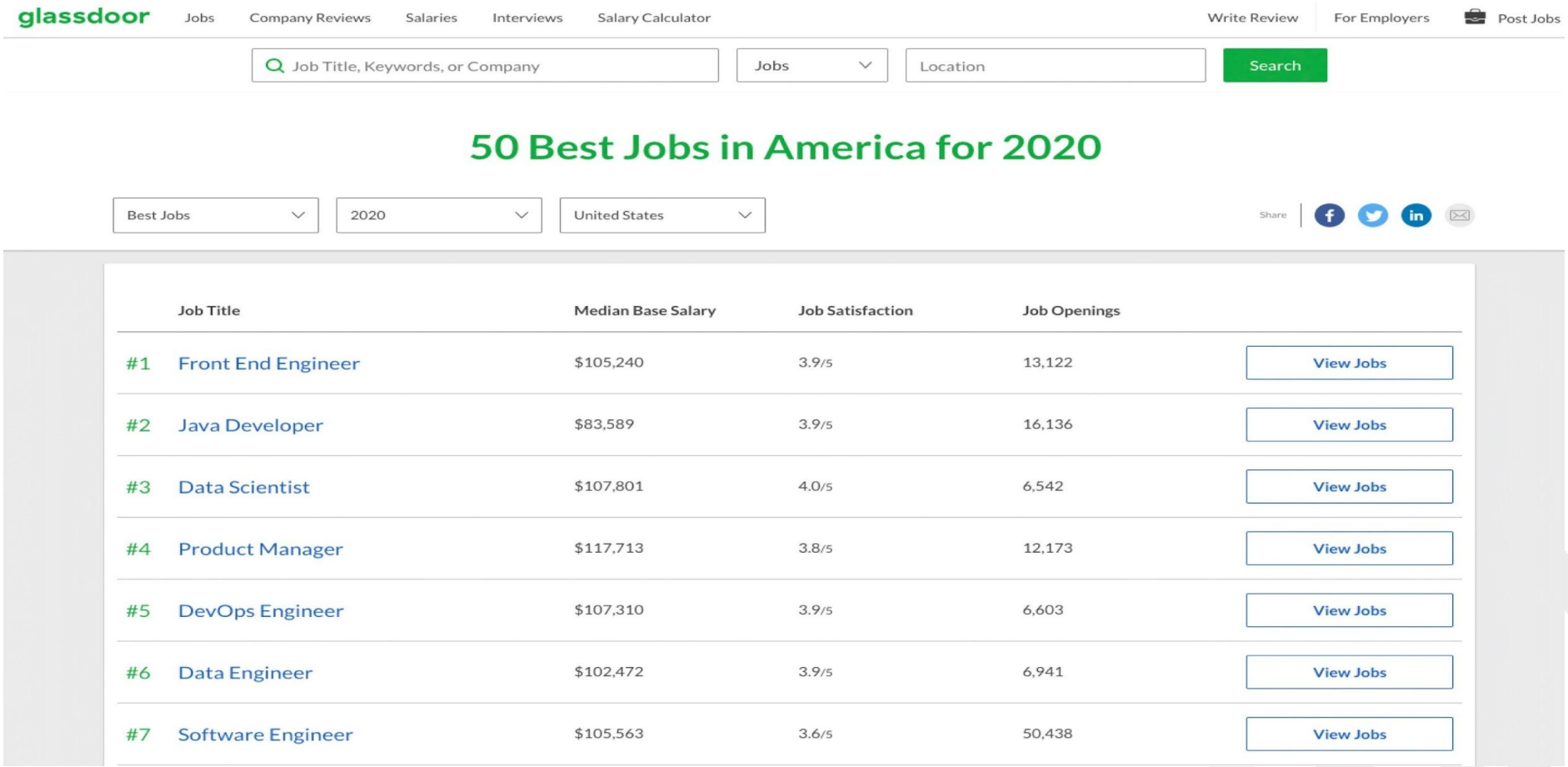
Unsupervised Image-to-Image Translation

Day to night

(Liu et al., 2017)

(Goodfellow 2019)

facebook research


glassdoor
Jobs
Company Reviews
Salaries
Interviews
Salary Calculator
Write Review
For Employers
Post Jobs
Job Title, Keywords, or Company
Jobs
Location
Search
50 Best Jobs in America for 2020
Best Jobs
2020
United States
Share
Job Title
Median Base Salary
Job Satisfaction
Job Openings
#1 Front End Engineer $105,240 3.9/5 13,122 View Jobs
#2 Java Developer $83,589 3.9/5 16,136 View Jobs
#3 Data Scientist $107,801 4.0/5 6,542 View Jobs
#4 Product Manager $117,713 3.8/5 12,173 View Jobs
#5 DevOps Engineer $107,310 3.9/5 6,603 View Jobs
#6 Data Engineer $102,472 3.9/5 6,941 View Jobs
#7 Software Engineer $105,563 3.6/5 50,438 View Jobs

50 Best Jobs in America

Awards

- Best Places to Work
- Highest Rated CEOs
- Best Places to Interview

Lists

- Best Jobs
- Best Cities for Jobs
- Highest Paying Jobs
- Oddball Interview Questions

Trends

- Overview

This report ranks jobs according to each job's Glassdoor Job Score, determined by combining three factors: number of job openings, salary, and overall job satisfaction rating.

Employers: Want to recruit better in 2017? Find out how.

United States | 2017

12k Shares

1 Data Scientist

4.8 / 5
Job Score

4.4 / 5
Job Satisfaction

$110,000
Median Base Salary

4,184
Job Openings

View Jobs

2 DevOps Engineer

History

Long time ago (thousands of years) science was only empirical and people counted stars

History

Long time ago (thousands of years) science was only empirical and people counted stars and used the data to create machines to describe the phenomena

History

Few hundred years ago: theoretical approaches, try to derive equations to describe general phenomena.

$$F = G\frac{m_1 m_2}{d^2}$$

$$i\hbar\frac{\partial}{\partial t} - \Psi = \hat{H}\Psi$$

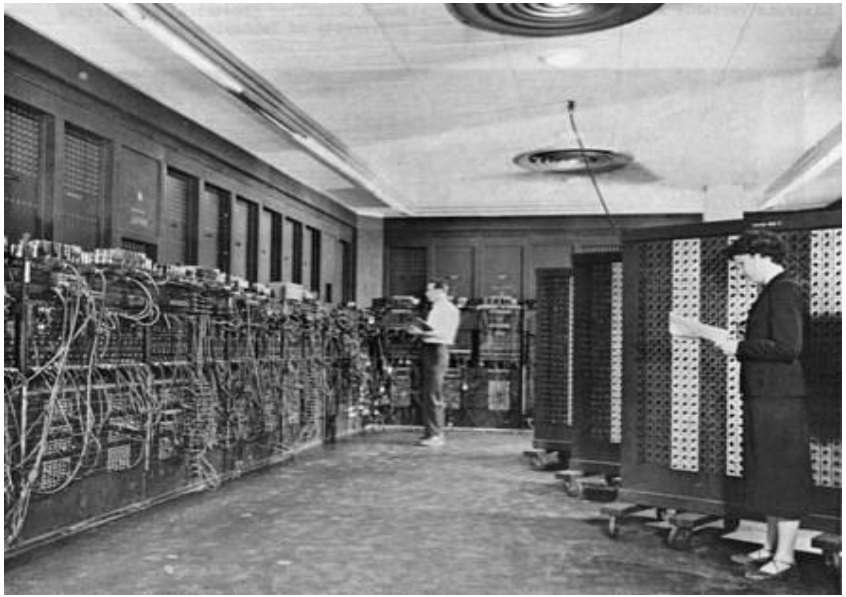
$$\nabla \cdot E = 0 \quad \nabla \times E = -\frac{1}{c}\frac{\partial H}{\partial t}$$
$$\nabla \cdot H = 0 \quad \nabla \times H = \frac{1}{c}\frac{\partial E}{\partial t}$$

$$E = mc^2$$

$$\rho\left(\frac{\partial v}{\partial t} + v \cdot \nabla v\right) = -\nabla p + \nabla \cdot T + f$$

History

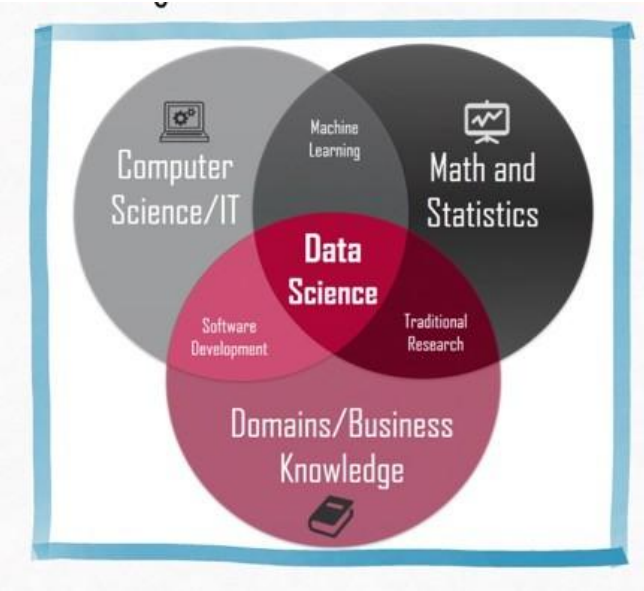
About a hundred years ago:
computational approaches appeared



History

And then it was data science

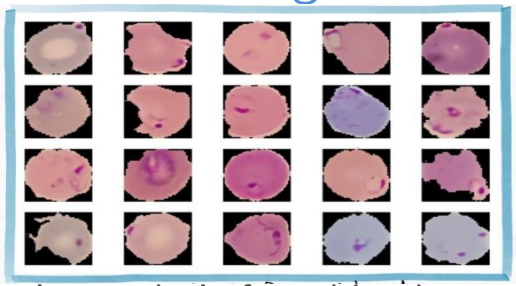
In both data science and machine learning we extract pattern and insights from data.



- Inter-disciplinary
- Data and task focused
- Resource aware
- Adaptable to changes in the environment and needs

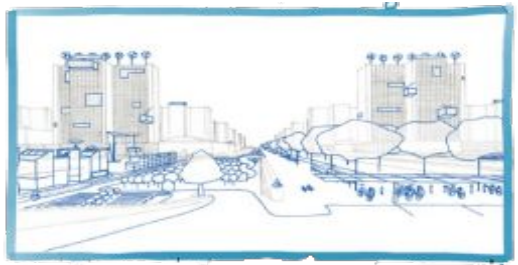
The Potential of Data Science

Disease Diagnosis



Detecting malaria from blood smears

Urban Planning



Predicting and planning for resource needs

Drug Discovery

Quickly discovering new drugs for COVID

Agriculture

Precision agriculture

Ask an interesting question

Get the Data

Explore the Data

Model the Data

Communicate/Visualize the Results

Ask an interesting question

Get the Data

Explore the Data

Model the Data

Communicate/Visualize the Results

What is the scientific goal?

What would you do if you had **all** of the data?
What do you want to predict or estimate?

Ask an interesting question

Get the Data

Explore the Data

Model the Data

Communicate/Visualize the Results

How were the data sampled?

Which data are relevant?

Are there privacy issues?

Ask an interesting question

Get the Data

Explore the Data

Model the Data

Communicate/Visualize the Results

Plot the data.

Are there anomalies or egregious issues?
Are there patterns?

Ask an interesting question

Get the Data

Explore the Data

Model the Data

Communicate/Visualize the Results

What did we learn?

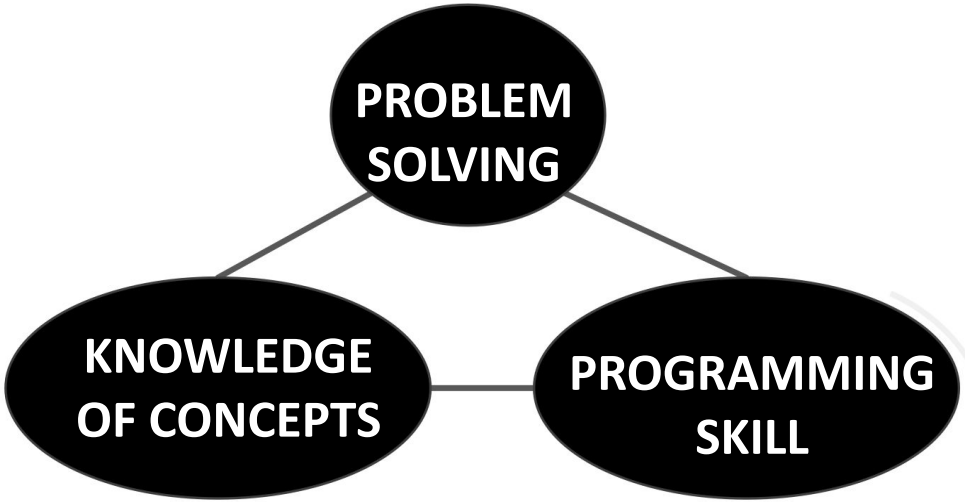
Do the results make sense?

Can we effectively tell a story?

What?

The material of the course will integrate the five key facets of an investigation using data:

1. Data collection: data wrangling, cleaning, and sampling to get a suitable data set.
2. Data management: accessing data quickly and reliably.
3. Exploratory data analysis; generating hypotheses and building intuition.
4. Prediction or statistical learning.
5. Communication: summarizing results through visualization, stories, and interpretable summaries.


PROBLEM SOLVING
KNOWLEDGE OF CONCEPTS
PROGRAMMING SKILL

TOPICS

- represent knowledge with **data structures**
- **iteration and recursion** as computational metaphors
- **abstraction** of procedures and data types
- **organize and modularize** systems using object classes and methods

WHAT DOES A COMPUTER DO

- Fundamentally:
  - performs **calculations**

    a billion calculations per second!
  - **remembers** results

    100s of gigabytes of storage!
- What kinds of calculations?
  - **built-in** to the language
  - ones that **you define** as the programmer
- computers only know what you tell them

TYPES OF KNOWLEDGE

- **declarative knowledge** is **statements of fact**.
  - someone will win a Google Cardboard before class ends

- **imperative knowledge** is a **recipe** or "how-to".

1) Students sign up for raffle
2) Ana opens her IDE
3) Ana chooses a random number between 1st and nth responder
4) Ana finds the number in the responders sheet. Winner!

A NUMERICAL EXAMPLE

- square root of a number `x` is `y` such that `y*y = x`
- recipe for deducing square root of a number `x` (16)

1) Start with a **guess**, `g`
2) If `g*g` is **close enough** to `x`, stop and say `g` answer
3) Otherwise make a **new guess** by averaging `g` and `x/g`
4) Using the new guess, **repeat** process until close enough

g	g*g	x/g	(g+x/g)/2
3	9	16/3	4.17
4.17	17.36	3.837	4.0035
4.0035	16.0277	3.997	4.000002

WHAT IS A RECIPE?

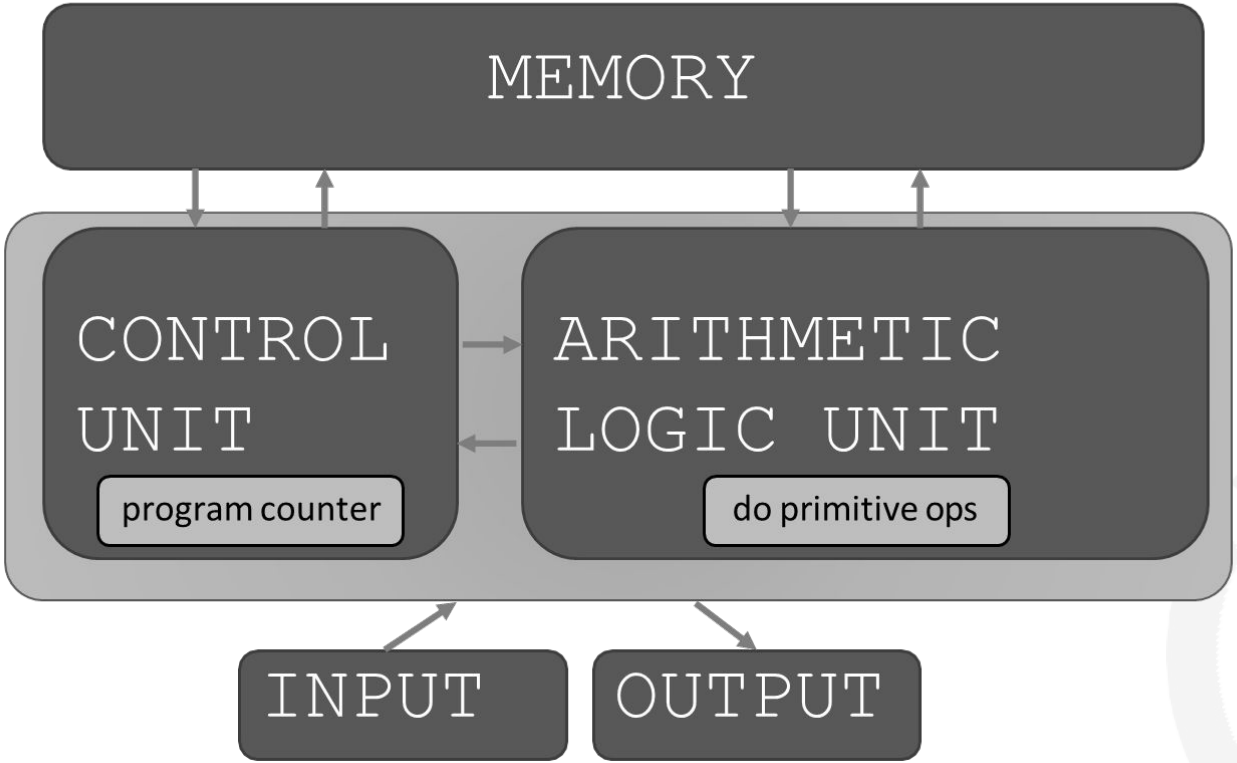
1. sequence of simple **steps**
2. **flow of control** process that specifies when each step is executed
3. a means of determining **when to stop**

1+2+3 = an **algorithm**!

COMPUTERS ARE MACHINES

- how to capture a recipe in a mechanical process
- **fixed program** computer
  - calculator
- **stored program** computer
  - machine stores and executes instructions

BASIC MACHINE ARCHITECTURE



STORED PROGRAM COMPUTER

- sequence of **instructions stored** inside computer
  - built from predefined set of primitive instructions
    1) arithmetic and logic
    2) simple tests
    3) moving data

- special program (interpreter) **executes each instruction in order**
  - use tests to change flow of control through sequence
  - stop when done

BASIC PRIMITIVES

- Turing showed that you can **compute anything** using 6 primitives
- modern programming languages have more convenient set of primitives
- can abstract methods to **create new primitives**
- anything computable in one language is computable in any other programming language

CREATING RECIPES

- a programming language provides a set of primitive **operations**

- **expressions** are complex but legal combinations of primitives in a programming language

- expressions and computations have **values** and meanings in a programming language

ASPECTS OF LANGUAGES

- **syntax**
  - English: `"cat dog boy"` ✗ not syntactically valid
    `"cat hugs boy"` ✓ syntactically valid
  - programming language: `"hi"5` ✗ not syntactically valid
    `3.2*5` ✓ syntactically valid

ASPECTS OF LANGUAGES

- **static semantics** is which syntactically valid strings have meaning
  - English: `"I are hungry"` ⇒ syntactically valid
    but static semantic error
  - programming language: `3.2*5` ⇒ syntactically valid
    `3+"hi"` ⇒ static semantic error

ASPECTS OF LANGUAGES

- **semantics** is the meaning associated with a syntactically correct string of symbols with no static semantic errors
  - English: can have many meanings "`Flying planes can be dangerous`"
  - programming languages: have only one meaning but may not be what programmer intended

WHERE THINGS GO WRONG

- **syntactic errors**
  - common and easily caught
- **static semantic errors**
  - some languages check for these before running program
  - can cause unpredictable behavior
- no semantic errors but **different meaning than what programmer intended**
  - program crashes, stops running
  - program runs forever
  - program gives an answer but different than expected

PYTHON PROGRAMS

- a **program** is a sequence of definitions and commands
  - definitions **evaluated**
  - commands **executed** by Python interpreter in a shell
- **commands** (statements) instruct interpreter to do something
- can be typed directly in a **shell** or stored in a **file** that is read into the shell and evaluated
- Lab 01 will introduce you to these in Anaconda

OBJECTS

- programs manipulate **data objects**
- objects have a **type** that defines the kinds of things programs can do to them
  - Ana is a human so she can walk, speak English, etc.
  - Chewbacca is a wookie so he can walk, “mwaaarhrhh”, etc.
- objects are
  - scalar (cannot be subdivided)
  - non-scalar (have internal structure that can be accessed)

SCALAR OBJECTS

- `int` – represent **integers**, ex. `5`
- `float` – represent **real numbers**, ex. `3.27`
- `bool` – represent **Boolean** values `True` and `False`
- `NoneType` – **special** and has one value, `None`
- can use `type()` to see the type of an object

TYPE CONVERSIONS (CAST)

- can **convert object of one type to another**
- `float(3)` converts integer `3` to float `3.0`
- `int(3.9)` truncates float `3.9` to integer `3`

PRINTING TO CONSOLE

- to show output from code to a user, use `print` command

```
In [11]: 3+2
Out[11]: 5

In [12]: print(3+2)
5
```

EXPRESSIONS

- **combine objects and operators** to form expressions
- an expression has a **value**, which has a type
- syntax for a simple expression

```
<object> <operator> <object>
```

OPERATORS ON ints and floats

- `i+j` → the **sum**
- `i-j` → the **difference**
- `i*j` → the **product**
- `i/j` → **division**

if both are ints, result is int
if either or both are floats, result is float

result is float

- `i%j` → the **remainder** when `i` is divided by `j`
- `i**j` → `i` to the **power** of `j`

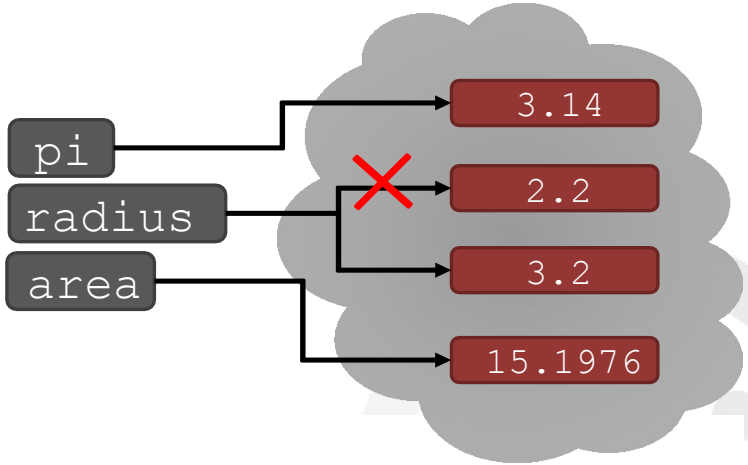
PROGRAMMING vs MATH

- in programming, you do not “solve for x”

```
pi = 3.14159
radius = 2.2
# area of circle
area = pi*(radius**2)
radius = radius+1
```

CHANGING BINDINGS

```
pi = 3.14
radius = 2.2
area = pi*(radius**2)
radius = radius+1
```



CHANGING BINDINGS

- can **re-bind** variable names using new assignment statements
- previous value may still stored in memory but lost the handle for it
- value for area does not change until you tell the computer to do the calculation again