Welcome!

Making Game with Python (2)

Zhihong (John) Zeng & Andrew Zeng

Today

- Course info
- Computer fundamentals
- Review python basics
- Install software

Course Info

Make Games with Graphics

Making Games with Python & Pygame covers the Pygame library with the source code for 11 games. Making Games was written as a sequel for the same age range as Invent with Python. Once you have an understanding of the basics of Python programming, you can now expand your abilities using the Pygame library to make games with graphics, animation, and sound.

The book features the source code to 11 games. The games are clones of classics such as Nibbles, Tetris, Simon, Bejeweled, Othello, Connect Four, Flood It, and others.

Buy on Amazon | Read Online for Free

CLI vs GUI

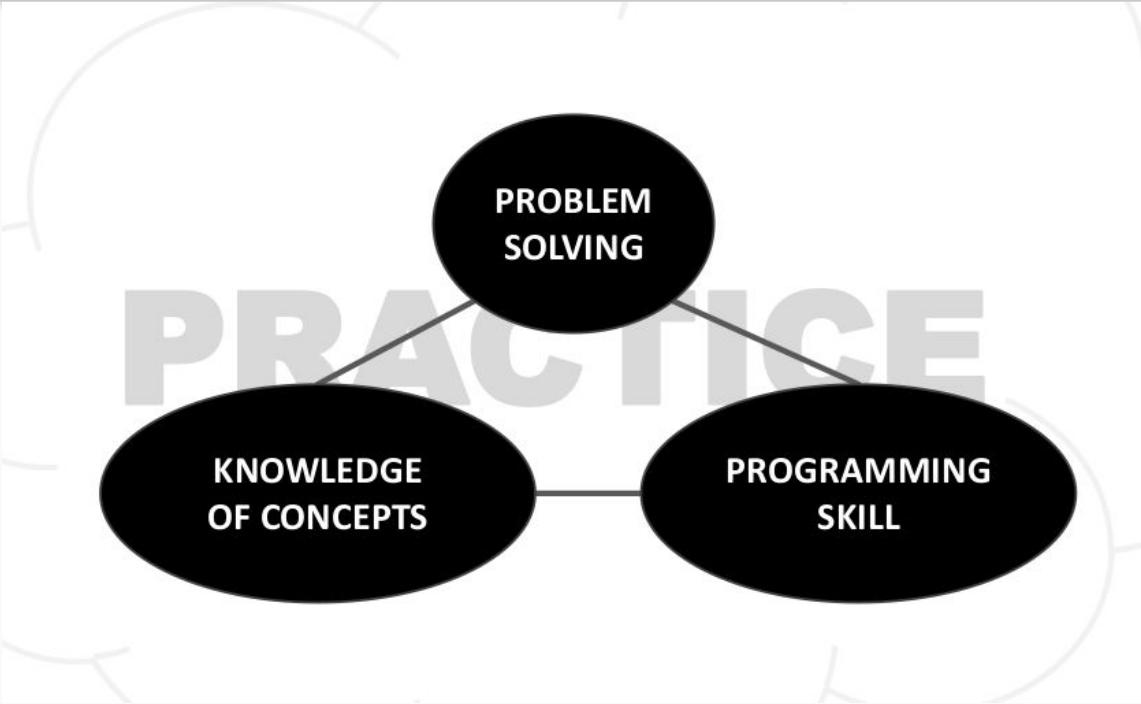
CLI:

- Command Line Interface
- Input: keyboard input
- Output: print to terminar

GUI:

- Graphical User Interface
- Input: mouse and keyboard event
- Output: windows with images and colors

Course Info

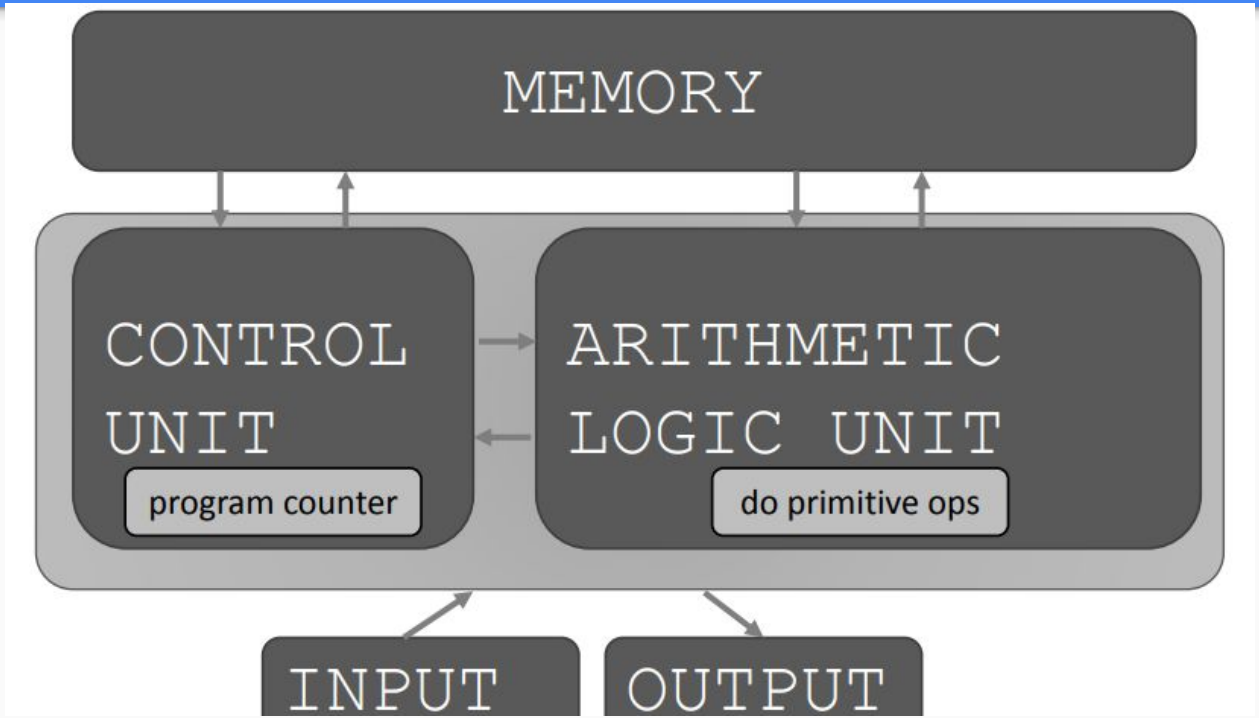

PROBLEM SOLVING
PRACTICE
KNOWLEDGE OF CONCEPTS
PROGRAMMING SKILL

Computer Fundamentals

What does a Computer do?

- Fundamentally:
  - Performs calculations
  - Remembers results
- What kinds of calculations:
  - Built-in to the language
  - Ones that you define as the programmer
- Computers only know what you tell them

Basic Computer Architecture



What is a programming recipe

- Sequence of simple steps
- Flow of control process that specifies when each step is executed
- A means of determining when to stop

Basic Primitives

- Tuning showed that you can compute anything using 6 primitives (left, right, scan, print, erase, and do nothing)
- Modern programming languages have more convenient set of primitives
- Use abstract methods to create new primitives
- Anything computable in one language is computable in any other programming language

Aspects of Languages

- Primitive constructs
  - English: words
  - Programming language: numbers, strings, simple operators

Aspects of Languages

- Syntax
  - English:
    - “Cat dog boy” -> not syntactically valid
    - “Cat hugs boy” -> syntactically valid but
  - Programming language
    - “hi” 5 -> not syntactically valid
    - 3.2*5 -> syntactically valid

Where Things Go Wrong

- Static semantic errors
  - Some languages check for these before running program
- No semantic errors but different meaning from what a programmer intended
  - Program crashes, stop running
  - Program run forever
  - Program gives an answer but different from expected

Python Basics

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 intended

Objects

- Programs manipulate data objects
- Objects have a type that defines the kinds of things program can do to them
  - Andrew is a human so he can walk, speak English, etc.
  - Chewbacca (Star wars) 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 no value, None
- Can use type() to see the type of an object
  - >>>type(5) -->int
  - >>>type(3.0) -->float

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
  - int(‘321’) converts string ‘321’ to integer 321
  - str(123) converts integer 123 to string ‘123’

Printing to Console

- To show output from code to a user, use print command
  - print(3)
  - print(3+2)
  - print('ABC')

Operators on ints and floats

- i + j → sum
- i - j → difference
- i * j → product

  Note: if both are ints, result is int. If either or both are floats, result is float

- i / j → division. result is float
- i % j → remainder when i is divided by j
- i ** j → i to the power of j

Operation precedence (order)

- Parentheses used to tell Python to do these operations first
- Operator precedence without parentheses
  - **
  - *
  - /
  - + and - executed left to right, as appear in expression

Binding Variables and Values

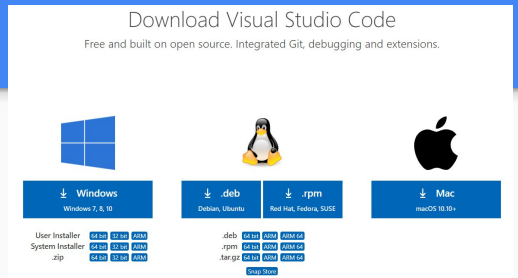
- Equal sign is an assignment of a value to a variable name
  - Pi = 3.14
  - Pi_approx = 22/7
- Value stored in computer memory
- An assignment binds name to value
- Retrieve value associated with name or variable by invoking the name

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

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

Install Software



- visual studio code (search: visual studio code install)
  - https://code.visualstudio.com/download
- Open visual studio code (vcode)
  - Add python extension for visual studio code
  - Add pylint extension
  - Open your work folder
  - If you have multiple python versions, press “ctr+shift+p” -> python: select interpreter ->choose python 3.7/8/9
  - Open a new terminal from vcode
  - Create virtualenv
    - enter the following command:
      - Windows: py -m venv .venv
      - MacOX or linus: python3 -m venv .venv
    - 1) Optional 1: Click yes for pop-up question “...new workspace..”, then open a new terminal
    - 2) Optional 2: press ctr+shift+p -> python: select interpreter -> choose python 3/7/8/9 **(.venv)**

Run: py -m pip install pygame

Additional question

Solution to “Running Scripts Is Disabled On This System” Error On Powershell

1) in visual studio code, find and open settings.json by ctr+shift+p and typing "setting.json"

2) ~~option: remove "terminal.integrated.shellArgs.windows": ["-ExecutionPolicy", "Bypass"] if you see them, because it is deprecated.~~

3) adding -ExecutionPolicy Bypass args shown here

```
"terminal.integrated.profiles.windows": {
  "PowerShell": {
    "source": "PowerShell",
    "icon": "terminal-powershell",
    "args": ["-ExecutionPolicy", "Bypass"]
  }
},
"terminal.integrated.defaultProfile.windows": "PowerShell",
```

4) restart VS Code and open a Terminal.