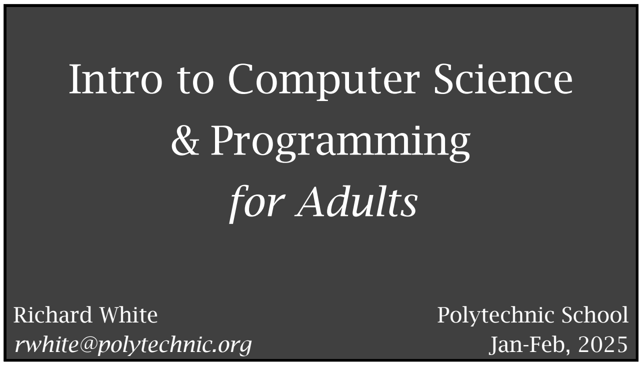
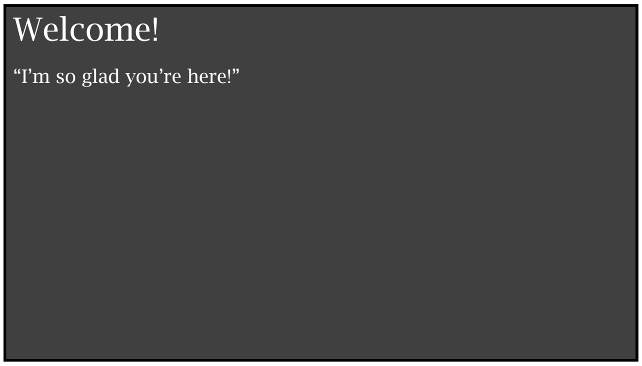


Intro to Computer Science & Programming *for Adults*

Richard White
rwhite@polytechnic.org

Polytechnic School
Jan-Feb, 2025

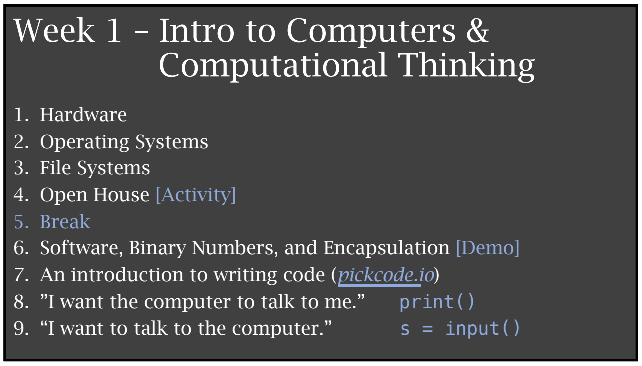
1



Welcome!

"I'm so glad you're here!"

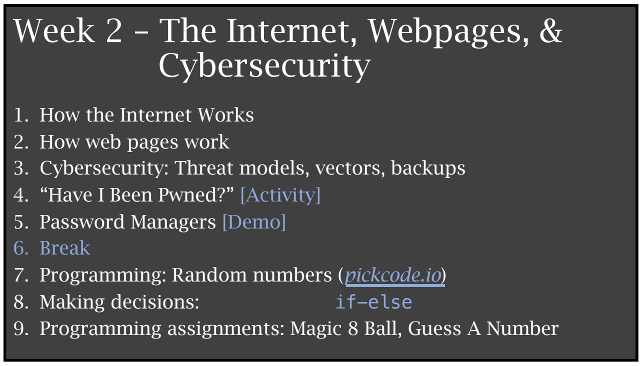
2



Week 1 – Intro to Computers & Computational Thinking

1. Hardware
2. Operating Systems
3. File Systems
4. Open House [Activity]
5. Break
6. Software, Binary Numbers, and Encapsulation [Demo]
7. An introduction to writing code (*pickcode.io*)
8. "I want the computer to talk to me." `print()`
9. "I want to talk to the computer." `s = input()`

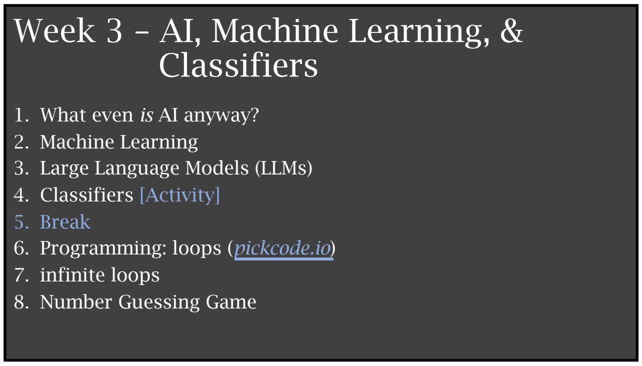
3



Week 2 – The Internet, Webpages, & Cybersecurity

1. How the Internet Works
2. How web pages work
3. Cybersecurity: Threat models, vectors, backups
4. "Have I Been Pwned?" [Activity]
5. Password Managers [Demo]
6. Break
7. Programming: Random numbers (*pickcode.io*)
8. Making decisions: `if-else`
9. Programming assignments: Magic 8 Ball, Guess A Number

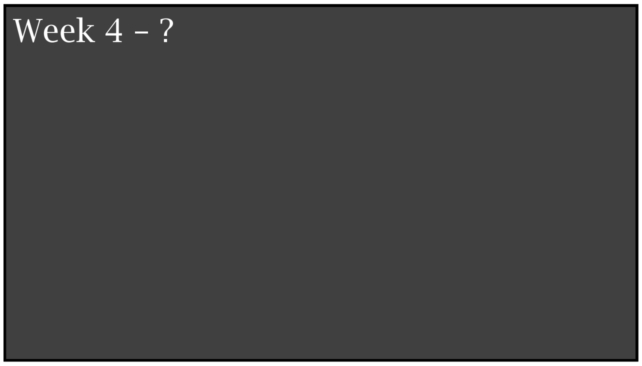
4



Week 3 – AI, Machine Learning, & Classifiers

1. What even *is* AI anyway?
2. Machine Learning
3. Large Language Models (LLMs)
4. Classifiers [Activity]
5. Break
6. Programming: loops (*pickcode.io*)
7. infinite loops
8. Number Guessing Game

5



Week 4 – ?

6

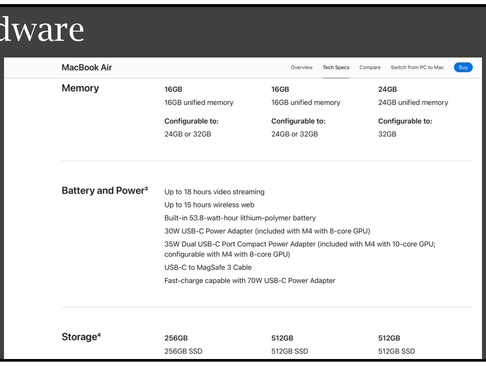
Hardware

The electronic components that make up a typical computer include:

- a Central Processing Unit (CPU)
- memory – Random Access (RAM, 2GigaBytes, 16GB, 64GB)
- memory – secondary storage (HD or SSD, 256GB RAM? 2TB RAM?)
- network – internet, wired or wireless
- peripherals
  - output: monitor, printer, speakers
  - input: keyboard, mouse, microphone

7

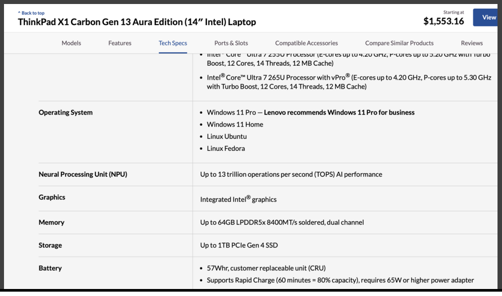
Hardware



Memory	16GB 16GB unified memory Configurable to: 24GB or 32GB	16GB 16GB unified memory Configurable to: 24GB or 32GB	24GB 24GB unified memory Configurable to: 32GB
Battery and Power[3]	Up to 18 hours video streaming Up to 15 hours wireless web Built-in 53.8-watt-hour lithium-polymer battery 30W USB-C Power Adapter (included with M4 with 8-core GPU) 35W Dual USB-C Port Compact Power Adapter (included with M4 with 10-core GPU; configurable with M4 with 8-core GPU) USB-C to MagSafe 3 Cable Fast-charge capable with 70W USB-C Power Adapter		
Storage[4]	256GB 256GB SSD	512GB 512GB SSD	512GB 512GB SSD

8

Hardware



ThinkPad X1 Carbon Gen 13 Aura Edition (14″ Intel) Laptop

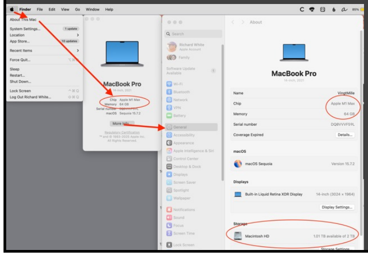
Starting at $1,553.16

	• Intel® Core™ Ultra 7 255U Processor (E-cores up to 4.20 GHz, P-cores up to 5.20 GHz with Turbo Boost, 12 Cores, 14 Threads, 12 MB Cache) • Intel® Core™ Ultra 7 265U Processor with vPro® (E-cores up to 4.20 GHz, P-cores up to 5.30 GHz with Turbo Boost, 12 Cores, 14 Threads, 12 MB Cache)
Operating System	• Windows 11 Pro — **Lenovo recommends Windows 11 Pro for business** • Windows 11 Home • Linux Ubuntu • Linux Fedora
Neural Processing Unit (NPU)	Up to 13 trillion operations per second (TOPS) AI performance
Graphics	Integrated Intel® graphics
Memory	Up to 64GB LPDDR5x 8400MT/s soldered, dual channel
Storage	Up to 1TB PCIe Gen 4 SSD
Battery	• 57Whr, customer replaceable unit (CRU) • Supports Rapid Charge (60 minutes = 80% capacity), requires 65W or higher power adapter

9

Hardware [Activity]

Identify your CPU, your RAM, and your Disk Drive for your computer.



MacBook Pro
14-inch, 2021

Chip Apple M1 Max
Memory 64 GB
Serial number DQ0VVVF59L
macOS Sequoia 15.7.2

Name	VingtMille
Chip	Apple M1 Max
Memory	64 GB
Serial number	DQ0VVVF59L
Coverage Expired	

macOS

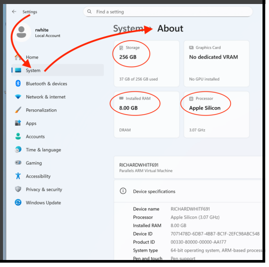
macOS Sequoia Version 15.7.2

Displays

Built-in Liquid Retina XDR Display 14-inch (3024 × 1964)

Storage

Macintosh HD 1.01 TB available of 2 TB



Storage 256 GB
37 GB of 256 GB used

Graphics Card No dedicated VRAM
No GPU installed

Installed RAM 8.00 GB
DRAM

Processor Apple Silicon
3.07 GHz

RICHARDWHITF691
Parallels ARM Virtual Machine

Device specifications

Device name	RICHARDWHITF691
Processor	Apple Silicon (3.07 GHz)
Installed RAM	8.00 GB
Device ID	7071478D-6DB7-4BB7-BC1F-2EFC98ABC548
Product ID	00330-80000-00000-AA177
System type	64-bit operating system, ARM-based process
Pen and touch	Pen support

10

Hardware [Activity]

Find these computers in the room:

- a desktop computer
- a “smart phone”
- a “smart watch”
- a server
- Raspberry Pi
- a computer running Windows
- a computer running macOS
- a computer running Linux

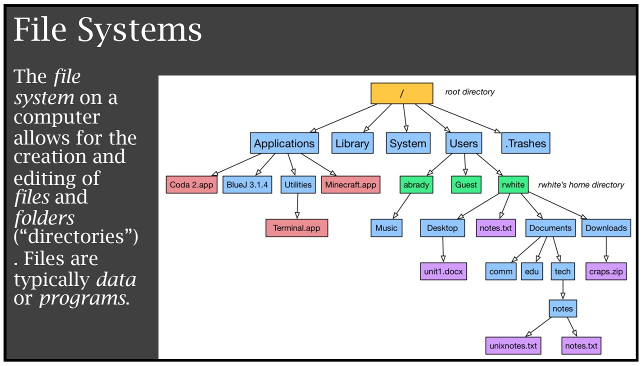
11

Operating Systems

The *operating system* is the lowest level software running on your computer. It allows other programs to run on the computer so that you can interact with it.

- Apple’s *macOS*
- Microsoft’s *Windows*
- Open-source *Linux*

12

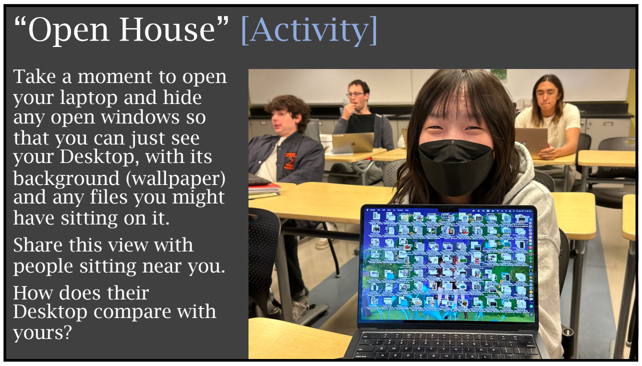


File Systems

The *file system* on a computer allows for the creation and editing of *files* and *folders* ("directories"). Files are typically *data* or *programs.*

/
root directory
Applications
Library
System
Users
.Trashes
Coda 2.app
BlueJ 3.1.4
Utilities
Minecraft.app
abrady
Guest
rwhite
rwhite's home directory
Terminal.app
Music
Desktop
notes.txt
Documents
Downloads
unit1.docx
comm
edu
tech
craps.zip
notes
unixnotes.txt
notes.txt

13



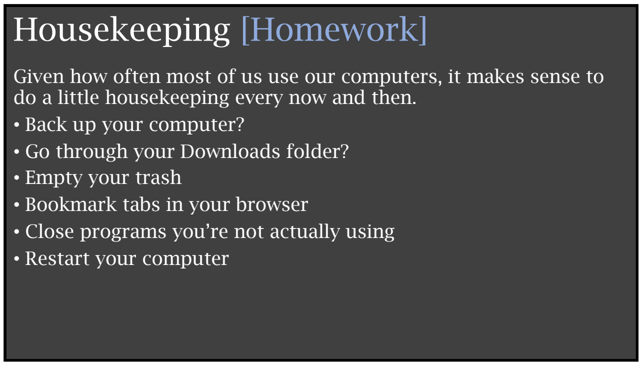
"Open House" [Activity]

Take a moment to open your laptop and hide any open windows so that you can just see your Desktop, with its background (wallpaper) and any files you might have sitting on it.

Share this view with people sitting near you.

How does their Desktop compare with yours?

14

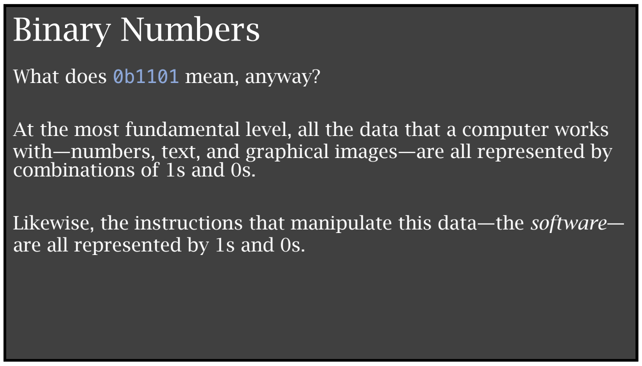


Housekeeping [Homework]

Given how often most of us use our computers, it makes sense to do a little housekeeping every now and then.

- Back up your computer?
- Go through your Downloads folder?
- Empty your trash
- Bookmark tabs in your browser
- Close programs you're not actually using
- Restart your computer

15



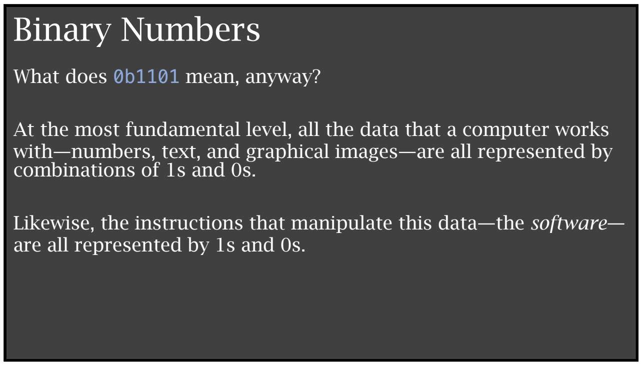
Binary Numbers

What does `0b1101` mean, anyway?

At the most fundamental level, all the data that a computer works with—numbers, text, and graphical images—are all represented by combinations of 1s and 0s.

Likewise, the instructions that manipulate this data—the *software*—are all represented by 1s and 0s.

16



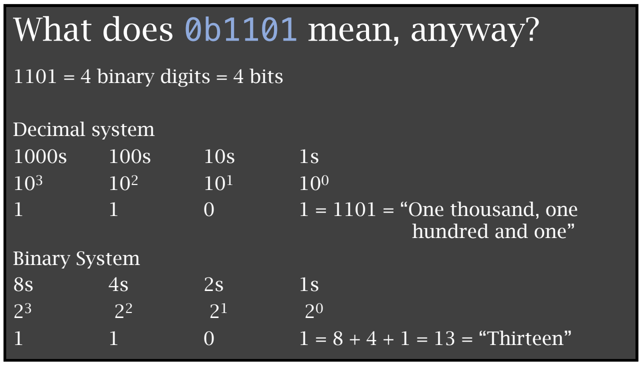
Binary Numbers

What does `0b1101` mean, anyway?

At the most fundamental level, all the data that a computer works with—numbers, text, and graphical images—are all represented by combinations of 1s and 0s.

Likewise, the instructions that manipulate this data—the *software*—are all represented by 1s and 0s.

17



What does `0b1101` mean, anyway?

1101 = 4 binary digits = 4 bits

Decimal system

1000s	100s	10s	1s
10^3	10^2	10^1	10^0
1	1	0	1 = 1101 = "One thousand, one hundred and one"

Binary System

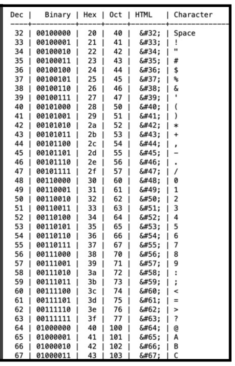
8s	4s	2s	1s
2^3	2^2	2^1	2^0
1	1	0	1 = 8 + 4 + 1 = 13 = "Thirteen"

18

How is text represented?

American Standard Code for Information Interchange (ASCII)

UTF-8 UTF-16 UTF-32



```
Dec |   Binary | Hex | Oct | HTML    | Character
----+----------+-----+-----+---------+----------
 32 | 00100000 |  20 |  40 |  &#32; | Space
 33 | 00100001 |  21 |  41 |  &#33; | !
 34 | 00100010 |  22 |  42 |  &#34; | "
 35 | 00100011 |  23 |  43 |  &#35; | #
 36 | 00100100 |  24 |  44 |  &#36; | $
 37 | 00100101 |  25 |  45 |  &#37; | %
 38 | 00100110 |  26 |  46 |  &#38; | &
 39 | 00100111 |  27 |  47 |  &#39; | '
 40 | 00101000 |  28 |  50 |  &#40; | (
 41 | 00101001 |  29 |  51 |  &#41; | )
 42 | 00101010 |  2a |  52 |  &#42; | *
 43 | 00101011 |  2b |  53 |  &#43; | +
 44 | 00101100 |  2c |  54 |  &#44; | ,
 45 | 00101101 |  2d |  55 |  &#45; | -
 46 | 00101110 |  2e |  56 |  &#46; | .
 47 | 00101111 |  2f |  57 |  &#47; | /
 48 | 00110000 |  30 |  60 |  &#48; | 0
 49 | 00110001 |  31 |  61 |  &#49; | 1
 50 | 00110010 |  32 |  62 |  &#50; | 2
 51 | 00110011 |  33 |  63 |  &#51; | 3
 52 | 00110100 |  34 |  64 |  &#52; | 4
 53 | 00110101 |  35 |  65 |  &#53; | 5
 54 | 00110110 |  36 |  66 |  &#54; | 6
 55 | 00110111 |  37 |  67 |  &#55; | 7
 56 | 00111000 |  38 |  70 |  &#56; | 8
 57 | 00111001 |  39 |  71 |  &#57; | 9
 58 | 00111010 |  3a |  72 |  &#58; | :
 59 | 00111011 |  3b |  73 |  &#59; | ;
 60 | 00111100 |  3c |  74 |  &#60; | <
 61 | 00111101 |  3d |  75 |  &#61; | =
 62 | 00111110 |  3e |  76 |  &#62; | >
 63 | 00111111 |  3f |  77 |  &#63; | ?
 64 | 01000000 |  40 | 100 |  &#64; | @
 65 | 01000001 |  41 | 101 |  &#65; | A
 66 | 01000010 |  42 | 102 |  &#66; | B
 67 | 01000011 |  43 | 103 |  &#67; | C
```

19

Binary Programming

https://youtu.be/wdP8WB8Dwbg?si=UnepyAdRDp73EgJw

20

Encapsulation [Demo]

Nobody wants to talk to a computer by flipping switches. *Encapsulation* is the process of hiding hard-to-do things, like programming in binary.

1. Machine language (0s and 1s)
2. Assembly (Human-readable machine language)
3. C (high-level language that is compiled into machine language)
4. Python (higher-level language, written in C)

21

Break

Restrooms down the hall.
Refreshments in here.
Back in 10 minutes?

22

In Introduction to writing code

What does coding look like?

23

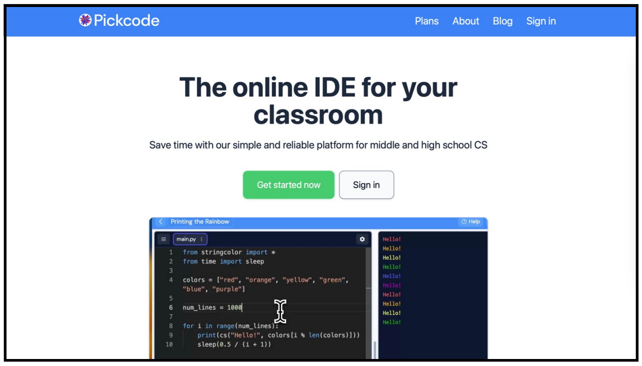
Writing, running, debugging

The process of *developing* typically involves installing a bunch of software tools onto your computer, and working in a *Terminal* or an *Integrated Development Environment.*

As an easy, simpler alternative, we'll be using a browser-based tool available at https://pickcode.io

Open up a browser (Firefox, Safari, Chrome), go to that address, and create a free account for yourself.

24



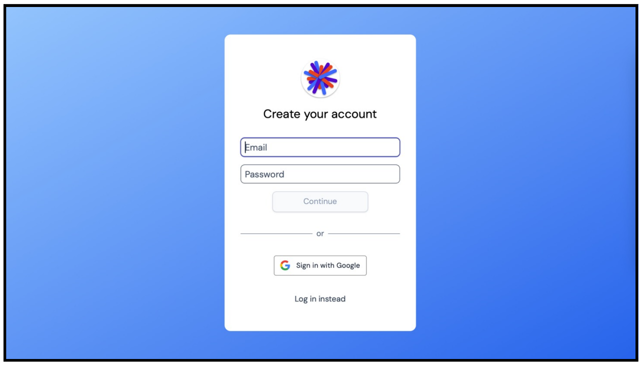
The online IDE for your classroom

Save time with our simple and reliable platform for middle and high school CS

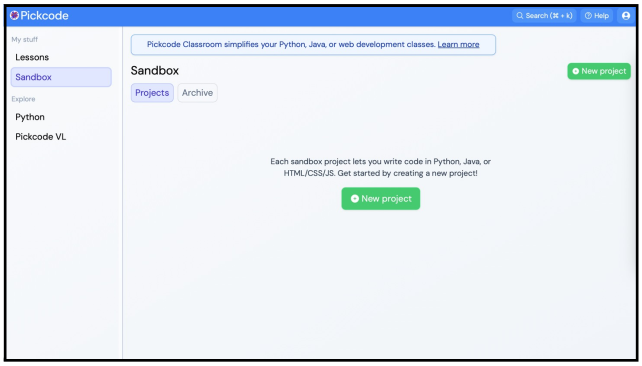
```python
from stringcolor import *
from time import sleep

colors = ["red", "orange", "yellow", "green",
"blue", "purple"]

num_lines = 1000

for i in range(num_lines):
    print(cs("Hello!", colors[i % len(colors)]))
    sleep(0.5 / (i + 1))
```

```
Hello!
Hello!
Hello!
Hello!
Hello!
Hello!
Hello!
Hello!
Hello!
Hello!
```

25



26

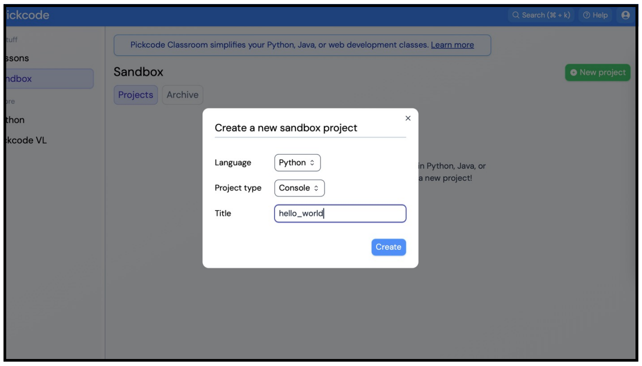


Pickcode Classroom simplifies your Python, Java, or web development classes. Learn more

Sandbox

Each sandbox project lets you write code in Python, Java, or HTML/CSS/JS. Get started by creating a new project!

27



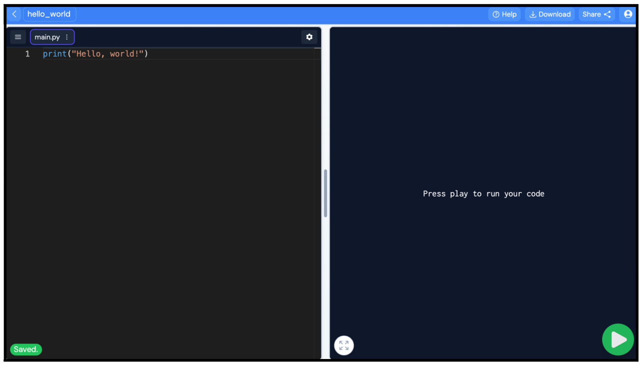
Create a new sandbox project

Language: Python

Project type: Console

Title: hello_world

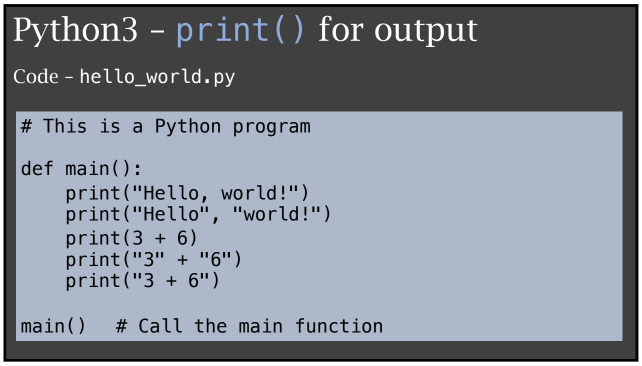
28



```python
print("Hello, world!")
```

```
Press play to run your code
```

29

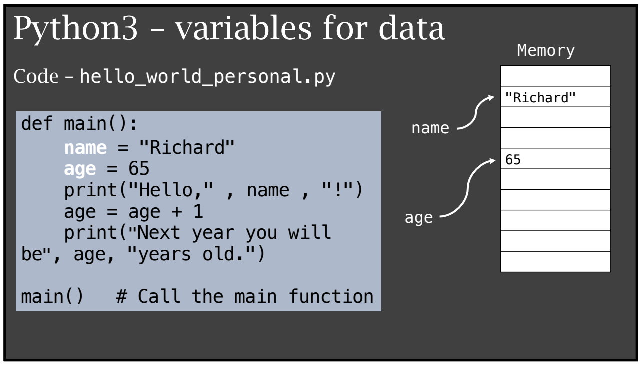


Python3 – print() for output

Code – hello_world.py

```python
# This is a Python program

def main():
    print("Hello, world!")
    print("Hello", "world!")
    print(3 + 6)
    print("3" + "6")
    print("3 + 6")

main()   # Call the main function
```

30



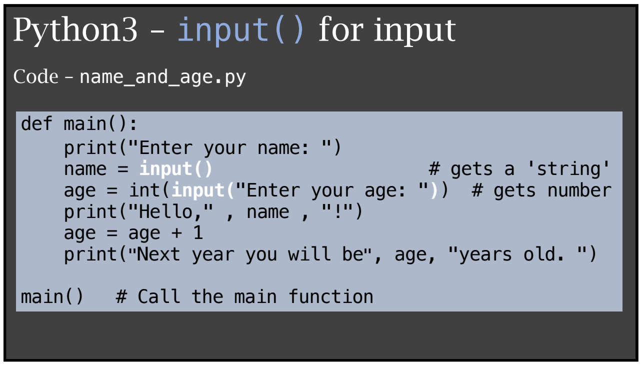
Python3 – variables for data

Code – hello_world_personal.py

```
def main():
    name = "Richard"
    age = 65
    print("Hello," , name , "!")
    age = age + 1
    print("Next year you will
be", age, "years old.")

main()   # Call the main function
```

Memory
"Richard"
65
name
age

31

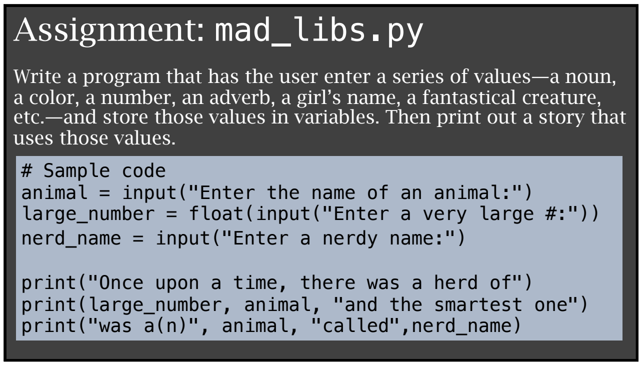


Python3 – input() for input

Code – name_and_age.py

```
def main():
    print("Enter your name: ")
    name = input()                         # gets a 'string'
    age = int(input("Enter your age: "))  # gets number
    print("Hello," , name , "!")
    age = age + 1
    print("Next year you will be", age, "years old. ")

main()   # Call the main function
```

32

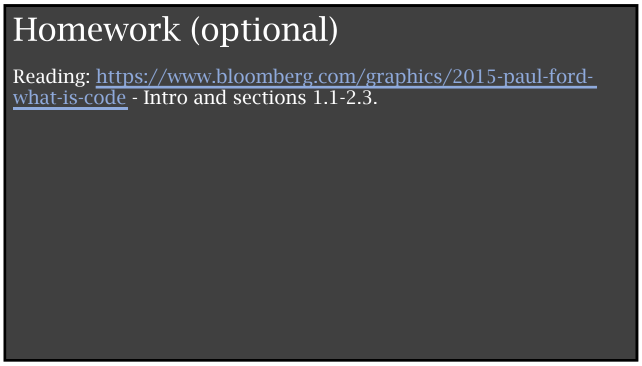


Assignment: mad_libs.py

Write a program that has the user enter a series of values—a noun, a color, a number, an adverb, a girl's name, a fantastical creature, etc.—and store those values in variables. Then print out a story that uses those values.

```
# Sample code
animal = input("Enter the name of an animal:")
large_number = float(input("Enter a very large #:"))
nerd_name = input("Enter a nerdy name:")

print("Once upon a time, there was a herd of")
print(large_number, animal, "and the smartest one")
print("was a(n)", animal, "called",nerd_name)
```

33



Homework (optional)

Reading: https://www.bloomberg.com/graphics/2015-paul-ford-what-is-code - Intro and sections 1.1-2.3.

34