

YSC2221-1

INTRODUCTION TO

PYTHON (GROUP 1)

INFORMATION ABOUT THIS COURSE

Your Lecturer

- ▶ CHENG, Ho Lun, Alan, 鄭浩璘
 - Hong Kong
 - UIUC, Duke (USA)
- ▶ Hobbies:
 - Teaching
 - 3G: Graphics, Geometry, Games
 - Animation, anime, comics, movies, etc.
- ▶ Email:
 - hcheng@comp.nus.edu
 - [\(alan@comp.nus.edu\)](mailto:(alan@comp.nus.edu))
 - **DO NOT send to dcschl**
 - **DO NOT send codes to me**
 - Office: AS6 #05-03
- ▶ Ext. 68732



My Other courses

- CS3241 Computer Graphics
- CS4235 Game Development Projects
- CS2020 Data Structure and Algorithm (Accel)
- CS5237 Computational Geometry

Something about yourself

- Where are you from
- What's your major
- Something interesting about yourself
 - or
- What do you want to use Python for

Syllabus

Week 1: Introduction to Python

Week 2: Simple functions and debugging

Week 3: More on functions and Program designs

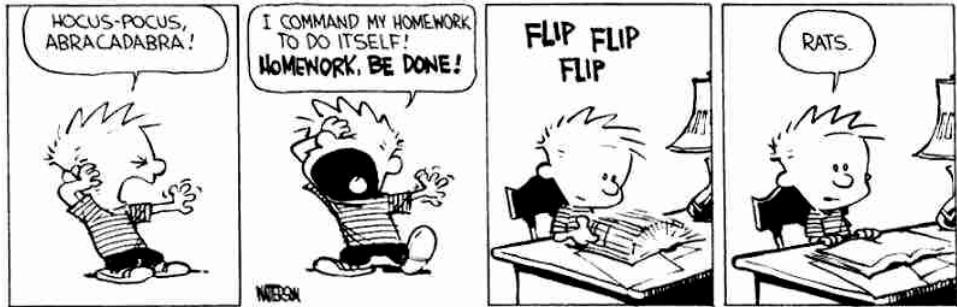
Week 4: Sequences and File I/O

Week 5: Data Visualization and Multi-dimensional Array

Week 6: More Packages

CA

- Lab/homework 30%
 - = 5 x 6%
 - Weekly
- Written Quiz 20%
 - 30th Aug
- Practical test 20%
 - Last Lecture
- Take home exam 30%
 - After all lectures



Today

- What is programming about?
- Why Python?

What is programming?

Programmer



What my friends think I do



What my mom thinks I do



What society thinks I do



What my boss thinks I do



What I think I do



What I actually do

Imagine



Running a Booth

- Dropping a ball from the top and you will get
 - 3: Big prize
 - 2: Medium prize
 - 1: Small prize



Let's see what we have for prizes

- Giant Teddy Bears
 - X 10



- Water guns
 - X 50



- Candy
 - X 200



Your Job is

- Run the booth until the end of the day or all prizes given out
 - 3: Giant Teddy Bear
 - 2: Water Gun
 - 1: Candy
- What is the potential problem?
- You may run out of Teddy Bears!
 - Or Water guns or candies
- Any suggestion?



Version 1

Let him play the Plinko

If someone strikes a “3”

 Give him a bear

If someone strikes a “2”

 Give him a water gun

If someone strikes a “1”

 Give him one pathetic candy



Version 1.1

Let him play the Plinko

If someone strikes a “3”

If we have bears

Give him a bear

If someone strikes a “2”

If we have water guns

Give him a water gun

If someone strikes a “1”

If we have candies

Give him one pathetic candy



Version 1.2

Let him play the Plinko

If someone strikes a “3”

If we have bears

Give him a bear

Otherwise

Give him a water gun

If someone strikes a “2”

If we have water guns

Give him a water gun

If someone strikes a “1”

If we have candies

Give him one pathetic candy



Version 1.3

Let him play the Plinko
If someone strikes a “3”
 If we have bears
 Give him a bear
 Otherwise if we have water guns
 Give him a water gun
 Otherwise
 Give him a pathetic candy
If someone strikes a “2”
 If we have water guns
 Give him a water gun
If someone strikes a “1”
 If we have candies
 Give him one pathetic candy



Version 2

While we have prizes left, we do the following:

Let someone play the Plinko

If he strikes a “3”

If we have bears

 Give him a bear

Otherwise let him choose any prize

If someone strikes a “2”

If we have water guns

 Give him a water gun

Otherwise give him a candy

If someone strikes a “1”

If we have candies

 Give him one pathetic candy

Otherwise, sorry, you don’t even have a pathetic candy

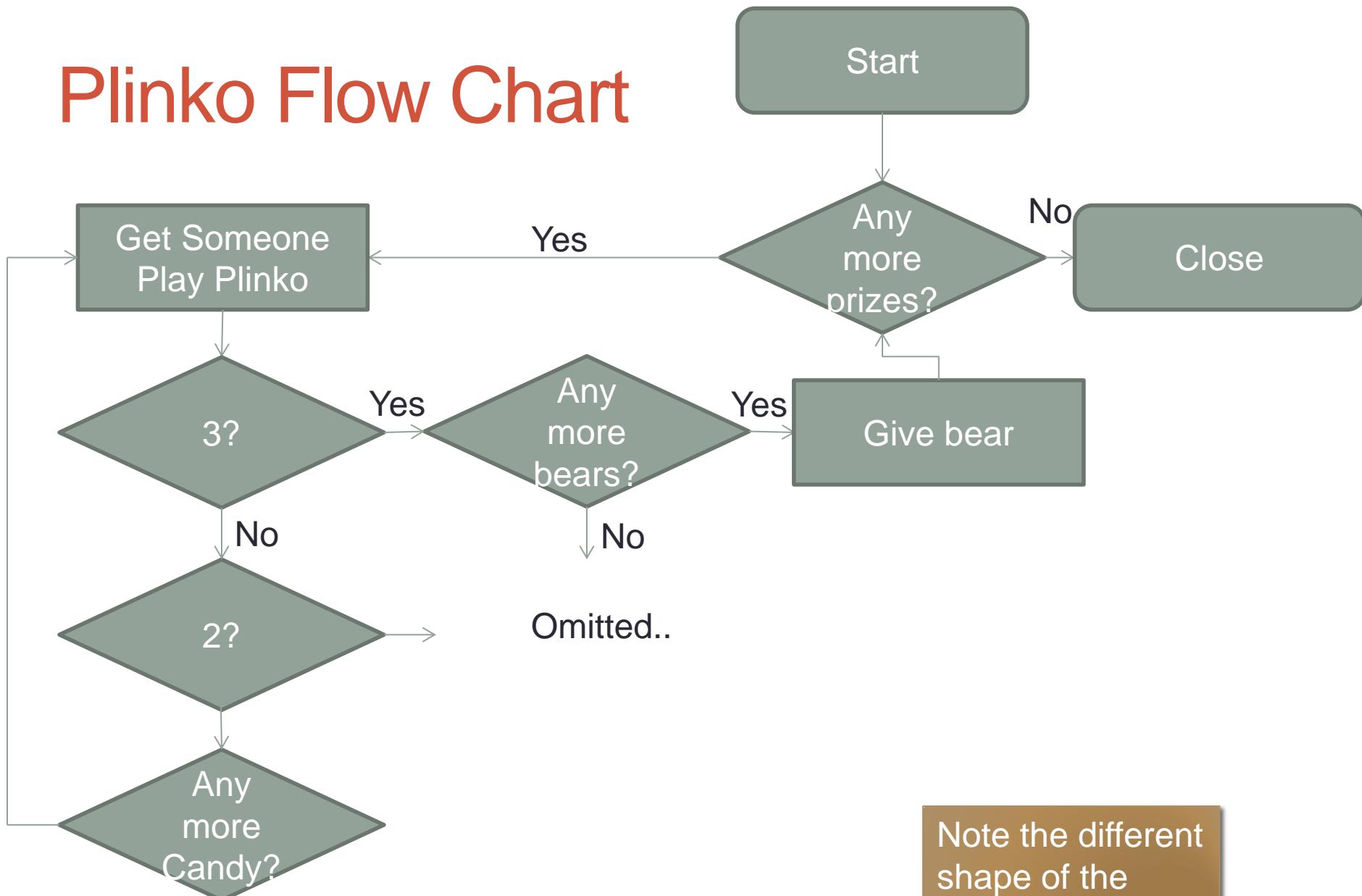
This is called
pseudocode

But still not perfect

Before Real Coding

- A Program can be expressed by
 - **Pseudocode**
 - An artificial and informal language that helps programmers develop algorithms.
 - "text-based"
 - Or A **Flow Chart**

Plinko Flow Chart



Note the different shape of the bubbles

Choose Your Own Adventure

From Wikipedia, the free encyclopedia

This article is about the trademarked book series. For the genre, see [Gamebook](#). For the TV series, see [Lawrence Leung's Choose Your Own Adventure](#).

Choose Your Own Adventure is a series of children's [gamebooks](#) where each story is written from a second-person point of view, with the reader assuming the role of the protagonist and making choices that determine the main character's actions and the plot's outcome. The series was based upon a concept created by [Edward Packard](#) and originally published by Constance Cappel's and [R. A. Montgomery](#)'s Vermont Crossroads Press as the "Adventures of You" series, starting with Packard's *Sugarcane Island* in 1976.^[1]

Choose Your Own Adventure, as published by [Bantam Books](#), was one of the most popular children's series during the 1980s and 1990s, selling more than 250 million copies between 1979 and 1998.^[2] When Bantam, now owned by [Random House](#), allowed the *Choose Your Own Adventure* trademark to lapse, the series was relaunched by [Chooseco](#), which now owns the trademark. Chooseco does not reissue titles by Packard, who has started his own imprint, U-Ventures.^[3]

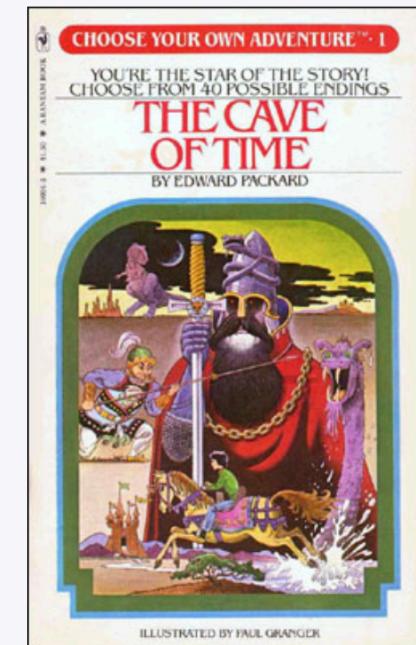
Contents [hide]

- 1 Format
- 2 History
- 3 See also
- 4 References
- 5 External links

Format [edit]

Originally created for 7- to 14-year-olds, the books are written in the second person. The protagonist—that is, the reader—takes on a role relevant to the adventure; for example, private investigator, mountain climber, race car driver, doctor, or spy. Stories are generally gender and race neutral, though in some cases, particularly in illustrations, presumption of a male

Choose Your Own Adventure



The Cave of Time by Edward Packard, the first book in the series

Cover artist Paul Granger
Language English

Flow Chart Bubble Types

 Terminator
Indicates the beginning or end of a program flow in your diagram.

 Process
Indicates any processing function.

 Decision
Indicates a decision point between two or more paths in a flowchart.

 Delay
Indicates a delay in the process.

 Data
Can represent any type of data in a flowchart.

 Document
Indicates data that can be read by people, such as printed output.

 Multiple documents
Indicates multiple documents.



Subroutine
Indicates a predefined (named) process, such as a subroutine or a module.



Preparation
Indicates a modification to a process, such as setting a switch or initializing a routine.



Display
Indicates data that is displayed for people to read, such as data on a monitor or projector screen.



Manual input
Indicates any operation that is performed manually (by a person).



Manual loop
Indicates a sequence of commands that will continue to repeat until stopped manually.



Loop limit
Indicates the start of a loop. Flip the shape vertically to indicate the end of a loop.



Stored data
Indicates any type of stored data.



Connector
Indicates an inspection point.



Off-page connector
Use this shape to create a cross-reference and hyperlink from a process on one page to a process on another page.



Off-page connector



Off-page connector



Off-page connector



Or
Logical OR



Summing junction
Logical AND



Collate
Indicates a step that organizes data into a standard format.



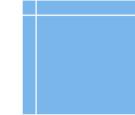
Sort
Indicates a step that organizes items list sequentially.



Merge
Indicates a step that combines multiple sets into one.



Database
Indicates a list of information with a standard structure that allows for searching and sorting.

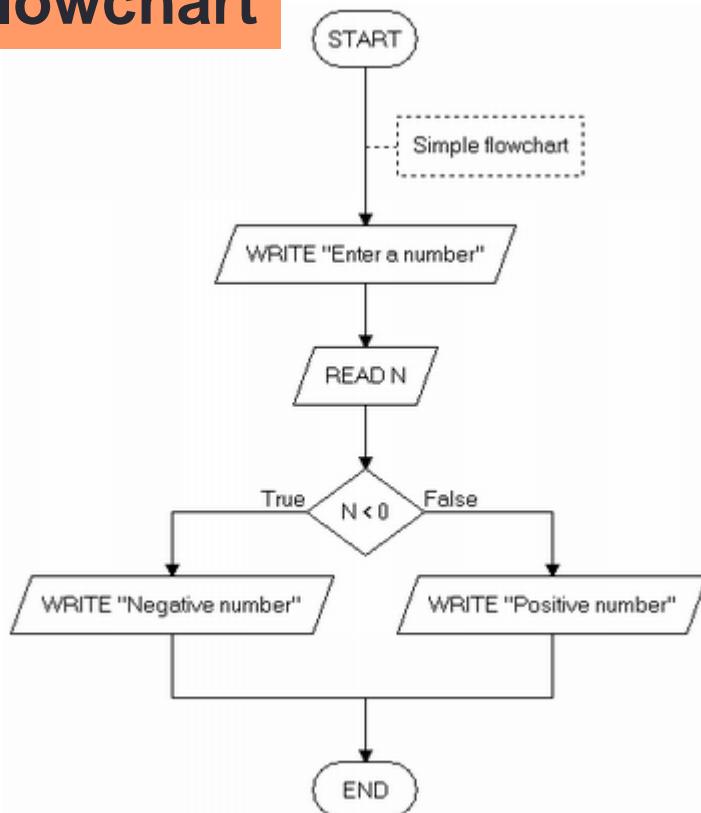


Internal storage
Indicates an internal storage device.

Algorithm

■ Ways of representing an algorithm:

Flowchart



Pseudocode

PSEUDOCODE

set total to zero

get list of numbers

loop through each number in the list
 add each number to total
end loop

if number more than zero
 print "it's positive" message
else
 print "it's zero or less" message
end if

Algorithms

- Named for al-Khwārizmī (780-850)
 - Persian mathematician
- Many ancient algorithms
 - Multiplication: Rhind Papyrus
 - Babylon and Egypt: ~1800BC
 - Euclidean Algorithm: Elements
 - Greece: ~300BC
 - Sieve of Eratosthenes
 - Greece: ~200BC



What is an
Algorithm?



Algorithm (noun.)

Word used by programmers when...
they do not want to explain what they did.

Algorithm (1/3)

- An **algorithm** is a well-defined computational procedure consisting of *a set of instructions*, that takes some value or set of values as *input*, and produces some value or set of values as *output*.



‘Algorithm’ stems from ‘Algoritmi’, the Latin form of al-Khwārizmī, a Persian mathematician, astronomer and geographer.
Source: <http://en.wikipedia.org/wiki/Algorithm>

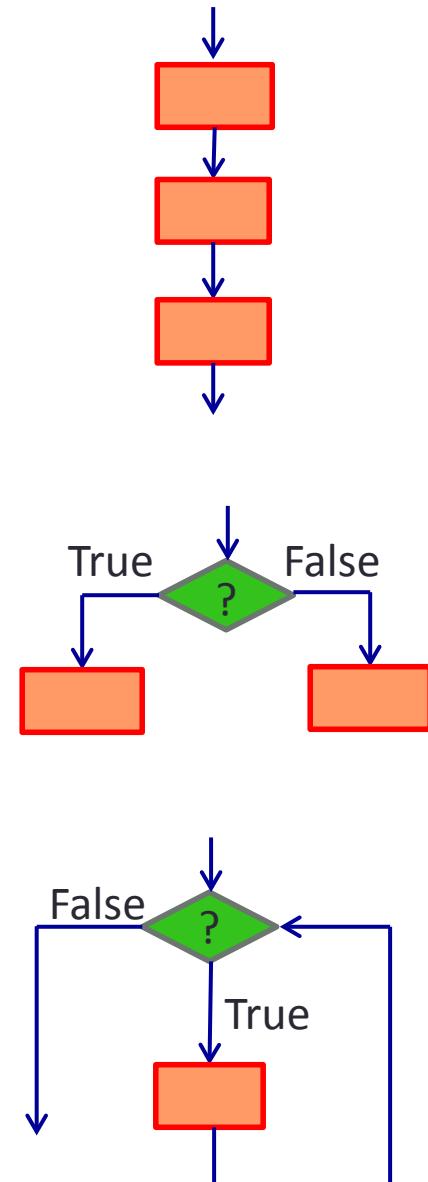
Control Structures

- An algorithm is a set of instructions, which are followed sequentially by default.
- However, sometimes we need to change the default sequential flow.
- We study 3 control structures.

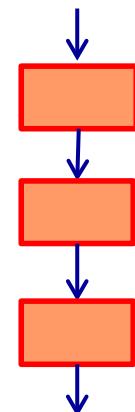
CONTROL STRUCTURES

Control Structures

Sequence	• Default
Selection	• Also called branching
Repetition	• Also called loop



Control Structure: Sequence



Method 1 Making Vanilla Pound Cake

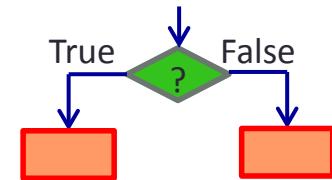
1. Gather your ingredients. Pound cake is one of the simplest cakes to bake. ...
2. Preheat the oven to 325 degrees.
3. Grease a cake pan. ...
4. Cream the butter and sugar. ...
5. Add the eggs and vanilla. ...
6. Stir in the cake flour. ...
7. Pour the batter into the pan. ...|
8. Bake the cake for an hour and 15 minutes.



4 Ways to Bake a Cake - wikiHow

www.wikihow.com/Bake-a-Cake

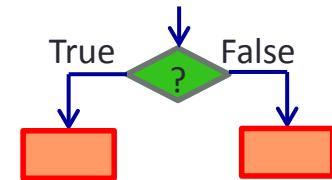
Control Structure: Selection



- **If** the player strikes a “3”
 - Give him a bear
- **Else**
 - Let him choose a water gun or a candy



Control Structure: Selection



```
If (a condition is true)
```

```
    Do A
```

```
Else
```

```
    Do B
```

Can be **MORE THAN** one single instruction

- For example:

```
If (I have $1000000000000000)
```

```
    Buy a car
```

```
    Eat a lot of buffets
```

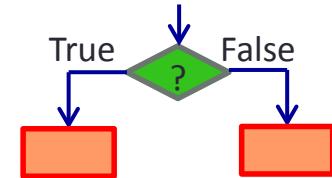
```
    Go travel
```

```
    Quit NUS!
```

```
Else
```

```
    Be good and study
```

Control Structure: Selection



```
If (a condition is true)
```

```
Do A
```

```
Else
```

```
Do B
```

Can be **MORE THAN** one single instruction

- For example:

```
If (I have $1000000000000000)
```

```
If (I am heartless)
```

```
Buy a car
```

```
Eat a lot of buffets
```

```
Go travel
```

```
Quit NUS!
```

```
Else
```

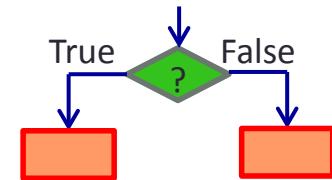
```
    donate all the money to charity
```

```
Else
```

```
    Be good and study
```

Nested "if"

Control Structure: Selection



If (a condition is true)

Do A

Else

Do B

Can be **WITHOUT** “else”

- For example:

If (I have \$1000000000000000)

Buy a car

Eat a lot of buffets

Go travel

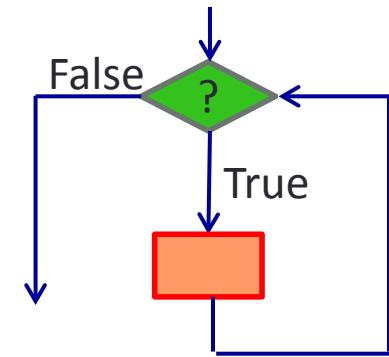
Quit NUS!

Else

Be good and study

Control Structure: Repetition

- While there are prizes left
 - Play Plinko and give prizes



Control Structure: Repetition

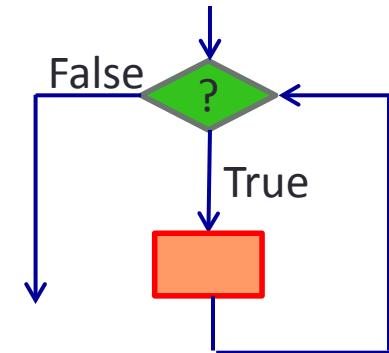
- While (a condition)
 - Do something

- For example

```
While ( I am hungry )  
    Eat a bun
```

- Again, can be more than one single instruction

```
While(I have money in bank)  
    Take some money out from bank  
    Eat an expensive meal  
While(I have money in my wallet)  
    Go Shopping
```



Example: Scrabble

- Let's say you have four letters in your hands:
 - 'B', 'A', 'G' and 'Z'
- With points
 - $B = 3, A = 1, G = 2, Z = 10$
- Assuming I can place any of them on the board
- How do I come up with a valid English word that maximize my points?
- Any ideas?

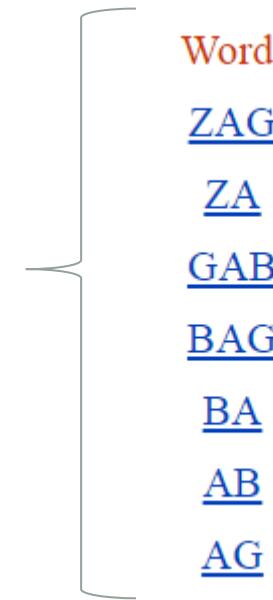


Layer 1

- Find out every combinations of words formed by the letters
- Calculate the score for every combination and choose the largest one

Layer 1 in Pseudo Code

- For every combination w of { ‘B’, ‘A’, ‘G’, ‘Z’ }
- How?
- Let’s skip it first, assuming we can compute all of the “valid” words in the dictionary, i.e.
 - From scrabblecheat.com
- Let’s store these words in a set called S



Layer 1 in Pseudo Code

- Calculate the score for every word w in the set S
- Choose the largest score

Value	Word
13	<u>ZAG</u>
11	<u>ZA</u>
6	<u>GAB</u>
6	<u>BAG</u>
4	<u>BA</u>
4	<u>AB</u>
3	<u>AG</u>

Layer 2 Pseudo Code

- Let x be any word in S (say the first one)
- And $s =$ the score of x
 - In which, later, I assume s stores the maximum score so far, and x is the word with that score
- For every combination w in S
 - Calculate the score for w , and store this score in a number n
 - if $n > s$
 - replace s by n and replace x by w

Layer 2 Pseudo Code

- Let x be any word in S (say the first one)
- And $s =$ the score of x
 - In which, later, I assume s stores the maximum score so far, and x is the word with that score

w	n	x	s
		AG	3

Layer 2 Pseudo Code

- For every combination **w** in S
 - Calculate the score for w, and store this score in a number n
 - if $n > s$
 - replace s by n and replace x by w

w	n	x	s
		AG	3
AG	3	AG	3
AB			
BAG			
GAB			
BA			
ZAG			
ZA			

Layer 2 Pseudo Code

- For every combination **w** in S
 - Calculate the score for w, and store this score in a number n
 - if $n > s$
 - replace s by n and replace x by w

w	n	x	s
		AG	3
AG	3	AG	3
AB	4	AB	4
BAG			
GAB			
BA			
ZAG			
ZA			

Layer 2 Pseudo Code

- For every combination **w** in S
 - Calculate the score for w, and store this score in a number n
 - if $n > s$
 - replace s by n and replace x by w

w	n	x	s
		AG	3
AG	3	AG	3
AB	4	AB	4
BAG	6	BAG	6
GAB			
BA			
ZAG			
ZA			

Layer 2 Pseudo Code

- For every combination **w** in S
 - Calculate the score for w, and store this score in a number n
 - if $n > s$
 - replace s by n and replace x by w

w	n	x	s
		AG	3
AG	3	AG	3
AB	4	AB	4
BAG	6	BAG	6
GAB	6	BAG	6
BA			
ZAG			
ZA			

Layer 2 Pseudo Code

- For every combination **w** in S
 - Calculate the score for w, and store this score in a number n
 - if $n > s$
 - replace s by n and replace x by w

w	n	x	s
		AG	3
AG	3	AG	3
AB	4	AB	4
BAG	6	BAG	6
GAB	6	BAG	6
BA	4	BAG	6
ZAG			
ZA			

Layer 2 Pseudo Code

- For every combination **w** in S
 - Calculate the score for w, and store this score in a number n
 - if $n > s$
 - replace s by n and replace x by w

w	n	x	s
		AG	3
AG	3	AG	3
AB	4	AB	4
BAG	6	BAG	6
GAB	6	BAG	6
BA	4	BAG	6
ZAG	13	ZAG	13
ZA			

Layer 2 Pseudo Code

- For every combination w in S
 - Calculate the score for w , and store this score in a number n
 - if $n > s$
 - replace s by n and replace x by w

w	n	x	s
		AG	3
AG	3	AG	3
AB	4	AB	4
BAG	6	BAG	6
GAB	6	BAG	6
BA	4	BAG	6
ZAG	13	ZAG	13
ZA	11	ZAG	13

Layer 2 Pseudo Code

- Final answer lies in **x** and **s**
 - The word is “ZAG” with a score of 13

w	n	x	s
		AG	3
AG	3	AG	3
AB	4	AB	4
BAG	6	BAG	6
GAB	6	BAG	6
BA	4	BAG	6
ZAG	13	ZAG	13
ZA	11	ZAG	13

How do we generate S?

- Let x be any word in **S** (say the first one)
- And $s =$ the score of x
 - In which, later, I assume s stores the maximum score so far, and x is the word with that score
- For every combination w in **S**
 - Calculate the score for w , and store this score in a number n
 - if $n > s$
 - replace s by n and replace x by w

Let's leave it for now

Does it Work for Other Sets of Letters?

- Let x be any word in S (say the first one)
- And $s =$ the score of x
 - In which, later, I assume s stores the maximum score so far, and x is the word with that score
- For every combination w in S
 - Calculate the score for w , and store this score in a number n
 - if $n > s$
 - replace s by n and replace x by w

Yes! If we can generate the set S

What is the difference between

Algorithm vs Program

- Algorithm
 - Ideas
 - Machine independent

- Program
 - The final code on a machine
 - Machine dependent

Research Idea vs Thesis

- Research Idea
 - Can be drawings, sketches, concepts, in any languages
- Thesis
 - Well-written documents in any **languages**, English, German, Chinese, etc...

Programming Languages History

- <https://james-iry.blogspot.sg/2009/05/brief-incomplete-and-mostly-wrong.html>



THURSDAY, MAY 7, 2009

A Brief, Incomplete, and Mostly Wrong History of Programming Languages

1801 - Joseph Marie Jacquard uses punch cards to instruct a loom to weave "hello, world" into a tapestry. Redditors of the time are not impressed due to the lack of tail call recursion, concurrency, or proper capitalization.

1842 - Ada Lovelace writes the first program. She is hampered in her efforts by the minor inconvenience that she doesn't have any actual computers to run her code. Enterprise architects will later relearn her techniques in order to program in UML.

1936 - Alan Turing invents every programming language that will ever be but is changed by British Intelligence to be lost before he can



ABOUT ME

 JAMES IRY

SAN FRANCISCO, CA, UNITED STATES

AN OVERVIEW OF



Why are we learning Python?

- Clear and readable syntax
- Intuitive
- Natural expression
- Powerful
- Popular & Relevant
- Example: Paypal
 - ASF XML Serialization
 - C++
 - 1580 lines
 - Python
 - 130 lines



Who uses Python?

- Google
- Red Hat
- Dropbox
- Rackspace
- Twitter
- Facebook
- Raspberry Pi
- NASA
- CERN
- ITA
- Yahoo!
- Walt Disney
- IBM
- Reddit
- YouTube

Python Program without Learning

```
a = 1  
b = 2  
c = a + b  
if c < 0:  
    print('Yes')  
else:  
    print('No')
```

Intuitive!



The Environment: IDLE

- We use **IDLE** as an IDE
 - IDE: Integrated development environment
 - Meaning you can edit your program, run and debug it
 - Many different IDEs for different languages
- IDLE stands for
 - Integrated development and learning environment

A Screenshot of IDLE

Console
- Input
- output

Let's go for
some demo

Editor
and Your
program

The screenshot shows the Python 3.6.0 Shell and an open editor window. The Shell window displays the Python version information and a prompt (>>>). The editor window shows the code for `hi_graph.py`, which includes imports for tkinter, math, time, and atexit, along with various constants and helper functions for creating a graphical interface.

```
Python 3.6.0 (v3.6.0:41df79263a11, Dec 23 2016, 08:06:12) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.

>>>

hi_graph.py - C:\Users\dcschl\Dropbox\Courses\IT1007\re...
File Edit Format Run Options Window Help
from tkinter import *
from math import sin, cos, pi, sqrt, atan2
import time
import atexit, sys, threading

## Increase recursion stack size
sys.setrecursionlimit(2**20)

## Configuration
WINDOW_SIZE = 512
BORDER_OFFSET = 8

## Some derived values
CANVAS_SIZE = {'x': BORDER_OFFSET,
               'y': BORDER_OFFSET,
               'width': WINDOW_SIZE-2*BORDER_OFFSET,
               'height': WINDOW_SIZE-2*BORDER_OFFSET}

## Some helper functions
def identity(x):
    return x

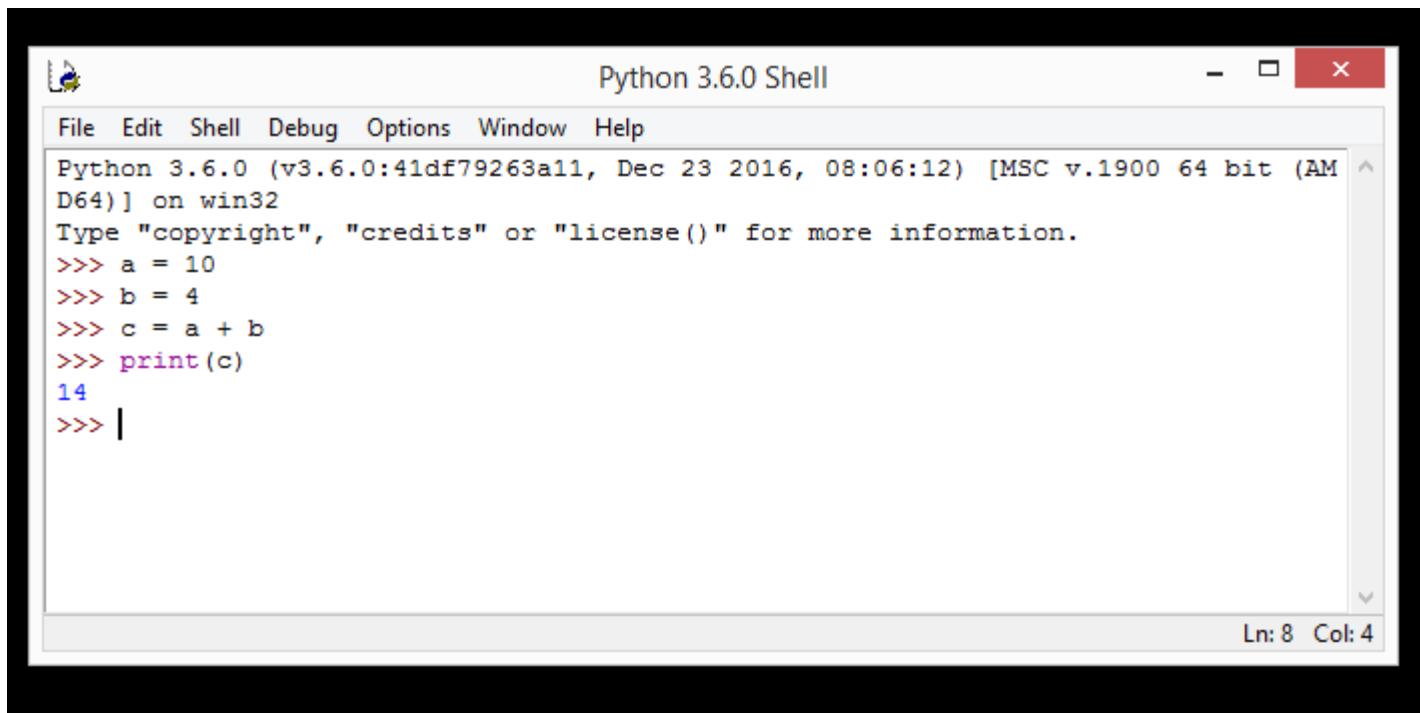
def composed(f, g):
    return lambda x: f(g(x))

def repeated(f, n):
    if (n == 0):
        return identity
    return composed(f, repeated(f, n-1))

## Data abstraction
```

You can

- Directly type into the console



The screenshot shows the Python 3.6.0 Shell window. The title bar reads "Python 3.6.0 Shell". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main window displays the following text:

```
Python 3.6.0 (v3.6.0:41df79263a11, Dec 23 2016, 08:06:12) [MSC v.1900 64 bit (AM  
D64)] on win32  
Type "copyright", "credits" or "license()" for more information.  
>>> a = 10  
>>> b = 4  
>>> c = a + b  
>>> print(c)  
14  
>>> |
```

The status bar at the bottom right indicates "Ln: 8 Col: 4".

- In which, we **seldom** do this

Or Run a file

The screenshot shows two windows. The top-left window is titled "Python 3.6.0 Shell" and displays the Python 3.6.0 startup message and a command prompt. The top-right window is titled "test1.py - C:\Users\dcschl\Desktop\test1.py (3.6.0)" and shows the Python source code for a script named test1.py.

```
Python 3.6.0 (v3.6.0:41df79263a11, Dec 23 2016, 08:06:12)
D64) on win32
Type "copyright", "credits" or "license()" for more information
>>>
```

```
a = 3
b = 1
c = a + b
if c < 0:
    print('c < 0')
else:
    print('c > 0')
```

The screenshot shows a Python shell window with a red circle highlighting the output of the run command. A callout arrow points from the text "Output of your" to the highlighted output.

```
Python 3.6.0 (v3.6.0:41df79263a11, Dec 23 2016, 08:06:12)
D64) on win32
Type "copyright", "credits" or "license()" for more information
>>>
=====
RESTART: C:\Users\dcschl\Desktop\test1.py
c > 0
>>>
```

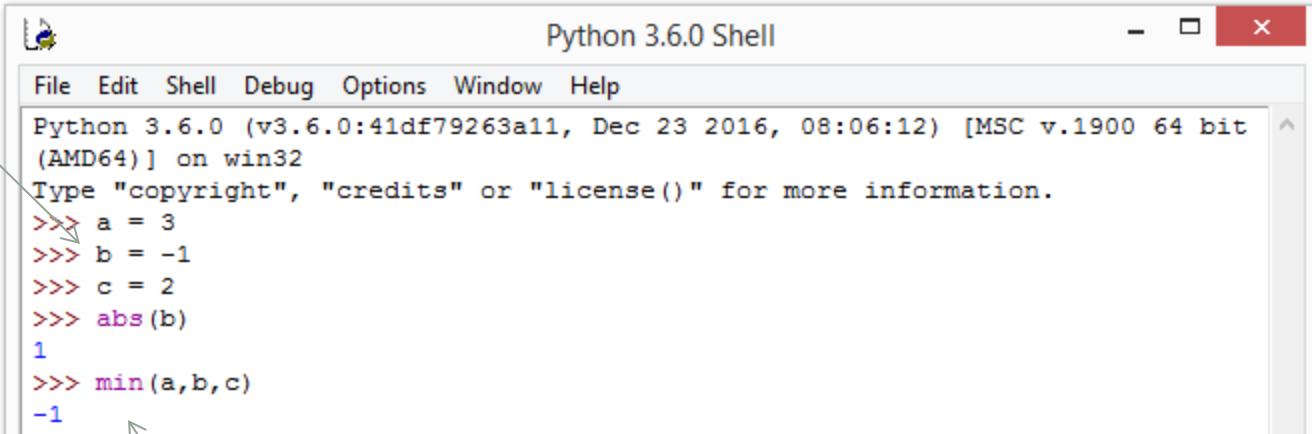
Output of your

```
File Edit Format Run Options Window Help
File Edit Format Run Options Window Help
Python Shell
Check Module Alt+X
Run Module F5
print('c < 0')
else:
    print('c > 0')
```

Functions and Variables

- We will give more details on this in the next lecture

a, b, c
are
variables



The screenshot shows the Python 3.6.0 Shell window. The title bar reads "Python 3.6.0 Shell". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main window displays the following Python session:

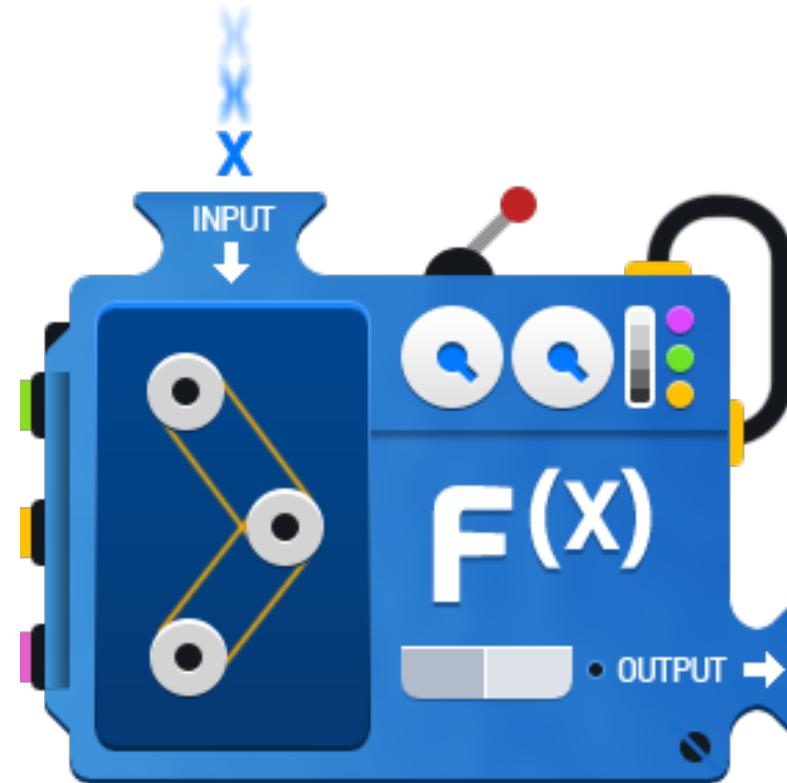
```
Python 3.6.0 (v3.6.0:41df79263a11, Dec 23 2016, 08:06:12) [MSC v.1900 64 bit
(AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.

>>> a = 3
>>> b = -1
>>> c = 2
>>> abs(b)
1
>>> min(a,b,c)
-1
```

abs() and min()
are functions

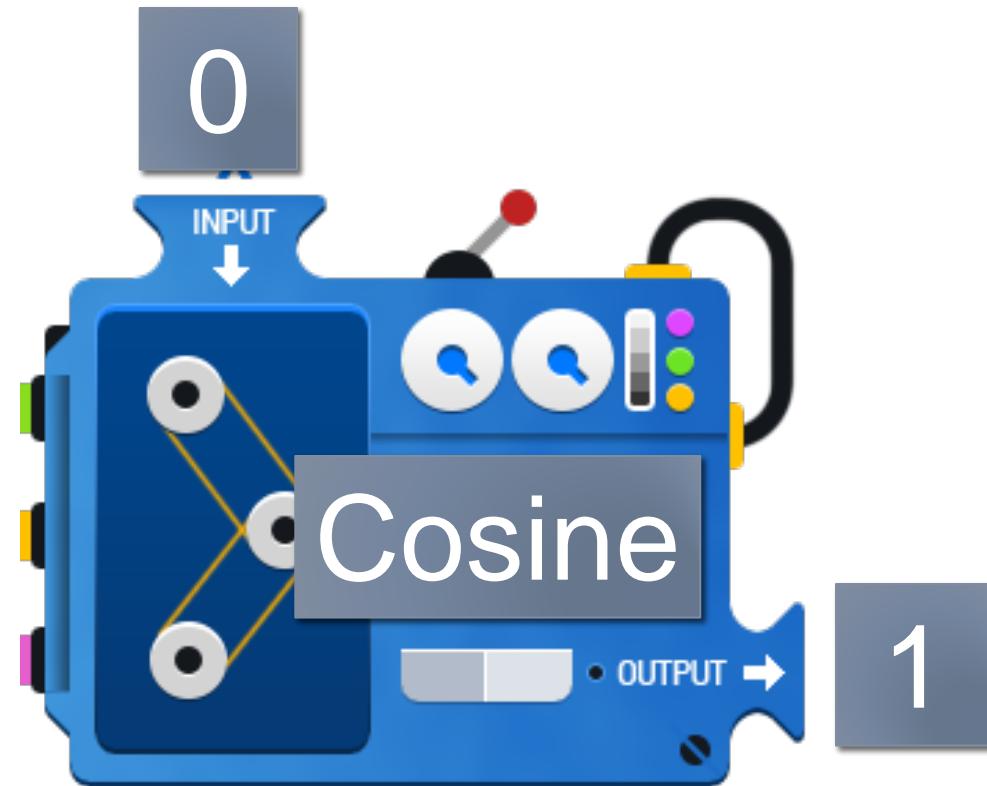
Functions

- A function is like a black box
 - You put in something for the input
 - And it will produce a new thing for the output



For example

- “Cosine” is a function
 - Input 0
 - Output 1

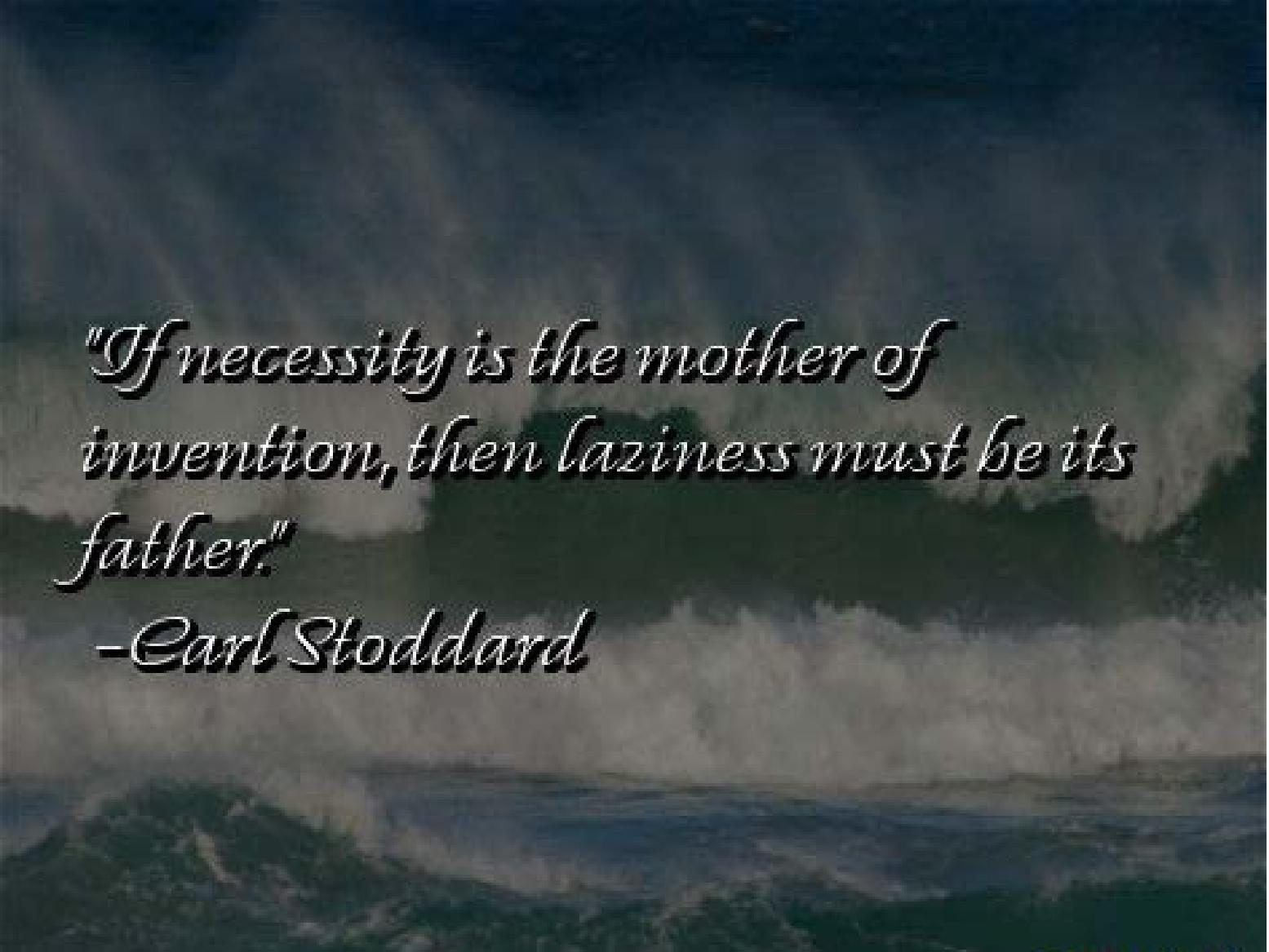


Python Packages

- What if I want to compute $\cos(\pi/2)$?
 - Do I need to write the code for this?

$$\cos(x) = \underbrace{\frac{1}{0!}}_{n=0} - \underbrace{\frac{x^2}{2!}}_{n=1} + \underbrace{\frac{x^4}{4!}}_{n=2} - \underbrace{\frac{x^6}{6!}}_{n=3} + \underbrace{\frac{x^8}{8!}}_{n=4} + \dots$$

- Or draw a graph
- Or implement some network features?
- Or other complicated tasks?



"If necessity is the mother of invention, then laziness must be its father"

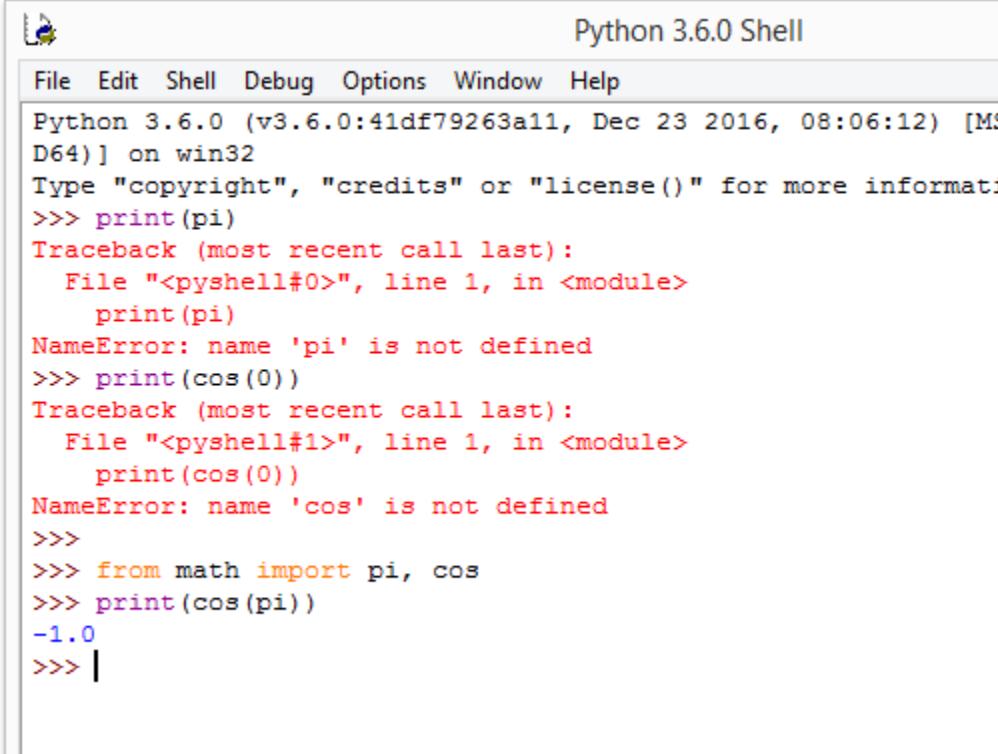
- Earl Stoddard

Different Scenario with Diff. Equipment



Python Packages

- Import the **function** `cos` and the **constant** `pi` from the `math` package
 - (Or library)

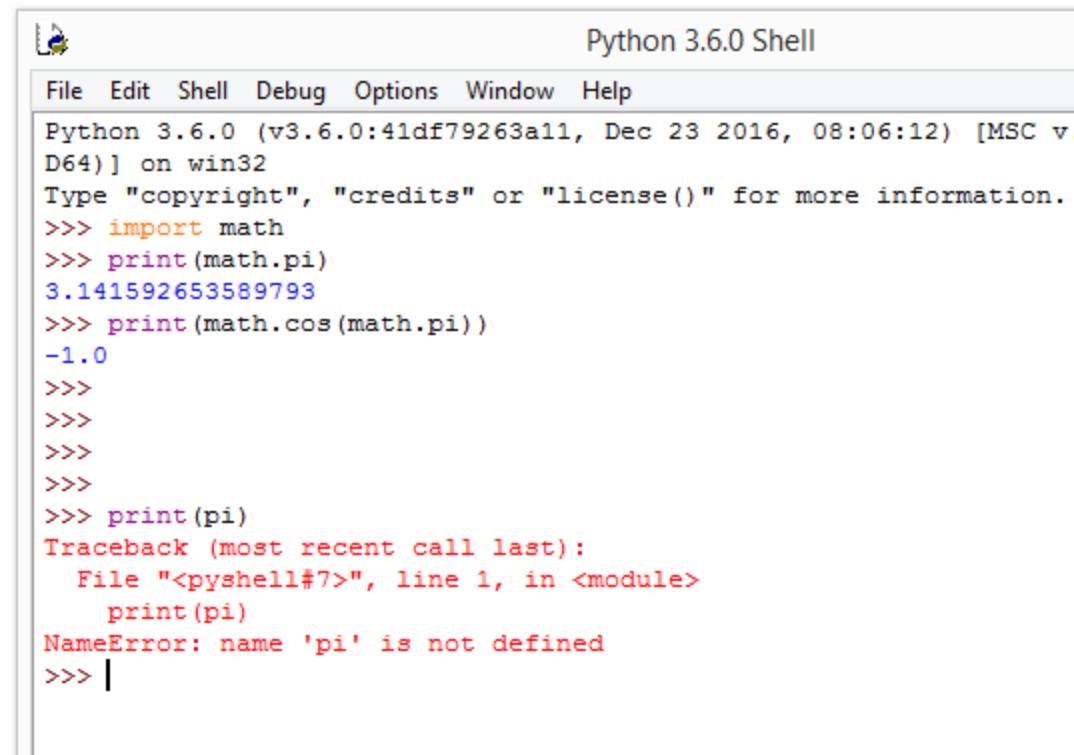


The screenshot shows a Python 3.6.0 Shell window. The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The title bar says "Python 3.6.0 Shell". The shell area displays the following code and its execution:

```
Python 3.6.0 (v3.6.0:41df79263a11, Dec 23 2016, 08:06:12) [MS
D64] on win32
Type "copyright", "credits" or "license()" for more information
>>> print(pi)
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    print(pi)
NameError: name 'pi' is not defined
>>> print(cos(0))
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    print(cos(0))
NameError: name 'cos' is not defined
>>>
>>> from math import pi, cos
>>> print(cos(pi))
-1.0
>>> |
```

The Other way to Import cos

- You need at add the prefix “math.” before the function
- Otherwise..



```
Python 3.6.0 (v3.6.0:41df79263a11, Dec 23 2016, 08:06:12) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> import math
>>> print(math.pi)
3.141592653589793
>>> print(math.cos(math.pi))
-1.0
>>>
>>>
>>>
>>>
>>> print(pi)
Traceback (most recent call last):
  File "<pyshell#7>", line 1, in <module>
    print(pi)
NameError: name 'pi' is not defined
>>> |
```

Which way should I use?

- First, “from math import cos”
 - Make your code shorter
 - Save memory
 - But you need to know exactly which function you have to import
- Second, “import math”
 - You can be more relax to use functions in the whole library
 - But your code will be longer and require more memory

```
>>>
>>> from math import pi, cos
>>> print(cos(pi))
-1.0
>>> |
```

```
>>> import math
>>> print(math.pi)
3.141592653589793
>>> print(math.cos(math.pi))
-1.0
>>> |
```

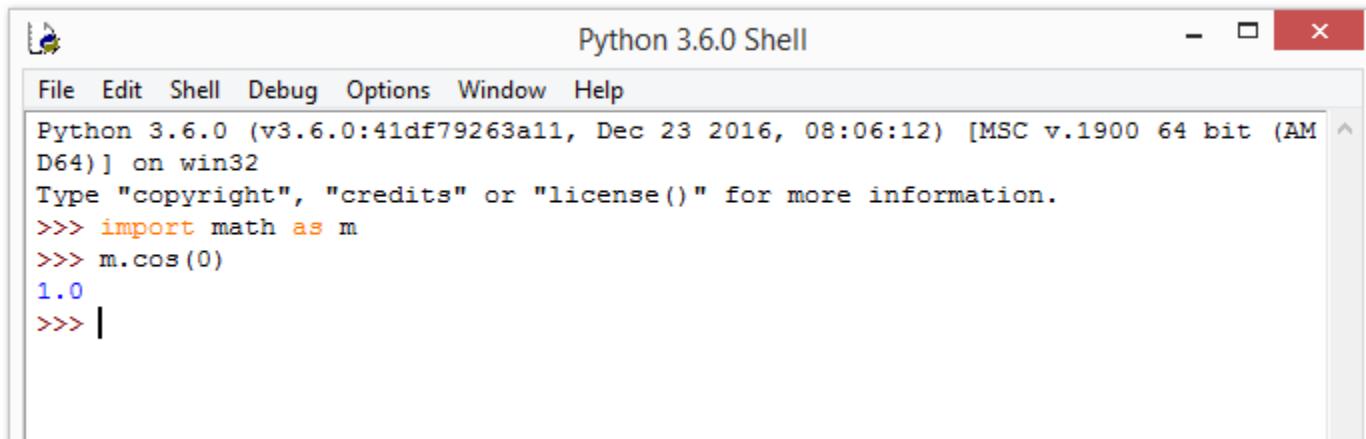
You don't want to be like this in real life



Try NOT to import
too many packages



To Avoid Long Name

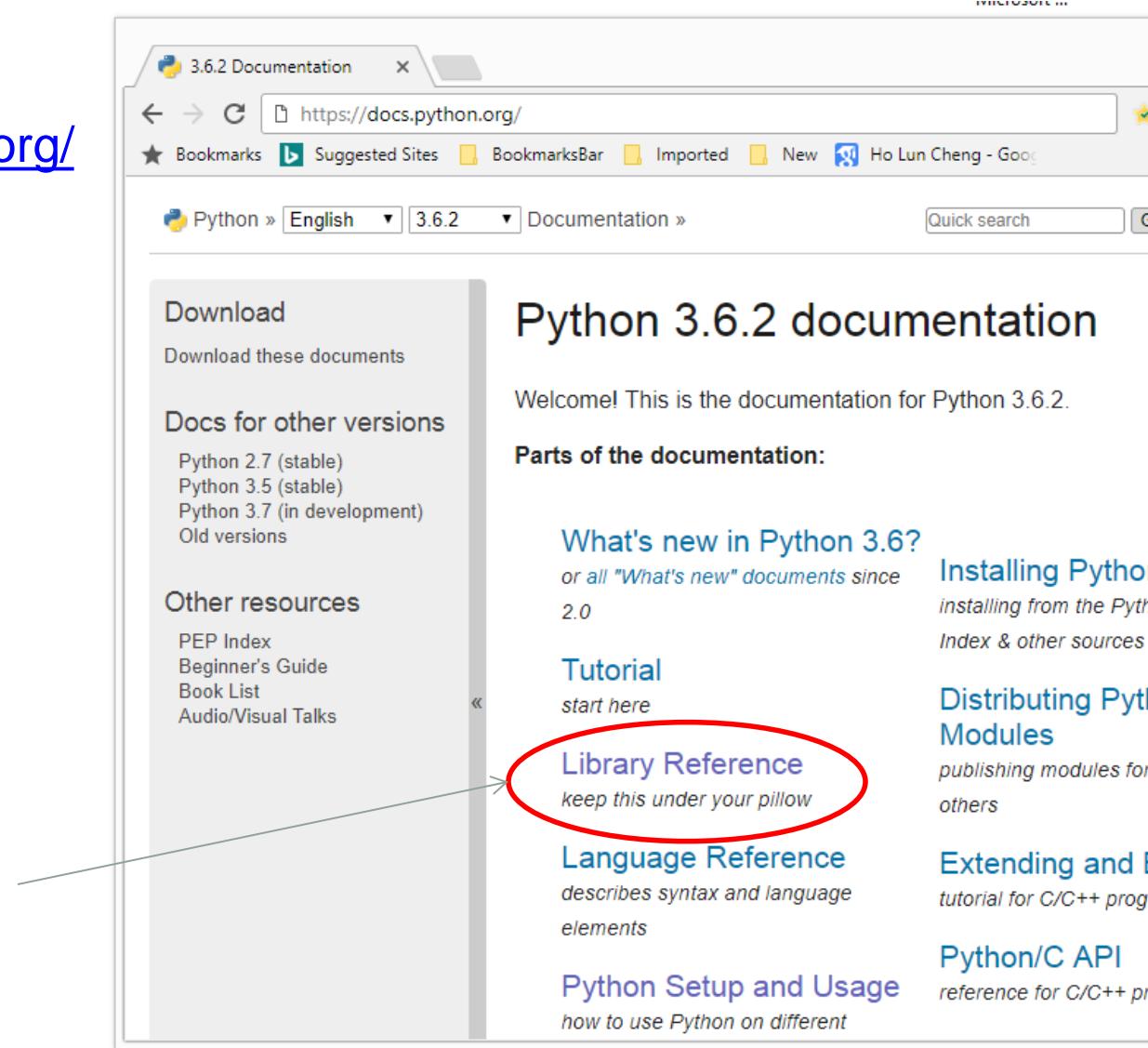


The screenshot shows the Python 3.6.0 Shell window. The title bar reads "Python 3.6.0 Shell". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main window displays the following text:

```
Python 3.6.0 (v3.6.0:41df79263a11, Dec 23 2016, 08:06:12) [MSC v.1900 64 bit (AM  
D64)] on win32  
Type "copyright", "credits" or "license()" for more information.  
>>> import math as m  
>>> m.cos(0)  
1.0  
>>> |
```

For More Packages

- Visit
 - <https://docs.python.org/>



Keep this
under your
pillow

Python Packages

The screenshot shows a Microsoft Edge browser window titled "The Python Standard Lib". The address bar displays "Python Software Foundation [US] | https://docs.python.org/3/library/index.html". The page content lists various Python modules:

- 8.7. array — Efficient arrays of numeric values
- 8.8. weakref — Weak references
- 8.9. types — Dynamic type creation and names for built-in types
- 8.10. copy — Shallow and deep copy operations
- 8.11. pprint — Data pretty printer
- 8.12. reprlib — Alternate repr() implementation
- 8.13. enum — Support for enumerations
- 9. Numeric and Mathematical Modules
 - 9.1. numbers — Numeric abstract base classes
 - 9.2. math — Mathematical functions
 - 9.3. cmath — Mathematical functions for complex numbers
 - 9.4. decimal — Decimal fixed point and floating point arithmetic
 - 9.5. fractions — Rational numbers
 - 9.6. random — Generate pseudo-random numbers
 - 9.7. statistics — Mathematical statistics functions
- 10. Functional Programming Modules
 - 10.1. itertools — Functions creating iterators for efficient looping
 - 10.2. functools — Higher-order functions and operations on callable objects
 - 10.3. operator — Standard operators as functions

Built-in Functions

- Functions that need **NOT** to be imported

Built-in Functions				
<code>abs()</code>	<code>dict()</code>	<code>help()</code>	<code>min()</code>	<code>setattr()</code>
<code>all()</code>	<code>dir()</code>	<code>hex()</code>	<code>next()</code>	<code>slice()</code>
<code>any()</code>	<code>divmod()</code>	<code>id()</code>	<code>object()</code>	<code>sorted()</code>
<code>ascii()</code>	<code>enumerate()</code>	<code>input()</code>	<code>oct()</code>	<code>staticmethod()</code>
<code>bin()</code>	<code>eval()</code>	<code>int()</code>	<code>open()</code>	<code>str()</code>
<code>bool()</code>	<code>exec()</code>	<code>isinstance()</code>	<code>ord()</code>	<code>sum()</code>
<code>bytearray()</code>	<code>filter()</code>	<code>issubclass()</code>	<code>pow()</code>	<code>super()</code>
<code>bytes()</code>	<code>float()</code>	<code>iter()</code>	<code>print()</code>	<code>tuple()</code>
<code>callable()</code>	<code>format()</code>	<code>len()</code>	<code>property()</code>	<code>type()</code>
<code>chr()</code>	<code>frozenset()</code>	<code>list()</code>	<code>range()</code>	<code>vars()</code>
<code>classmethod()</code>	<code>getattr()</code>	<code>locals()</code>	<code>repr()</code>	<code>zip()</code>
<code>compile()</code>	<code>globals()</code>	<code>map()</code>	<code>reversed()</code>	<code>__import__()</code>
<code>complex()</code>	<code>hasattr()</code>	<code>max()</code>	<code>round()</code>	
<code>delattr()</code>	<code>hash()</code>	<code>memoryview()</code>	<code>set()</code>	

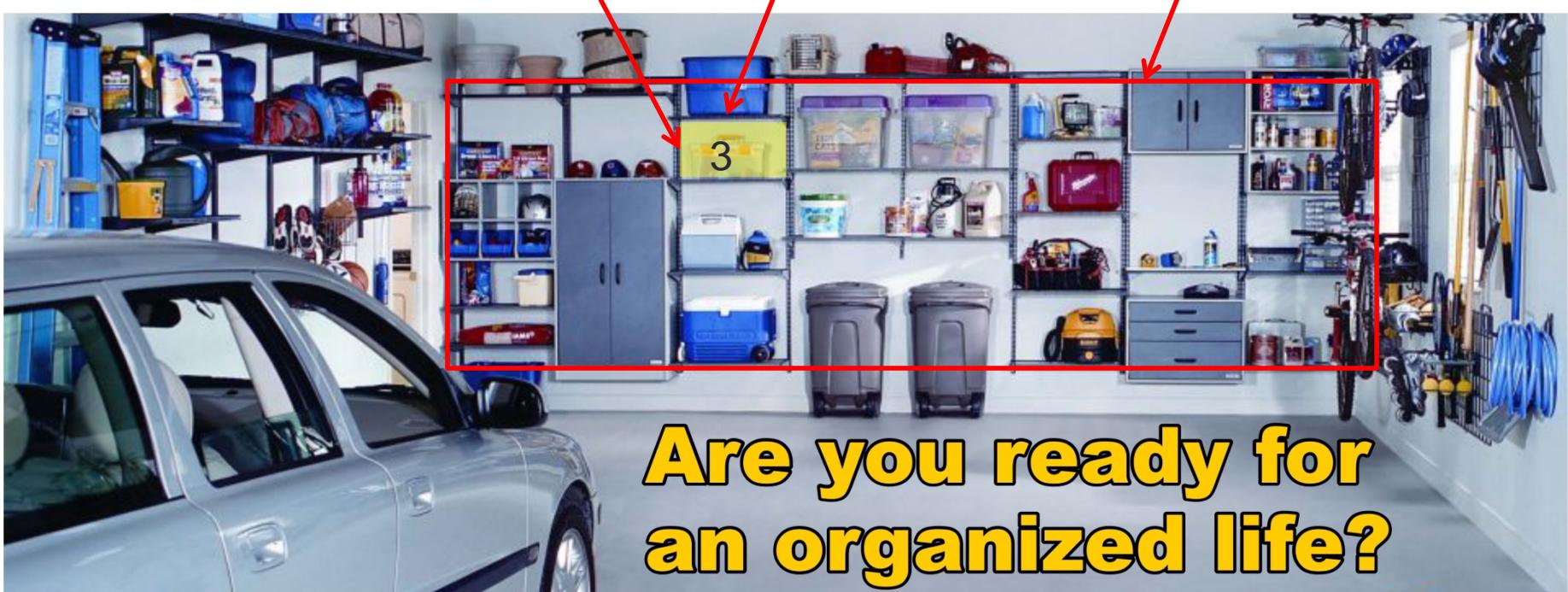
Variables

- For storage of data

```
>>> x = 3
```

This space is allocated and labeled as “x”. And we store ‘3’ inside

Computer
Memory



Variable Naming

- Start with 'a'-'z' or 'A'-'Z' or '_'
- Contain only alphanumeric characters or '_'
- Case sensitive

x_1 != x_1

- Avoid reserved keywords e.g. if
- Python convention: lower case letters separated by '_'
 - e.g. count_change

Data Types

8 , 45 , 123 int

2.71828 , 3.14159 , 1.0 float

True , False bool

"it1007" str
'it1007'

None

Variable Type

```
>>> x = 3
```

```
>>> name = "Alan"
```

This space is allocated and labeled as “x”. And we store ‘3’ inside. And can store **integers** only

This space is allocated and labeled as “name”. And we store “Alan” inside. And can store **strings** only



The function Type(...)

```
>>> type(123)
```

```
<class 'int'>
```

```
>>> type('123')
```

```
<class 'str'>
```

```
>>> type(None)
```

```
<class 'None'>
```

Type conversion

```
>>> str(123)  
'123'
```

```
>>> float('45.2')  
45.2
```

```
>>> int(23.8)  
23
```

```
>>> int('cs1010s')  
ValueError!
```

Assignments

```
>>> abc = 18
```

```
>>> my_string = 'This is my string'
```

```
>>> x, y = 1, 2
```

Doesn't matter if it's quote
or double quote



Assignments

```
>>> x = 10  
>>> x = 2  
>>> x = 4  
>>> print(x)
```

???

```
>>> a, b, c = 1, 2, 3  
>>> a, b, c = c, b, a  
>>> print(a, b, c)  
???
```



Are you an or

“=” is different
from our usual
“equal” in math



Arithmetic: + - * / ** // %

```
>>> a = 2 * 3
```

```
>>> a
```

```
6
```

```
>>> 2 ** 3
```

```
8
```

```
>>> 11 / 3
```

```
3.6666666666666665
```

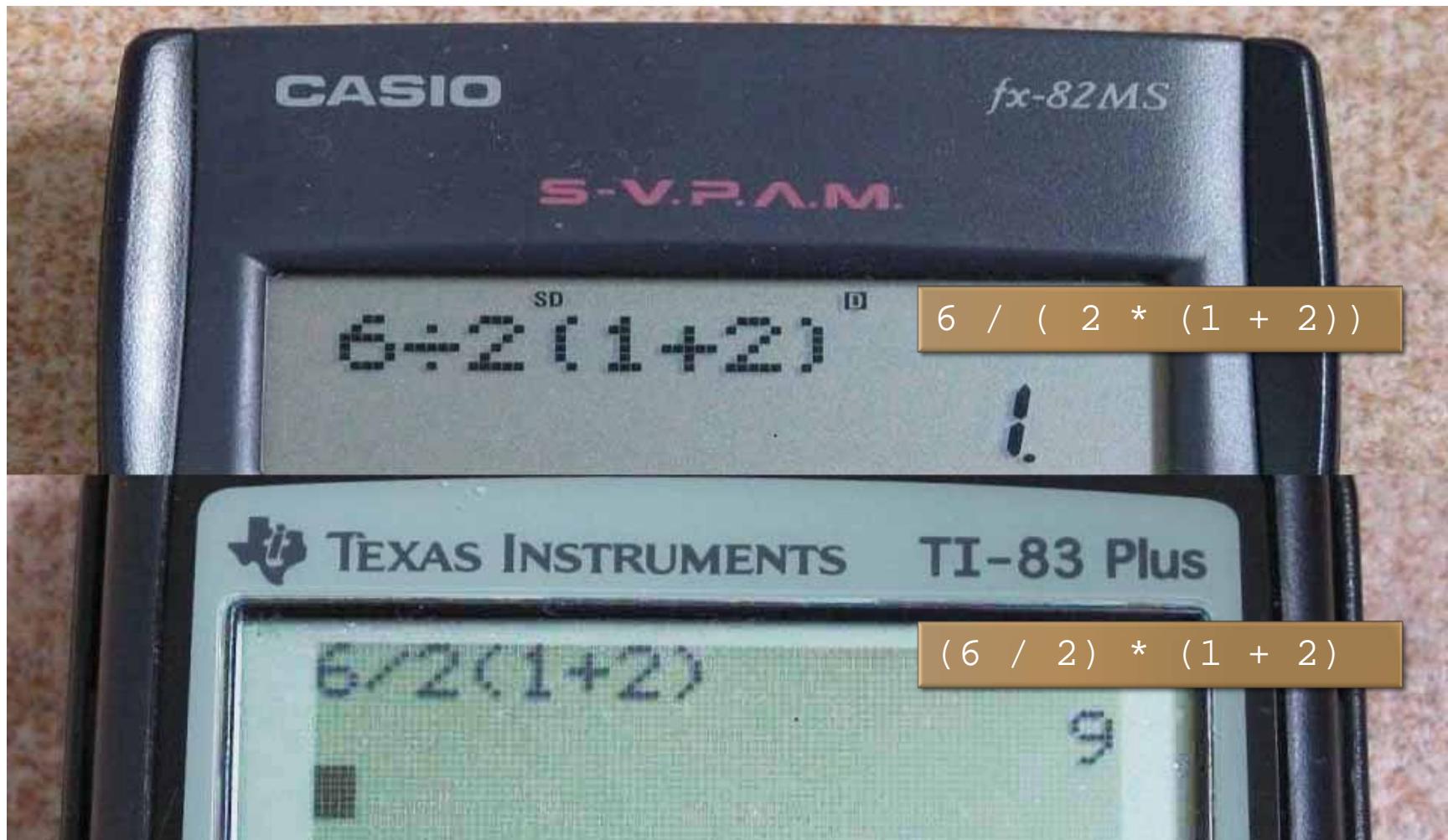
```
>>> 11 // 3
```

```
3
```

```
>>> 11 % 3
```

```
2
```

Operator Precedence



Python Operator Precedence

- $6 / 2 * (1+2)$
- $3 * (1+2)$
- $3 * (1+2)$
- $3 * 3$
- 9

Operator	Description
<code>**</code>	Exponentiation (raise to the power)
<code>~ + -</code>	Complement, unary plus and minus (method names for the last two are <code>+@</code> and <code>-@</code>)
<code>* / % //</code>	Multiply, divide, modulo and floor division
<code>+ -</code>	Addition and subtraction
<code>>> <<</code>	Right and left bitwise shift
<code>&</code>	Bitwise 'AND'>
<code>^ </code>	Bitwise exclusive 'OR' and regular 'OR'
<code><= < > >=</code>	Comparison operators
<code><> == !=</code>	Equality operators
<code>= %= /= //= -=</code> <code>+= *= **=</code>	Assignment operators
<code>is is not</code>	Identity operators
<code>in not in</code>	Membership operators
<code>not or and</code>	Logical operators

Boolean: Truth values

- Statements can be either **True** or **False**
- `2 > 1` is True
- `5 < 3` is False

Operators

- Comparison:

```
>>> 1 <= 10
```

True

```
>>> 5 > 15
```

False

```
>>> 5 <= 5
```

True

```
>>> 2 != 3
```

True

```
>>> '1' == 1
```

False

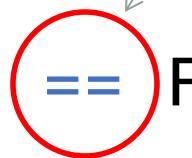
```
>>> False == False
```

True

```
>>> True != True
```

False

The very no. 1
trap for
programmers



Operators

- Logic:

```
>>> True or False
```

```
True
```

```
>>> True and False
```

```
False
```

```
>>> not False
```

```
True
```

- a **or** b True if either **a** or **b** is True
- a **and** b True if both **a** and **b** are True
- **not** a True if a is **not** True

Truth Tables

A	NOT A
True	False
False	True

A OR B		A	
		True	False
B	True	True	True
	False	True	False

A AND B		A	
		True	False
B	True	True	False
	False	False	False

Truth Value Revisited

- Python has keywords `True` and `False`
- In Python 3.x, `True` and `False` will be equal to `1` and `0`
- Anything that is not `0` or `empty` will be evaluated as `True`
- Logic:

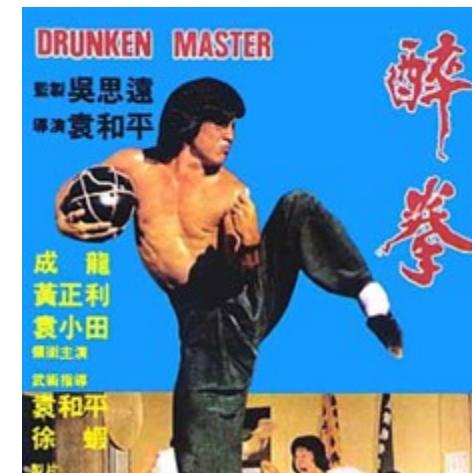
```
>>> True and 0  
0  
>>> not 'abc'  
False  
>>> 1 or 0  
1
```

Today

- Introduction to
 - Programming
 - Psuedocode
 - Flowchart
 - Control Structures
 - Python
- Python Setup guide (Not graded)
 - Lab 00 – Setting Up Pyhton

How to be a good programmer

- What is the different between old and new fighting movies?



Strings

```
>>> s = 'ba'          >>> 'z' in t  
>>> t = 'ck'          False  
>>> s + t            >>> 'bananb' > t  
'back'               True  
>>> t = s + 'na' *    >>> 'banan' <= t  
2                     True  
>>> t                >>> 'c' < t  
'banana'             False
```

lexicographical ordering: first the first two letters are compared, and if they differ this determines the outcome of the comparison; if they are equal, the next two letters are compared, and so on, until either sequence is exhausted.

Strings

```
>>> w = 'banana'          >>> s = (w+' ') * 2  
>>> s = w + w            >>> print(s)  
>>> print(s)              'banana banana '  
'bananabanana'  
>>> s = w*3  
>>> print(s)  
'bananabananabana'
```

Strings

- A String is a sequence of characters
- We can index a string, i.e.

```
>>> s = 'abcd'
```

```
>>> s[0]
```

```
'a'
```

```
>>> s[2]
```

```
'c'
```

- The index of the first character is 0

String Slicing (Leave it to next slide)

Non-inclusive

`s[start:stop:step]`

```
>>> s = 'abcdef'
```

```
>>> s[0:2]
```

```
'ab'
```

```
>>> s[1:2]
```

```
'b'
```

```
>>> s[:2]
```

```
'ab'
```

```
>>> s[1:5:3]
```

```
'be'
```

```
>>> s[::-2]
```

```
'ace'
```

```
>>> s[::-1]
```

```
???
```

Default
start = 0
stop = #letters
step = 1

Strings

```
c = "#"  
s = " "  
  
print(" ") *  
print(" *") #  
print(s*3 + c) ####  
print(s*2 + c * 3) #####  
print(s + c * 5) #  
print(s*3 + c) ####  
print(s*2 + c * 3) #####  
print(s + c * 5) #####  
print(c * 7) #  
print(s*3 + c)
```

ASCII Art

$$\begin{array}{c}
 (\overline{\text{C}}) . - . (\overline{\text{C}}) \\
 / \quad \cdot \underline{\text{Y}} \cdot \backslash \\
 \underline{\text{X}} \backslash (\overline{\text{Y}}) / \underline{\text{X}} \\
 (\underline{\text{X}} . - / ' - ' \backslash - \cdot \underline{\text{X}}) \\
 | \quad | \quad | \quad | \quad | \\
 (\underline{\text{X}} . - . / ' - ' \backslash - \cdot \underline{\text{X}})
 \end{array}$$