CSE4IP-CSE5CES
Lecture 1 - Introduction to Programming

Ayad Turky

Your friendly lecturer – Ayad Turky

- BSc (Hons) in Computer Science from Anbar University
- MSc in Computer Science (Artificial Intelligence) from UKM University
- PhD in Computer Science (Artificial Intelligence) from RMIT University
- email: A.Turky@latrobe.edu.au

Your subject coordinator – Nasser Sabar

- Dr Nasser Sabar is a Lecturer in Data Science at the Department of Computer Science and Information Technology, at La Trobe University in Melbourne, Australia.
- email: n.sabar@latrobe.edu.au

Subject Materials and Announcements

- **You may find everything you need on LMS!**
  - Subject materials (e.g. lecture notes, lab instructions, assignment questions) and other announcements will be posted to LMS.
  - Please check the LMS subject page as often as possible!

- **Recommended readings**
  - Gaddis, T. (2018). Starting Out with Python, Global Edition. (4th ed.).
  - Horstmann, C., & Necaise, Rance D. (2019). Python for everyone (Third ed.).

Learning Objectives

- **Upon the completion of the subject, you are expected to**

1. **Apply the programming-based problem-solving approach to analyse the requirements of a given problem and design and incrementally implement a solution.**
2. **Apply basic programming constructs of sequence, selection and iteration to explore, test and implement algorithm logic to solve problems.**
3. **Construct and execute test cases as part of the well-planned incremental software development approach.**
4. **Apply the concept of function to modularise programs and enable unit testing.**
5. **Use file input/output techniques to design data files, to enable programs store data in files and to read data from files.**
6. **Apply the object-oriented concepts and techniques to represent real-world objects in an application domain and use**

Policies & Grading

Assessment Description

Assignment-1	25%
Assignment-2	25%
Final Exam	50%

- Please refer to the **Special Consideration** Policy, if necessary.
  - https://policies.latrobe.edu.au/document/view.php?id=205

?

Click icon to add picture

QUESTION: What is your course?

Click icon to add picture

QUESTION: Why are you in this subject?

Click icon to add picture

?

QUESTION: What are your goals for this subject?

Click icon to add picture

latrobe.edu.au

?

QUESTION: What questions do you have for me?

?

Click icon to add picture

QUESTION: Are computers smart?

Q: Are computers smart?
A: Not really.

- By and large, computers can only do *exactly* what someone (a “programmer”) tells them to do
  - And not only that, you have to tell them exactly what to do in a *special language*, because they don’t understand English (or Mandarin, or Hindi, or German…)
    - Human language is too complicated!

- If computers were actually smart, we wouldn’t need computer programmers; you could just ask a question and get an answer

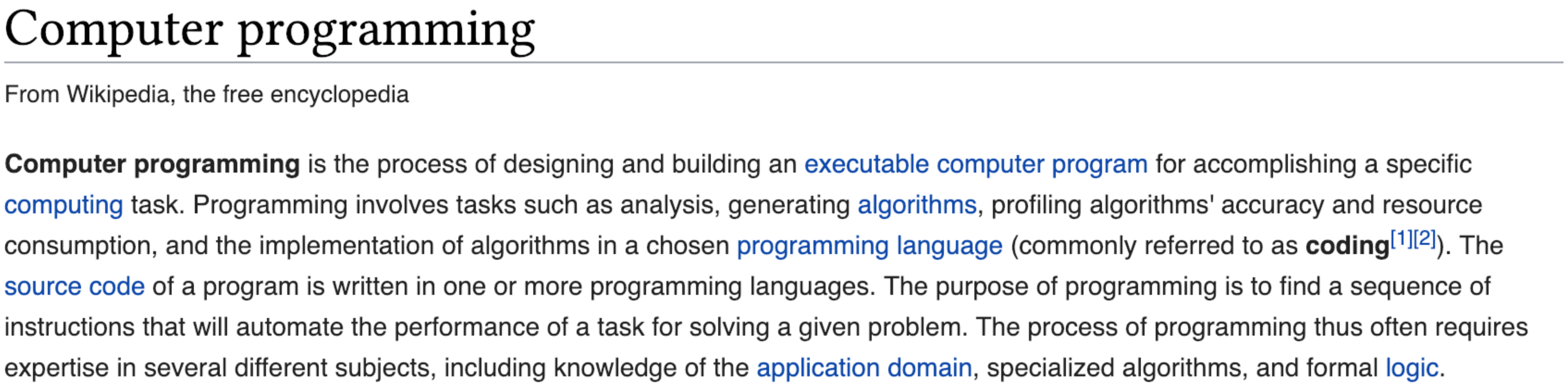
The “Peanut Butter and Jelly Prol

Sometimes, me
to my computer.
Other times, my
computer to me.

https://www.youtube.com/watch?v=cDA3_5982h8

What is “computer programming”?

- Formally:


Computer programming

From Wikipedia, the free encyclopedia

Computer programming is the process of designing and building an executable computer program for accomplishing a specific computing task. Programming involves tasks such as analysis, generating algorithms, profiling algorithms' accuracy and resource consumption, and the implementation of algorithms in a chosen programming language (commonly referred to as **coding**[1][2]). The source code of a program is written in one or more programming languages. The purpose of programming is to find a sequence of instructions that will automate the performance of a task for solving a given problem. The process of programming thus often requires expertise in several different subjects, including knowledge of the application domain, specialized algorithms, and formal logic.

- Informally: Telling a computer what you want it to do in a language that it can understand

So what do we mean by “a language that a computer understands?”

- Unfortunately, the answer to this is more complicated than it seems

- In order to answer this question, we need to talk a little bit about what computers are made of

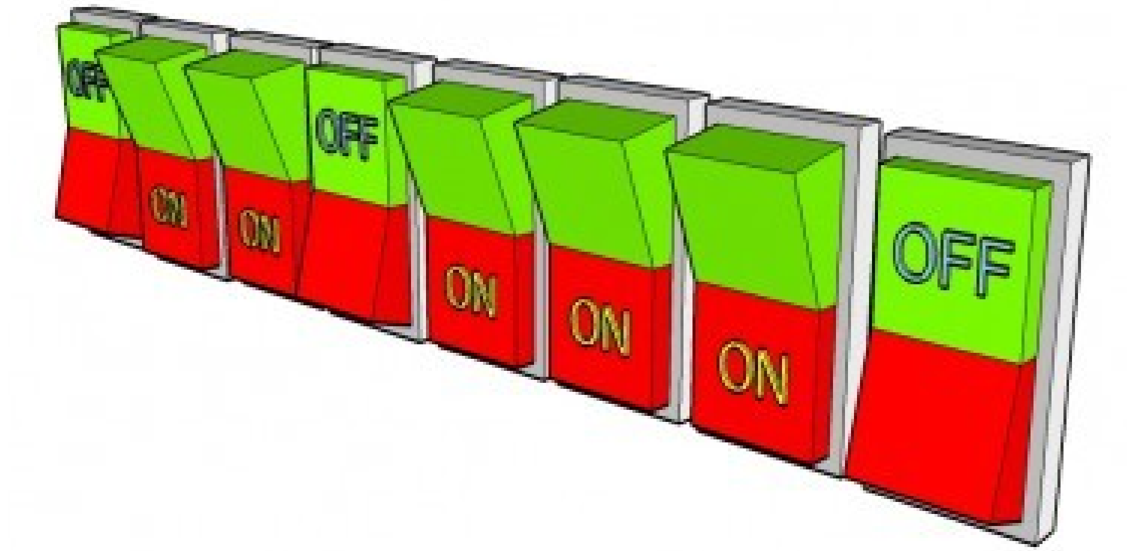
Transistors and Bits

- Modern computers contain millions or billions of *transistors*
- The full answer is a bit more complicated than this, but you can think of a transistor as a tiny switch
  - – Each can be in one of two states, on or off
- One transistor (or one switch) can be used to store 1 *bit* of data
  - – *Bit* is short for "Binary digIT"
    - Each bit can hold one of two values, 0 or 1 (off or on)

Discrete (that is, self-contained) transistors. Modern *integrated circuits* (ICs) have millions of transistors in a single chip.

Bits and Bytes

- A *byte* is 8 bits of data that are operated on together



A conceptual model of one byte. This “byte” has the value 01101110, which is 110 in decimal and 6E in hexadecimal.

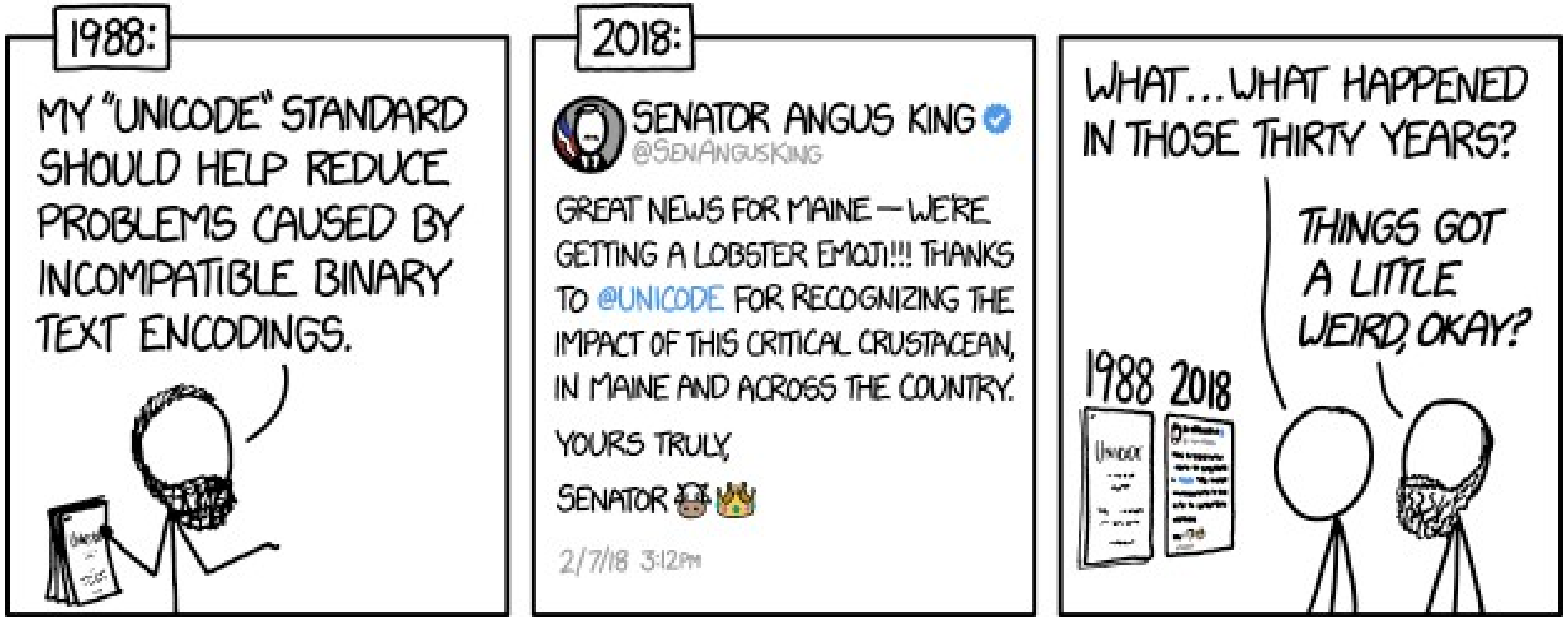
Why 8 bits in a byte?

- Historically, 8 bits were needed to store a single alphanumeric character
  - A group of 8 bits can distinguish between $2^8 = 256$ different values
  - This decision was made for the IBM System/360 mainframe in 1964, and it stuck
- This assumes, among other things, that everyone is writing in English
  - Chinese languages, for example, have far more than 256 characters

DIGRESSION: From ASCII to Unicode

- The 8-bit character standard (also known as ASCII) is still the default in many programming languages, including Python
- Eventually, the people writing software realized that there were languages other than English
  - For that reason, the Unicode standard was developed, which uses at least 2 bytes (16 bits = 2^{16} = 65536 different values), but up to 4 bytes (32 bits => 2^{32} ~ 4.3 billion different values) for each character.
    - 4 bytes are more than sufficient to encode every character in every language ever used on Earth

DIGRESSION: Unicode

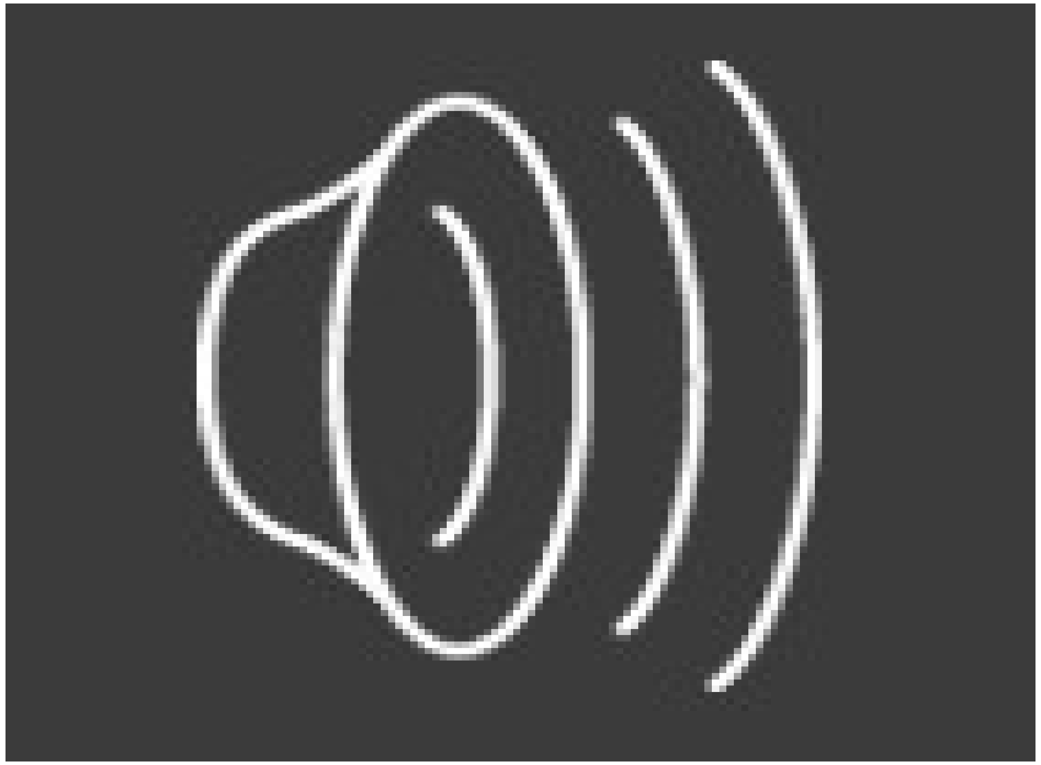

1988:
MY "UNICODE" STANDARD SHOULD HELP REDUCE PROBLEMS CAUSED BY INCOMPATIBLE BINARY TEXT ENCODINGS.
2018:
SENATOR ANGUS KING
@SENANGUSKING
GREAT NEWS FOR MAINE — WE'RE GETTING A LOBSTER EMOJI!!! THANKS TO @UNICODE FOR RECOGNIZING THE IMPACT OF THIS CRITICAL CRUSTACEAN, IN MAINE AND ACROSS THE COUNTRY.
YOURS TRULY,
SENATOR 🐮👑
2/7/18 3:12PM
WHAT... WHAT HAPPENED IN THOSE THIRTY YEARS?
THINGS GOT A LITTLE WEIRD, OKAY?
1988 2018
UNICODE

https://xkcd.com/1953/

Machine Language

- So, at the deepest level, “the language a computer understands” is usually a sequence of bytes, each containing 8 bits
  - I.e. 01010001 11001110 11110000 00101001…
    - This is also referred to as *machine language*

Machine Language



$$1010011010 =$$
$$1*2^9 + 0*2^8 + 1*2^7 + 0*2^6 + 0*2^5 + 1*2^4 + 1*2^3 + 0*2^2 + 1*2^1 + 0*2^0 =$$
$$512 + 128 + 16 + 8 + 0 + 2 + 0 =$$

666

Assembly Language

- Unfortunately, while robots might be able to understand machine language, it is basically impossible for humans to work with
- So, very early on (The first assembly language was developed in 1947 by Kathleen Booth), "assembly languages" were developed
  - These are languages that are human-interpretable, but have basically a 1-to-1 correspondence with a computer's machine language instructions

Assembly Language

- So, instead of

```
10110000 01100001
```

(which, in the assembly language of the Intel 8086 processors, means “move the value 97 into the memory register named AL”), we might write

```
MOV AL, 61h ; Load AL with 97 decimal (61 hex)
```

“High-Level” Languages

- Assembly language is a significant improvement on machine language, but it’s still very tedious for humans to understand and use
  - For that reason, so-called “high-level” programming languages have been developed
    - High-level languages include pretty much any language you might have heard of, such as C, C++, Java, **Python**, etc.
  - High-level can refer to many language features, but it mainly means a higher level of *abstraction*

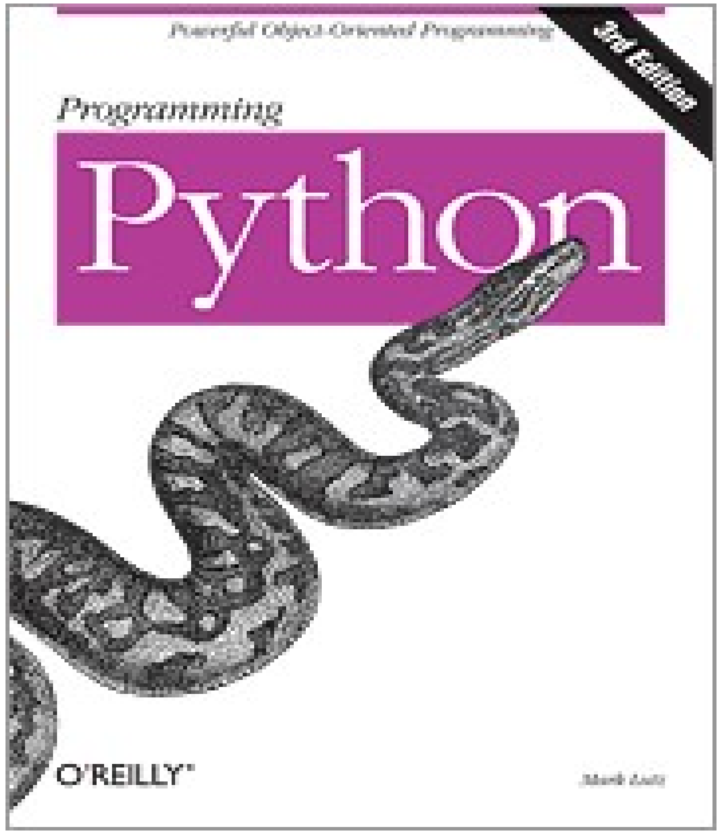
Abstraction

- Abstraction is a fundamental concept in computer science
  - In short, abstraction is the hiding of “irrelevant” details
    - “Irrelevant” is in the eye of the beholder, but we’ll gloss over that for now
- Some examples:
  - Do I, as a programmer, care whether a value is stored in physical register AL or BH?
    - Usually not
  - Do I care whether my program will run on an x86 processor or an ARM processor?
    - Probably not

Benefits of high-level languages

- Generally, much easier to read and write
  - Much closer to “human” language
- Hide difficult or tedious tasks, like register management, from the programmer
- They are abstracted from the physical hardware (unlike machine or assembly code), so code written in a HLL can run on a variety of systems
  - How? Via a special kind of program called a *compiler*

Python


Powerful Object-Oriented Programming
3rd Edition
Programming
Python
O'REILLY
Mark Lutz

Python

- The specific programming language that we will use in this subject is called Python
- Python dates back to 1991
- Python is an interpreted, high-level, general-purpose programming language.
- Python 2.0, released in 2000, introduced features like list and a garbage collection system with reference counting.
- Python 3.0, released in 2008, was a major revision of the language that is not completely backward-compatible, and much Python 2 code does not run unmodified on Python 3.

Python

- Python is a great language to learn programming

Why?

- All programming languages have advantages and disadvantages, so you might ask, why have we chosen Python as the language for this subject?
  - There are a lot of good reasons, some very general, some specific to science and engineering, and some specific to La Trobe

Why Python?

- Python is a great language to learn programming

  - It is easy to use
  - It is powerful
  - It provides an interactive environment
    - Enter a statement and Python responds immediately
    - Great to learn concepts and to experiment with ideas

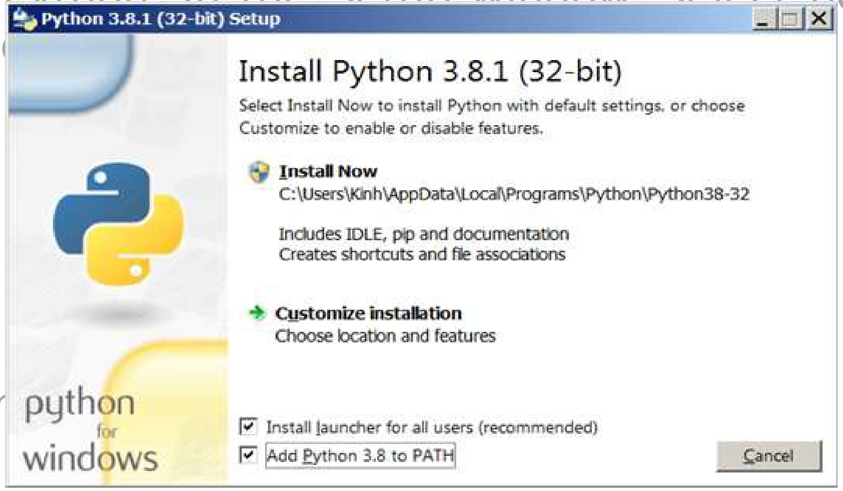
Download and install Python

- Before you can run Python programs on your computer, you need to download and install Python.
- Here is a description on how to install Python on Windows operating system. For Mac and Linux, you can follow the instructions on the Python download page.

Download and install Python

Download and install Python

- ✓ Go to the Python download page: http://python.org/download
- ✓ Click button Download Python 3.8.1 (or whatever the latest version is, depending on the date of downloading). Check that the download file (e.g. python-3.8.1.exe) is displayed at the bottom of the browser. Drag this file to a directory on your PC to save it.
- ✓ Double-click on the installation file to install Python. In the Setup window, select both options provided (Install Launcher for all users and Ad



Python 3.8.1 (32-bit) Setup

Install Python 3.8.1 (32-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

Install Now
C:\Users\Kinh\AppData\Local\Programs\Python\Python38-32

Includes IDLE, pip and documentation
Creates shortcuts and file associations

Customize installation
Choose location and features

☑ Install launcher for all users (recommended)
☑ Add Python 3.8 to PATH

- ✓ The installer will then complete the installatio

Testing your installation

Testing your installation

✓ **Open and work with the interactive shell:**

- From the command prompt, enter command python
- You should see the interactive shell open.
- At the interactive shell command (>>>), enter, for example, expression 10 +20. You should see 30 displayed.
- Enter command exit() to leave the shell.

Access and Use the IDLE editor

- In our subject, we will use IDLE as the preferred editor (for reasons explained in the next chapter). If all go well if the installation, IDLE should be readily available. To verify this:
  - Right-click on a Python program file with .py extension. Select option Open With, and you should see idle.bat as an option to open the file.
    - If you don't see idle.bat, it is likely that you have installed some version of Python before, and the registry for idle.bat points to a file that no longer exists. In this case, you have to edit the registry entry and let it point to the new idle.bat. Be careful with this operation.
  - Select idle.bat to open the file.

Access and Use the IDLE editor

- Edit the file.
- Run the program by choosing menu option Run > Run Module. You may be asked to save the program. The interactive shell (a window) will be opened and the program executed. Type exit() or simply close the window to leave the interactive shell.

Note: When IDLE is routinely used, some may prefer to specify idle.bat as the default program to open .py files. Then we can open a .py file with IDLE by double-clicking on the file.

Working with Python

Use Python in interactive mode

Talking to Python using the interactive mode

Talking to Python using the interactive mode

```
Microsoft Windows [Version 10.0.18363.959]

(c) 2019 Microsoft Corporation

C:\Users\Ayad Turky>
```

Talking to Python using the interactive mode

```
Microsoft Windows [Version 10.0.18363.959]
(c) 2019 Microsoft Corporation. All rights reserved.
C:\Users\Ayad Turky>python
Python 3.7.6 (default, Jan  8 2020, 20:23:39) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

What next?

Talking to Python using the interactive mode

```
Microsoft Windows [Version 10.0.18363.959]

(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Ayad Turky>python

Python 3.7.6 (default, Jan  8 2020, 20:23:39) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32

Type "help", "copyright", "credits" or "license" for more information.

>>> x = 1
>>> print(x)
1
>>> x = x + 1
>>> print(x)
2
>>> exit()
```

This is a good test to make sure that you have Python correctly installed. Note that quit() also works to end the interactive session.

Python Scripts (Script mode)

- **Python Interactive Mode** is good for experiments and programs of 3-4 lines long.
- Most programs are much longer, so we type them into a file and tell Python to run the commands in the file.
- In a sense, we are “giving Python a script”.
- As a convention, we add “.py” as the suffix on the end of these files to indicate they contain Python.

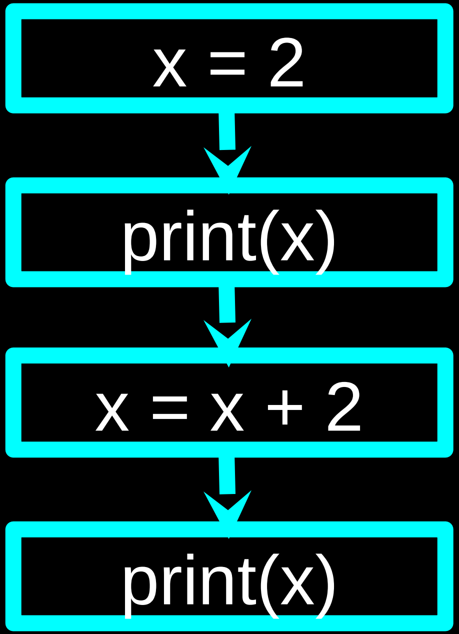
Interactive versus Script

- Interactive
  - You type directly to Python one line at a time and it responds
- Script
  - You enter a sequence of statements (lines) into a file using a text editor and tell Python to execute the statements in the file

Program Steps or Program Flow

- Like a recipe or installation instructions, a program is a sequence of steps to be done in order.
- Some steps are conditional - they may be skipped.
- Sometimes a step or group of steps is to be repeated.
- Sometimes we store a set of steps to be used over and over as needed several places throughout the program.

Sequential Steps



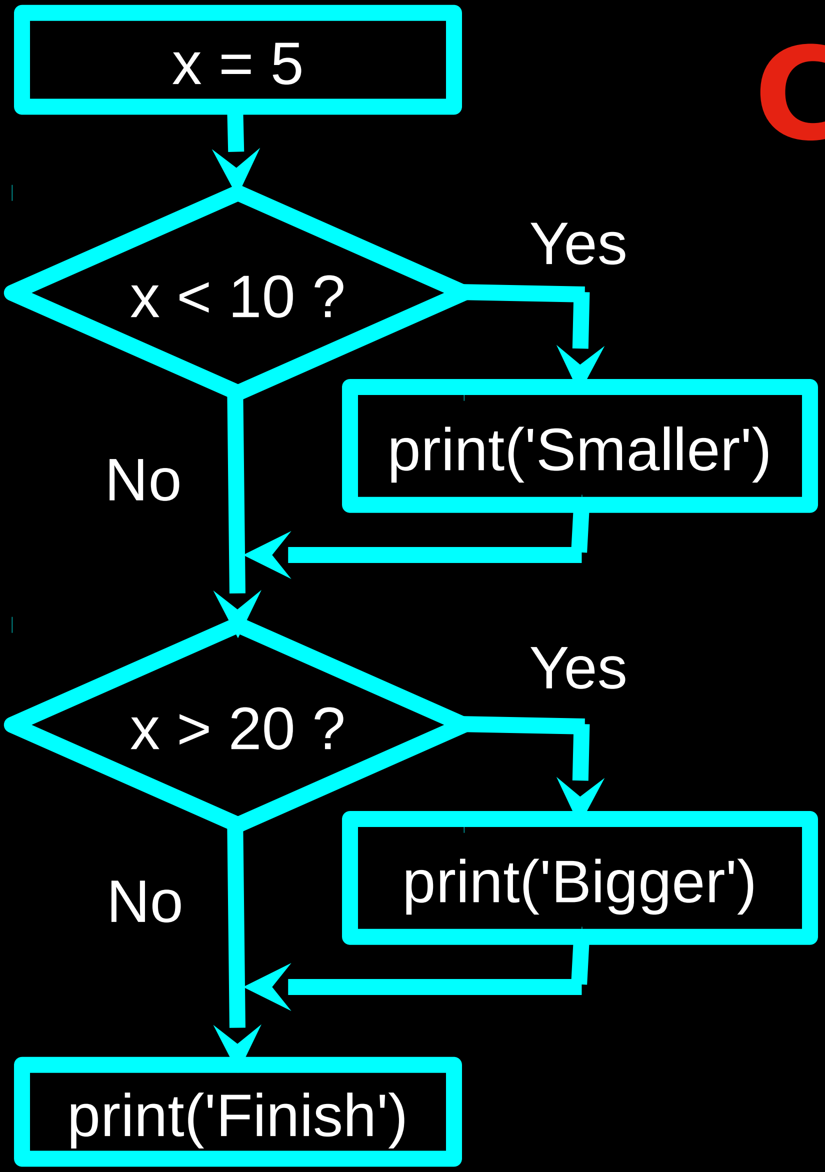
Program:

Output:

```
x = 2
print(x)        2
x = x + 2       4
print(x)
```

When a program is running, it flows from one step to the next. As programmers, we set up "paths" for the program to follow.

Conditional Steps



Program:

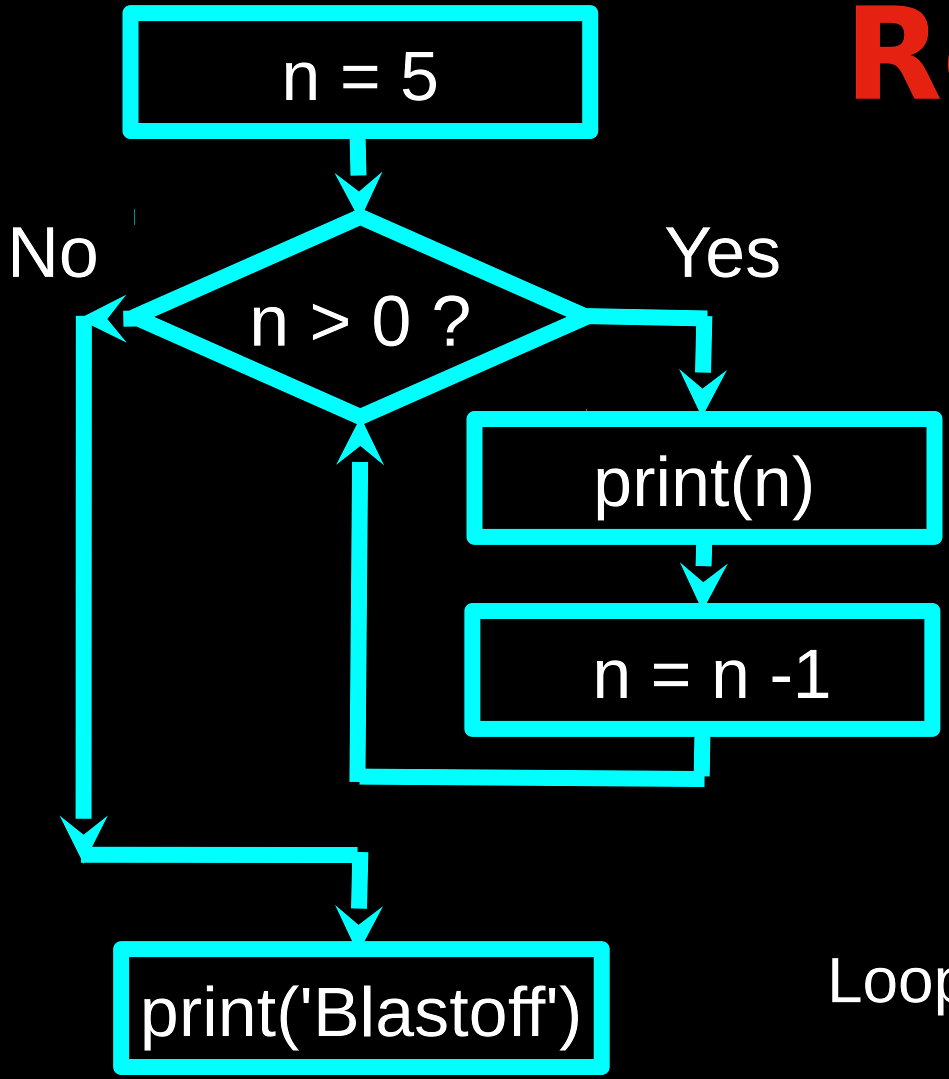
```
x = 5
if x < 10:
    print('Smaller')
if x > 20:
    print('Bigger')

print('Finish')
```

Output:

```
Smaller
Finish
```

Repeated Steps



Program:

```
n = 5
while n > 0 :
    print(n)
    n = n – 1
print('Blastoff!')
```

Output:

```
5
4
3
2
1
Blastoff!
```

Loops (repeated steps) have iteration variables that change each time through a loop.

```
name = input('Enter file:')
handle = open(name, 'r')

counts = dict()
for line in handle:
    words = line.split()
    for word in words:
        counts[word] = counts.get(word,0) + 1

bigcount = None
bigword = None
for word,count in counts.items():
    if bigcount is None or count > bigcount:
        bigword = word
        bigcount = count

print(bigword, bigcount)
```

Sequential

Repeated

Conditional

```
name = input('Enter file:')
handle = open(name, 'r')

counts = dict()
for line in handle:
    words = line.split()
    for word in words:
        counts[word] = counts.get(word,0) + 1

bigcount = None
bigword = None
for word,count in counts.items():
    if bigcount is None or count > bigcount:
        bigword = word
        bigcount = count

print(bigword, bigcount)
```

A short Python "Story" about how to count words in a file

A word used to read data from a user

A sentence about updating one of the many counts

A paragraph about how to find the largest item in a list

Recap

- Computers store data in the form of bits (0 or 1) and bytes (sets of 8 bits)
- For the most part, computers do exactly what a programmer tells them
- Computers only understand a special language of 1s and 0s
  - We call this *machine language* or *machine code*
- One level up from machine language is *assembly language*
  - Still dependent on the specific hardware
- Almost all actual programming is done in *high-level languages*

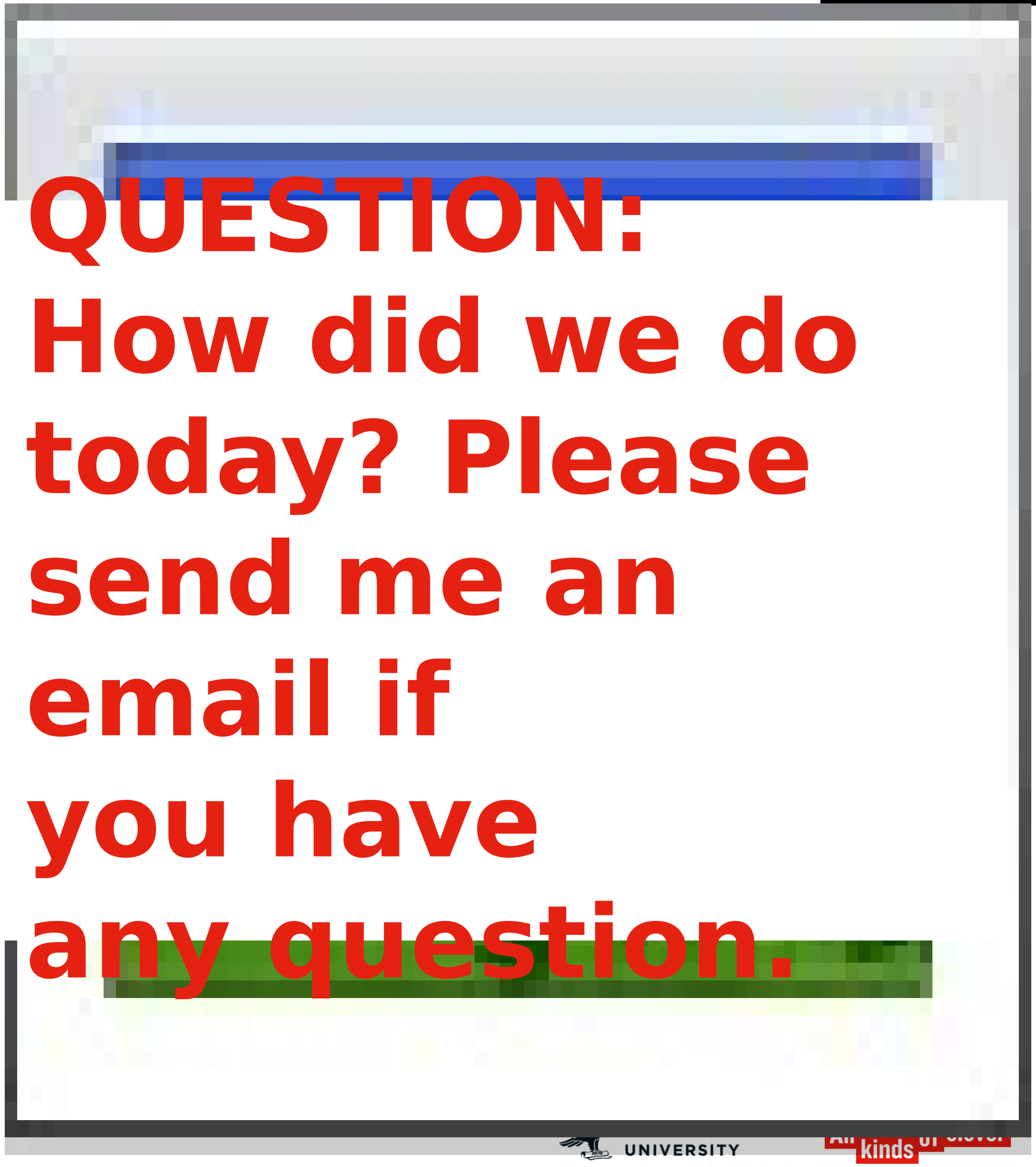
Acknowledgements

- These slides are based on those from Rick Skarbez (CSE1PES/CSE5CES).
- Also acknowledgement to https://www.py4e.com/

Click icon to add picture

latrobe.edu.au

?



QUESTION:
How did we do today? Please send me an email if you have any question.

NEXT LECTURE: “Hello, world”, Variables, Numbers and Strings

LA TROBE UNIVERSITY

Thank you.

Be well.

latrobe.edu.au