MAKE WITH PYTHON

TTU Library Makerspace Workshop
how to use Python for your magical code-making

2025

Who am I?

Sean W Scully

Lead Administrator – TTU Libraries - Emerging Technology

Rhino3D, TinkerCAD, AutoCAD Inventor, Fusion360, Blender, Solidworks
MakeCode, Python, Scratch, C++, C, MATLAB, Arduino, p5.js, JavaScript, Assembly, Verilog, Xilinx

GSB, General Studies – Math/Engineering/Renewable Energy, TTU

MFA, Studio Art - Metals/Jewelry/Enameling, Kent State Univ

BFA, Studio Art - Metalsmithing/Jewelry Design, TTU

AA, Fine Arts, South Plains College

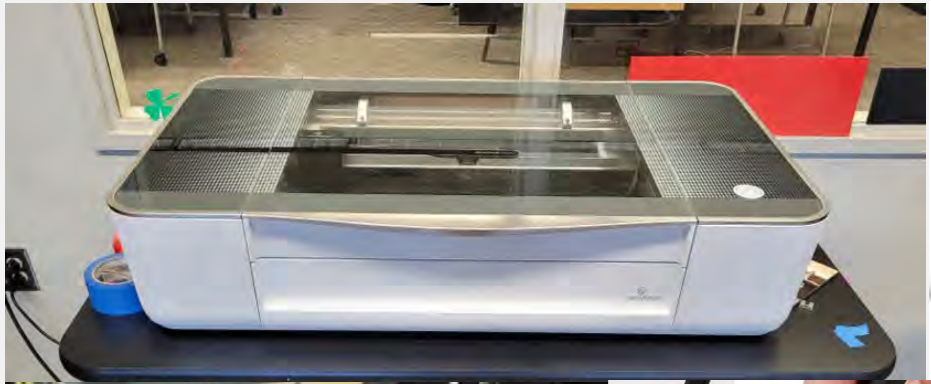
linkedin.com/in/seanwscully
Find me on the First Friday Art Trail, ffat.org, CASP Work Studio H

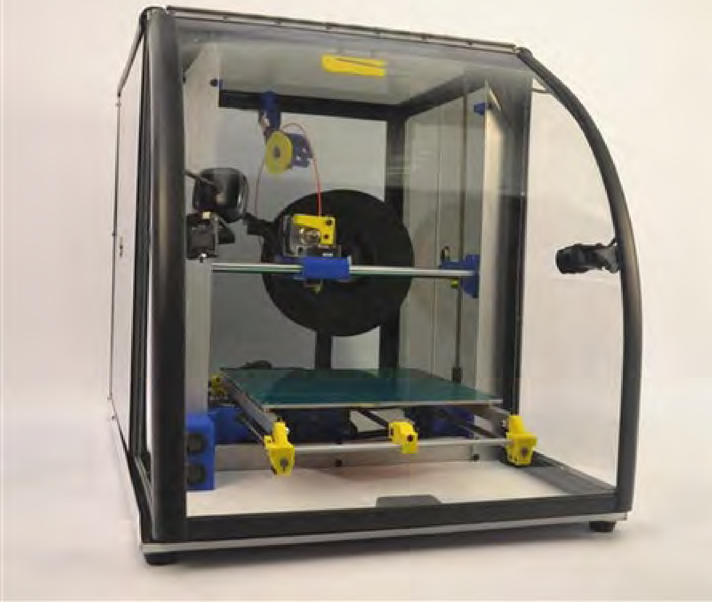
Emerging Technology Department

- Makerspace: Main Library, 2nd floor, room 210
- VR Lab: Main Library, 2nd floor, room 201A

make@ttu.edu

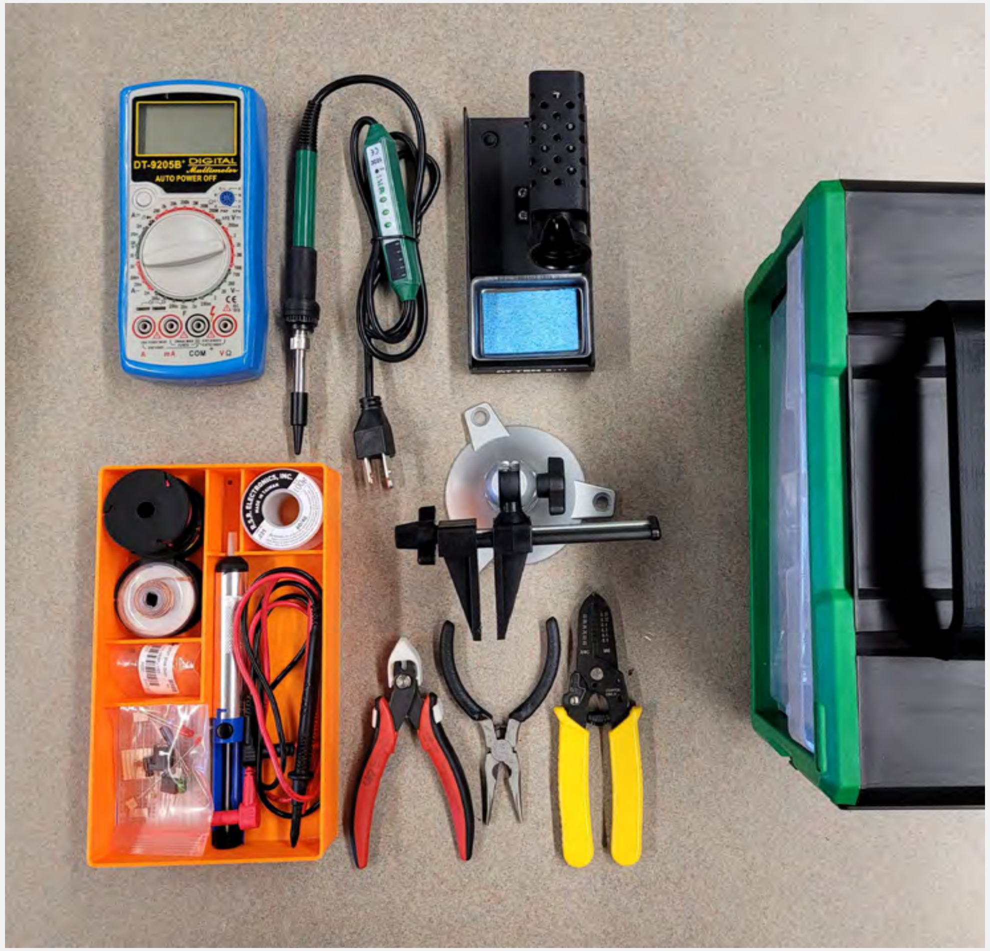
https://www.depts.ttu.edu/library/make/

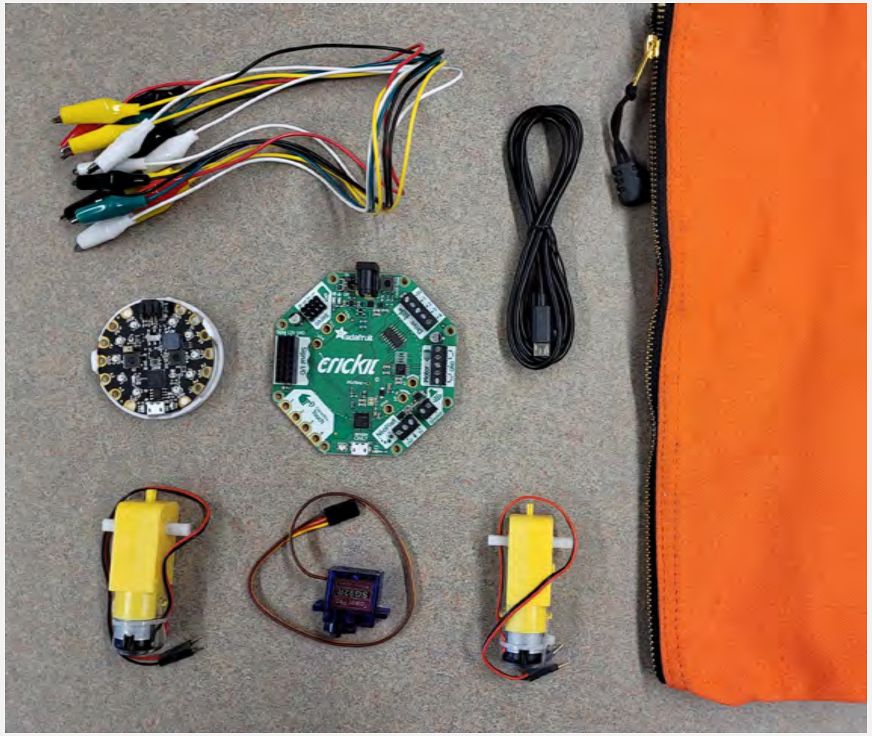




- Cricut Maker/Venture
- Glowforge Lasercutter
- Makyu Formbox
- Matter&Form 3D Scanner
- PolyPrinters 229 & 508
- Ultimaker 3, S7
- Sewing machines

TTU Library's Makerspace *Electro* and *Robo* Kits



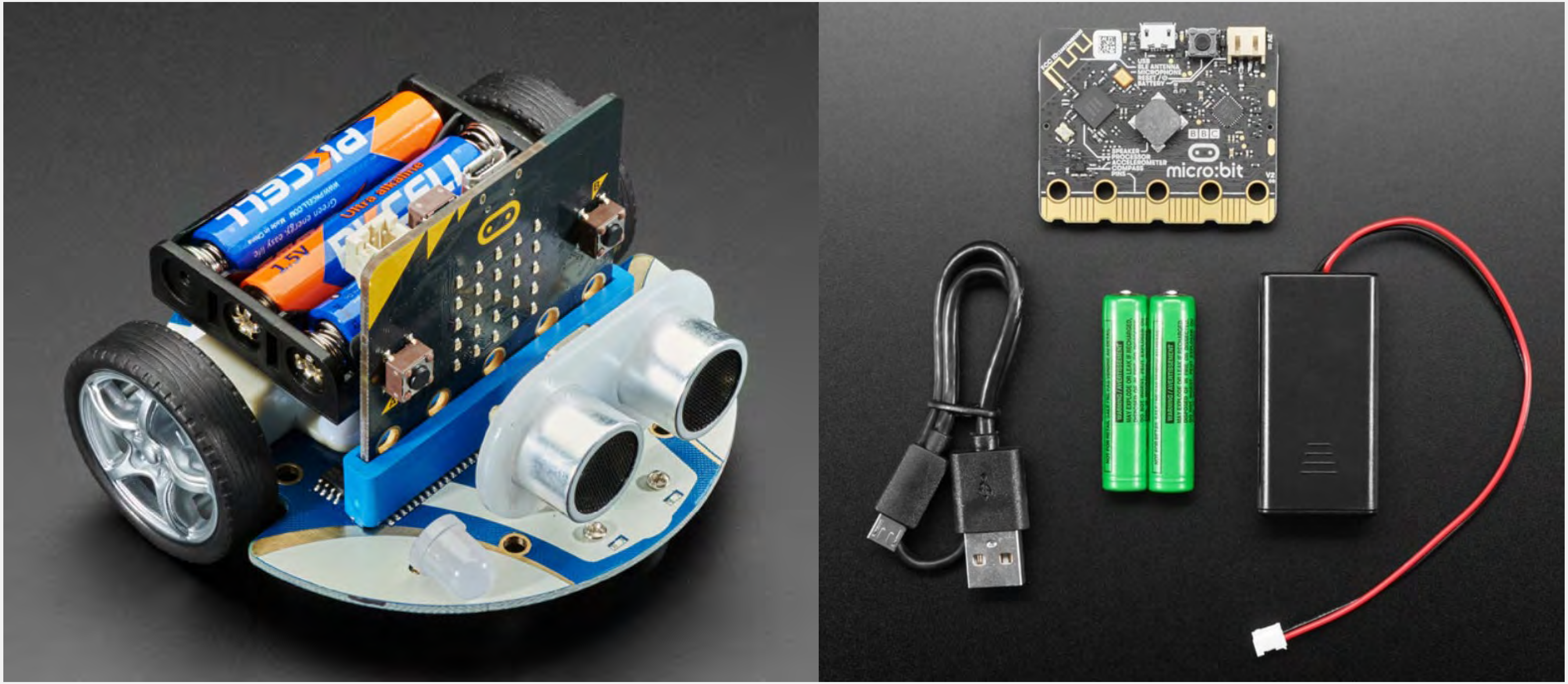


Robo Kits (15)

- Adafruit's Circuit Playground Express (CPX)
- Adafruit's Crickit robotic control board
- geared motors (2), micro servo

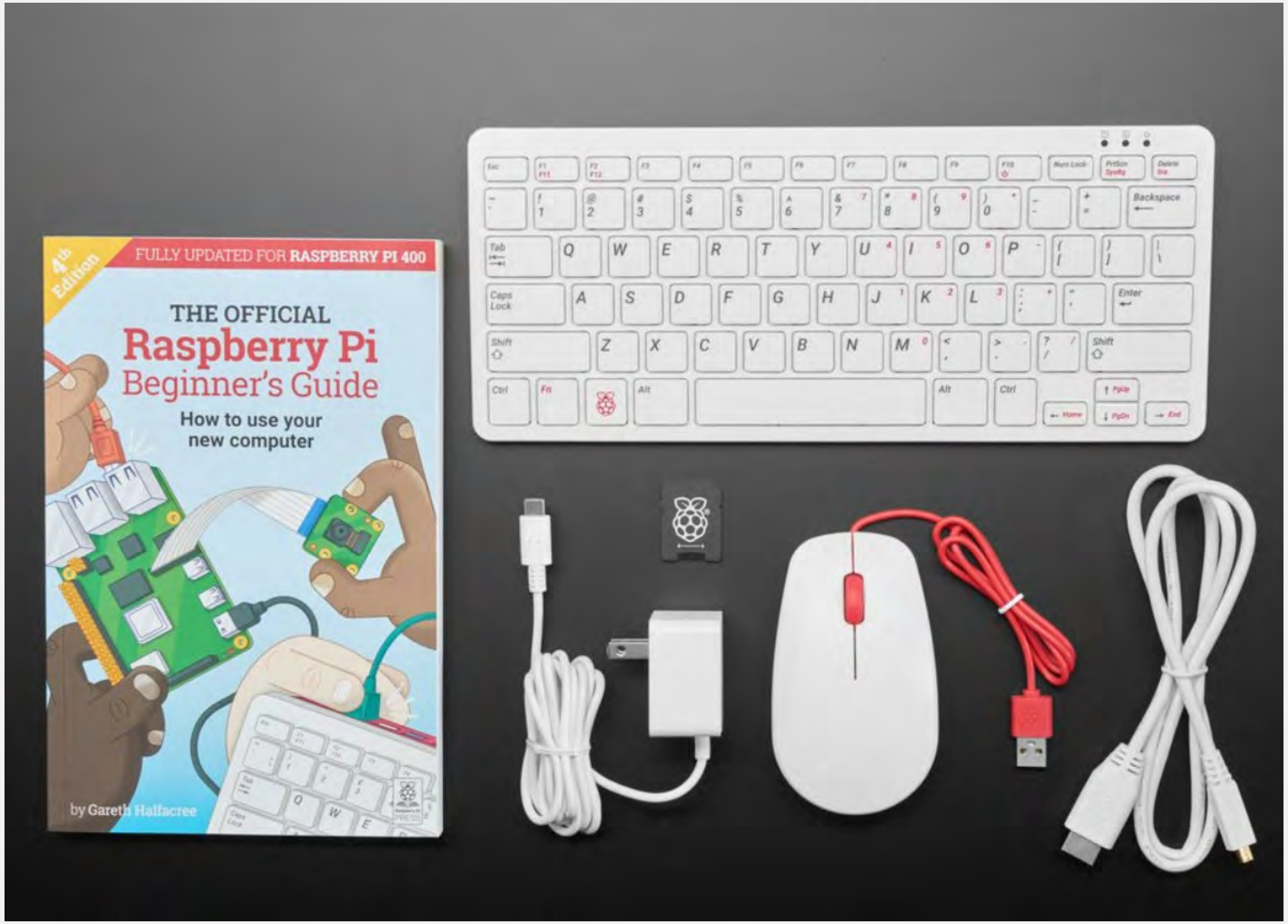
ElectroSoldering Kits (5)

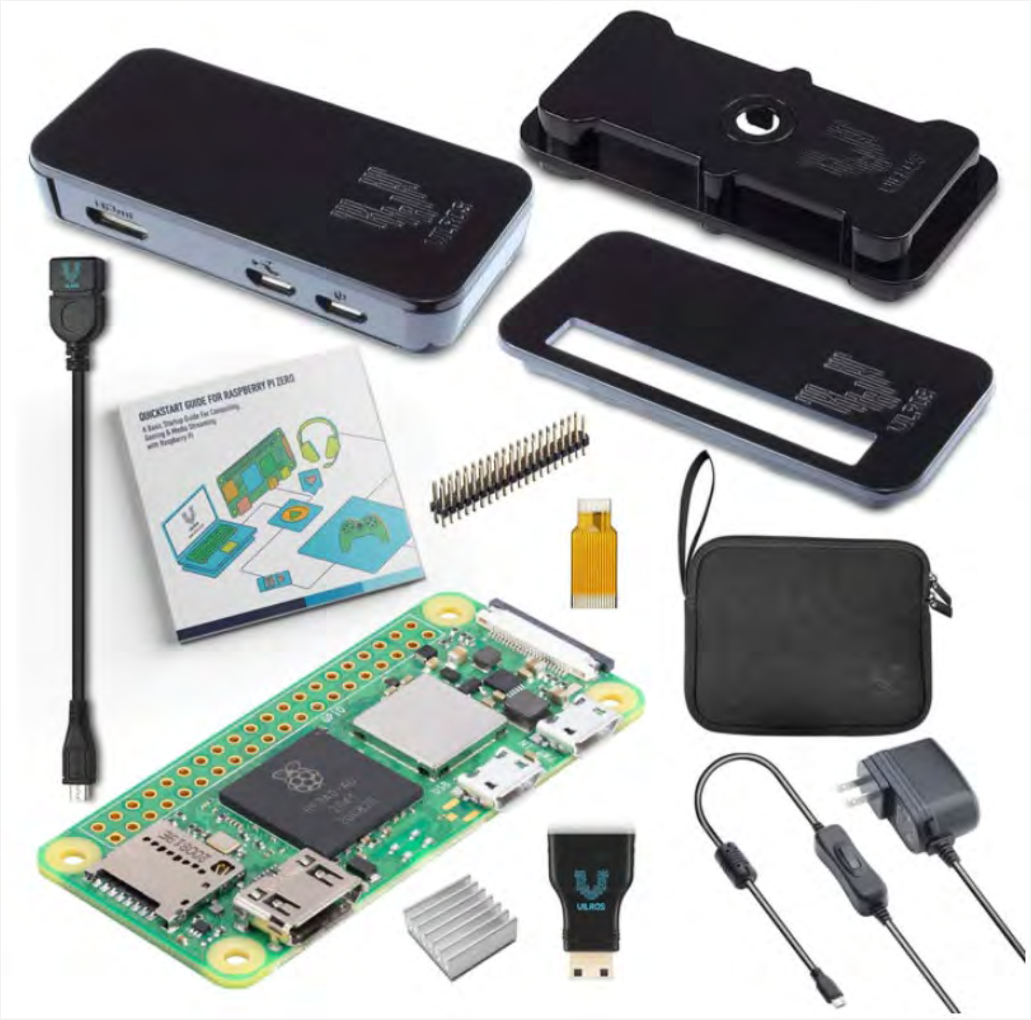
TTU Library's Makerspace *CuteBot* and *Micro:Bit v2* Kits



Images of our CuteBot kit and Micro:Bit v2 kit

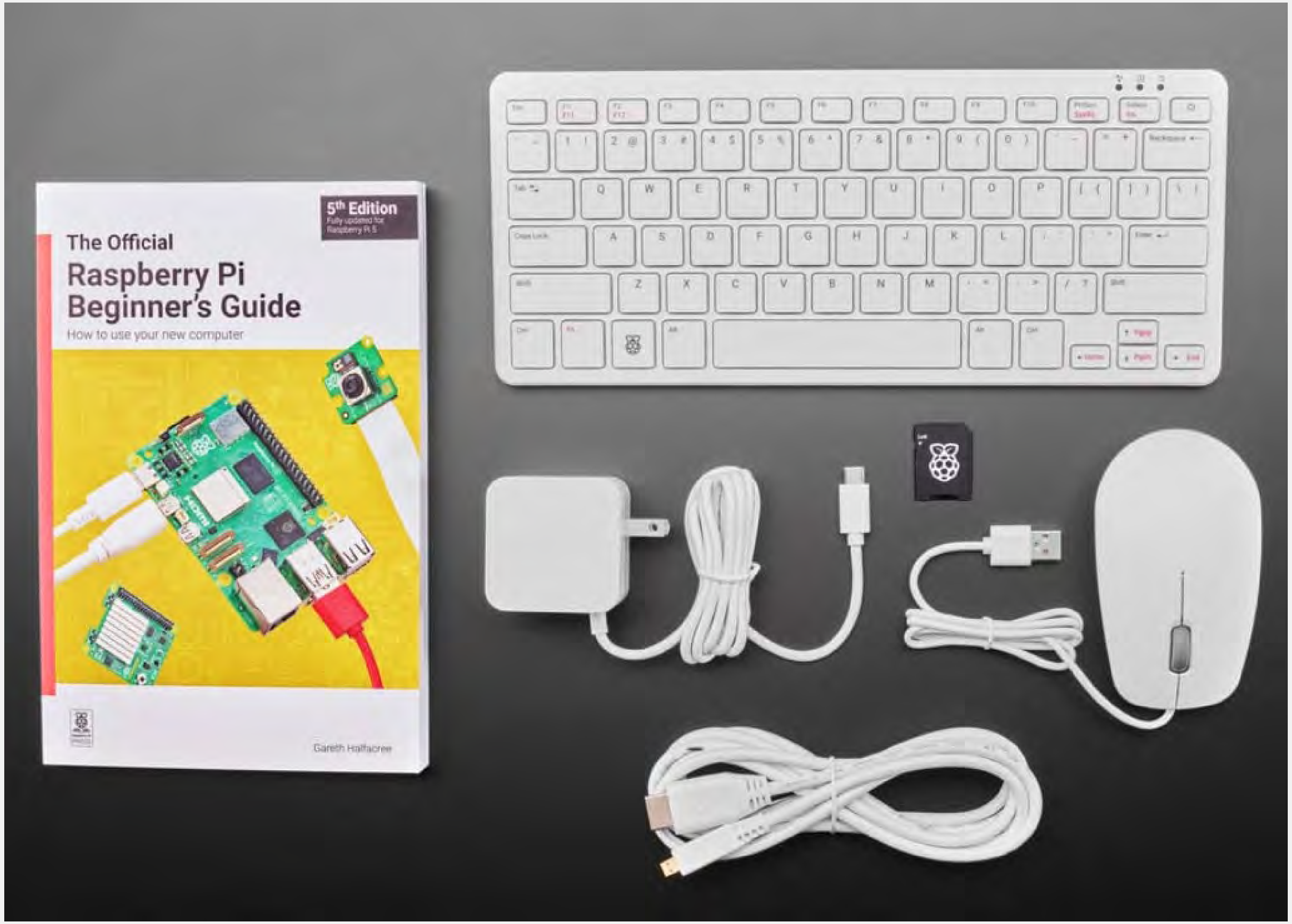
TTU Library's Makerspace *Pi-400* and *Pi-Zero W* Kits

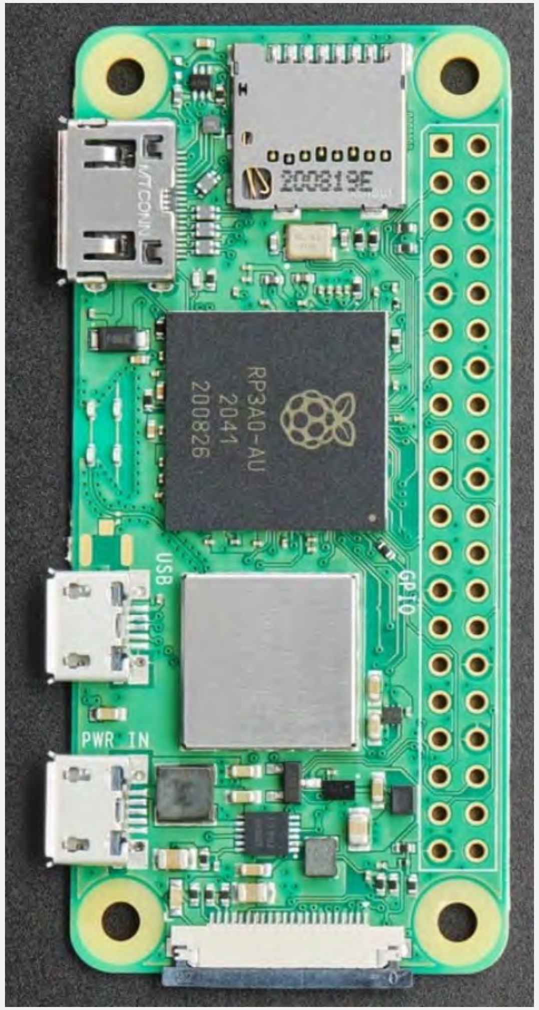




6 kits of each is available: recommend supplying your own microSD card (minimum size: 8Gb)

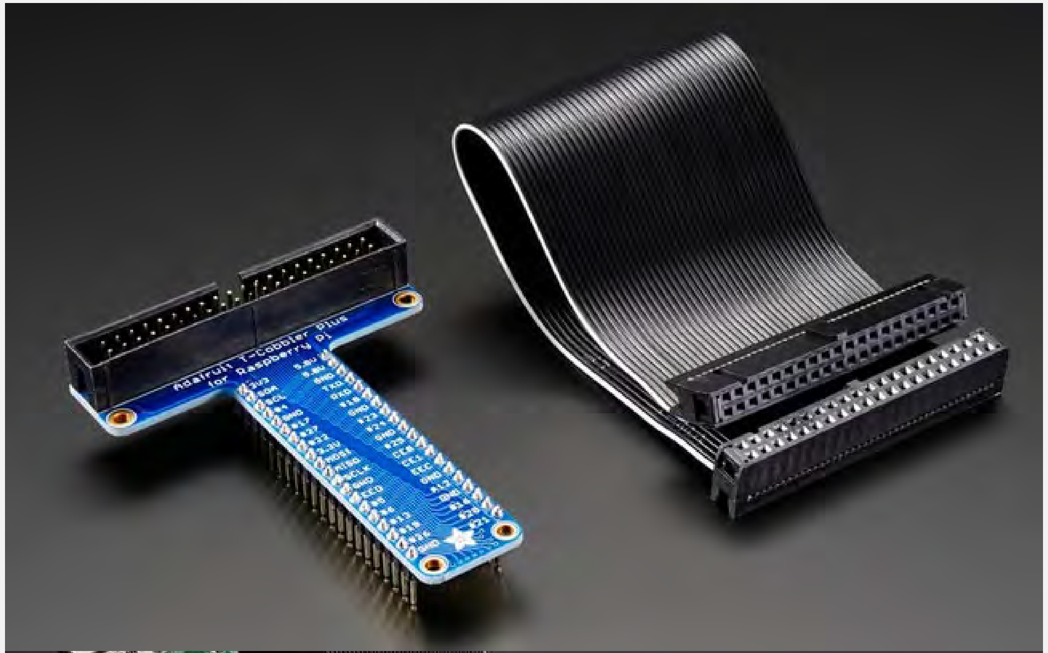
TTU Library's Makerspace *Pi-500* and *Pi-Zero2W* Kits

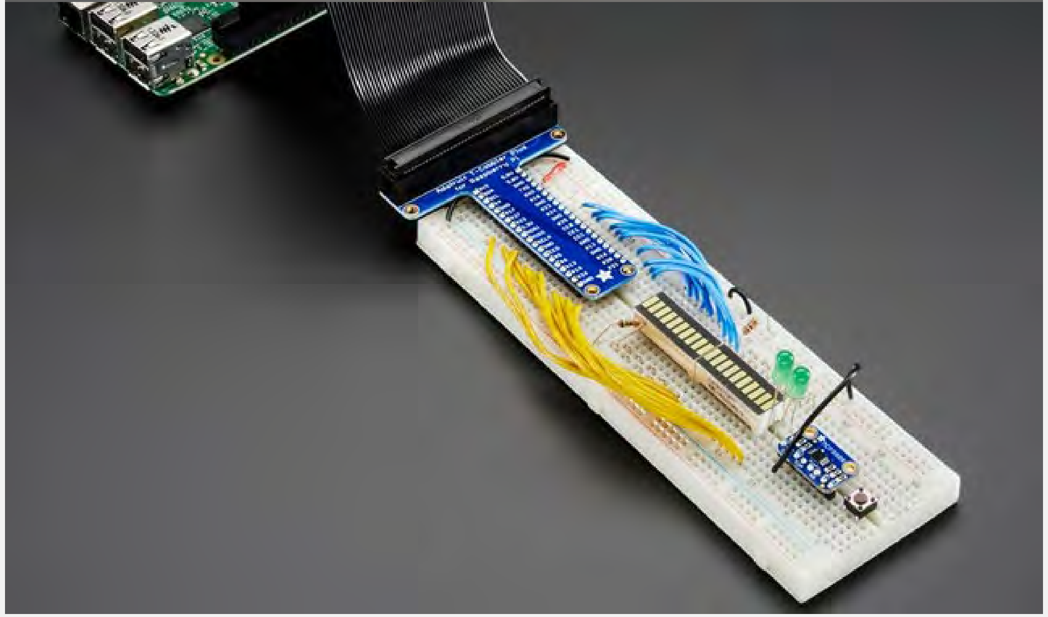


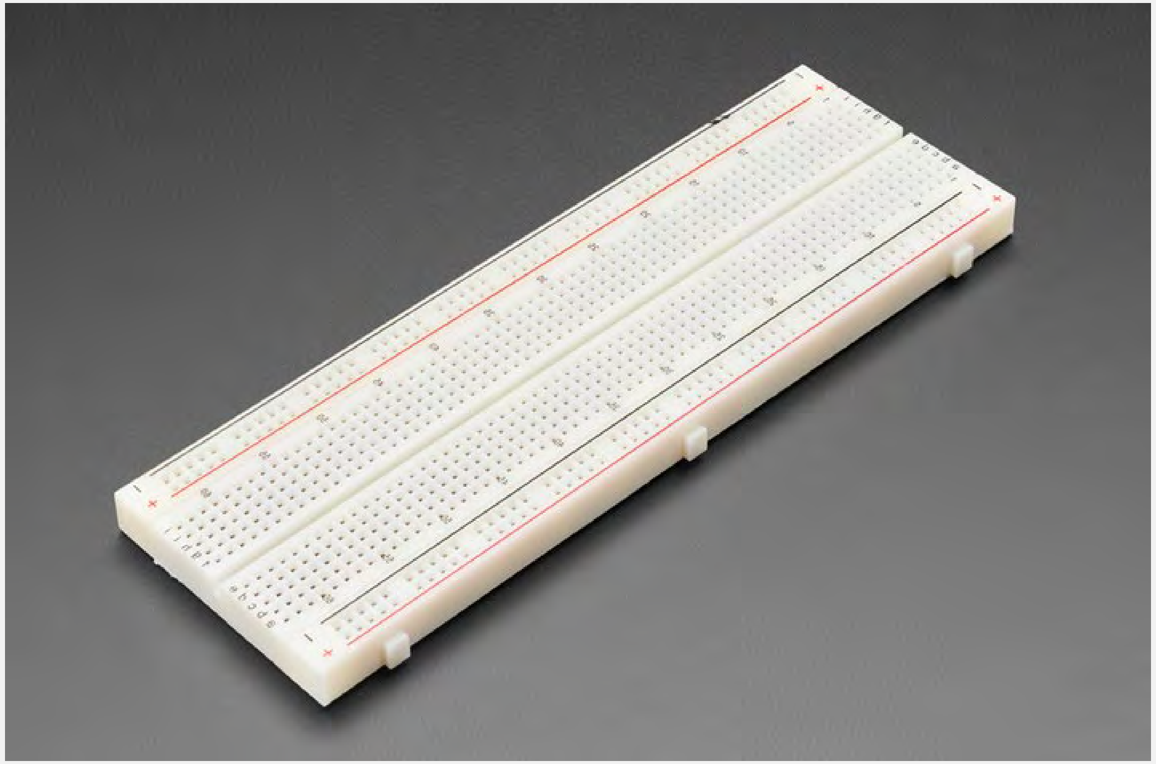


2 kits of each is available: recommend supplying your own microSD card (minimum size: 32Gb)

TTU Library's Makerspace *Pi – Interface* Kits







6 kits of each is available, one for every Pi-400

- Pi T-Cobbler Plus
    - GPIO Breakout – Pi A+, B+, Pi 2/3/4, Zero
- Full Sized Premium Breadboard
    - 830 Tie Points

TTU Library's Makerspace *iFixit* Kits

6 kits of each is available, one for every Pi-400

- all kinds of screwdriver tips, 64 total
- larger kit has spudgers and plastic pryers

intro to code

- what is coding?

- why are there so many computer languages?

- what is Python?

what is coding?

computer programming

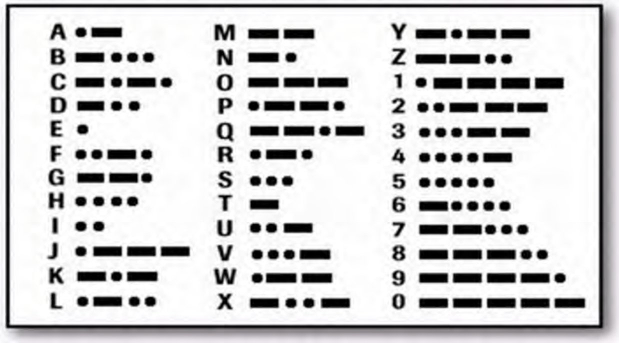
directly asking the computer to do something in a language it understands

language => code

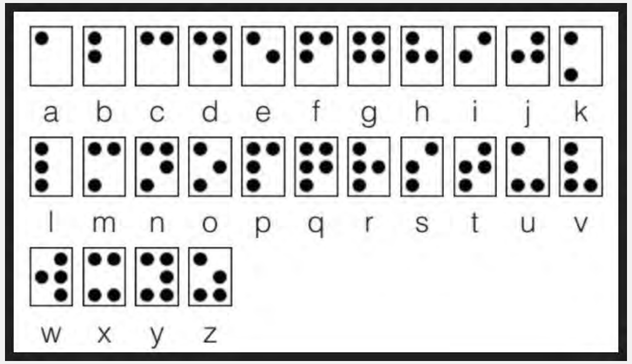
Speech => Text – text commands specific to computer's language

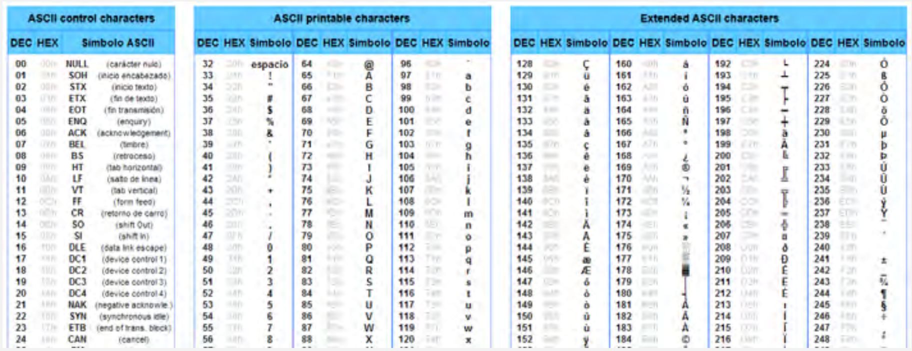
Text => Numbers – hexadecimal or binary numbers, e.g. ASCII

Numbers => Machine Codes – "microcode" encoded on the silicon chips



A	•—	M	——	Y	—•——
B	—•••	N	—•	Z	——••
C	—•—•	O	———	1	•————
D	—••	P	•——•	2	••———
E	•	Q	——•—	3	•••——
F	••—•	R	•—•	4	••••—
G	——•	S	•••	5	•••••
H	••••	T	—	6	—••••
I	••	U	••—	7	——•••
J	•———	V	•••—	8	———••
K	—•—	W	•——	9	————•
L	•—••	X	—••—	0	—————





ASCII control characters

DEC	HEX	Simbolo ASCII	
00	00h	NULL	(carácter nulo)
01	01h	SOH	(inicio encabezado)
02	02h	STX	(inicio texto)
03	03h	ETX	(fin de texto)
04	04h	EOT	(fin transmisión)
05	05h	ENQ	(enquiry)
06	06h	ACK	(acknowledgement)
07	07h	BEL	(timbre)
08	08h	BS	(retroceso)
09	09h	HT	(tab horizontal)
10	0Ah	LF	(salto de linea)
11	0Bh	VT	(tab vertical)
12	0Ch	FF	(form feed)
13	0Dh	CR	(retorno de carro)
14	0Eh	SO	(shift Out)
15	0Fh	SI	(shift In)
16	10h	DLE	(data link escape)
17	11h	DC1	(device control 1)
18	12h	DC2	(device control 2)
19	13h	DC3	(device control 3)
20	14h	DC4	(device control 4)
21	15h	NAK	(negative acknowle.)
22	16h	SYN	(synchronous idle)
23	17h	ETB	(end of trans. block)
24	18h	CAN	(cancel)

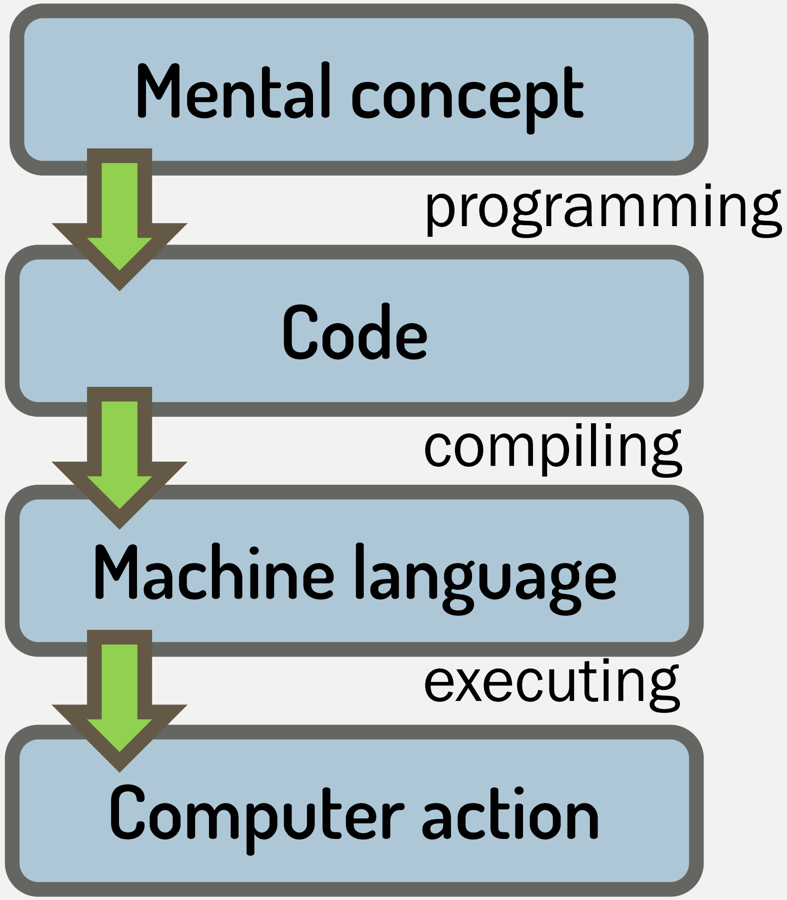
ASCII printable characters

DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo
32	20h	espacio	64	40h	@	96	60h	`
33	21h	!	65	41h	A	97	61h	a
34	22h	"	66	42h	B	98	62h	b
35	23h	#	67	43h	C	99	63h	c
36	24h	$	68	44h	D	100	64h	d
37	25h	%	69	45h	E	101	65h	e
38	26h	&	70	46h	F	102	66h	f
39	27h	'	71	47h	G	103	67h	g
40	28h	(	72	48h	H	104	68h	h
41	29h	)	73	49h	I	105	69h	i
42	2Ah	*	74	4Ah	J	106	6Ah	j
43	2Bh	+	75	4Bh	K	107	6Bh	k
44	2Ch	,	76	4Ch	L	108	6Ch	l
45	2Dh	-	77	4Dh	M	109	6Dh	m
46	2Eh	.	78	4Eh	N	110	6Eh	n
47	2Fh	/	79	4Fh	O	111	6Fh	o
48	30h	0	80	50h	P	112	70h	p
49	31h	1	81	51h	Q	113	71h	q
50	32h	2	82	52h	R	114	72h	r
51	33h	3	83	53h	S	115	73h	s
52	34h	4	84	54h	T	116	74h	t
53	35h	5	85	55h	U	117	75h	u
54	36h	6	86	56h	V	118	76h	v
55	37h	7	87	57h	W	119	77h	w
56	38h	8	88	58h	X	120	78h	x

Extended ASCII characters

DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo
128	80h	Ç	160	A0h	á	192	C0h	└	224	E0h	Ó
129	81h	ü	161	A1h	í	193	C1h	┴	225	E1h	ß
130	82h	é	162	A2h	ó	194	C2h	┬	226	E2h	Ô
131	83h	â	163	A3h	ú	195	C3h	├	227	E3h	Ò
132	84h	ä	164	A4h	ñ	196	C4h	─	228	E4h	õ
133	85h	à	165	A5h	Ñ	197	C5h	┼	229	E5h	Õ
134	86h	å	166	A6h	ª	198	C6h	ã	230	E6h	µ
135	87h	ç	167	A7h	º	199	C7h	Ã	231	E7h	þ
136	88h	ê	168	A8h	¿	200	C8h	╚	232	E8h	Þ
137	89h	ë	169	A9h	®	201	C9h	╔	233	E9h	Ú
138	8Ah	è	170	AAh	¬	202	CAh	╩	234	EAh	Û
139	8Bh	ï	171	ABh	½	203	CBh	╦	235	EBh	Ù
140	8Ch	î	172	ACh	¼	204	CCh	╠	236	ECh	ý
141	8Dh	ì	173	ADh	¡	205	CDh	═	237	EDh	Ý
142	8Eh	Ä	174	AEh	«	206	CEh	╬	238	EEh	¯
143	8Fh	Å	175	AFh	»	207	CFh	¤	239	EFh	´
144	90h	É	176	B0h	░	208	D0h	ð	240	F0h	
145	91h	æ	177	B1h	▒	209	D1h	Ð	241	F1h	±
146	92h	Æ	178	B2h	▓	210	D2h	Ê	242	F2h	‗
147	93h	ô	179	B3h	│	211	D3h	Ë	243	F3h	¾
148	94h	ö	180	B4h	┤	212	D4h	È	244	F4h	¶
149	95h	ò	181	B5h	Á	213	D5h	ı	245	F5h	§
150	96h	û	182	B6h	Â	214	D6h	Í	246	F6h	÷
151	97h	ù	183	B7h	À	215	D7h	Î	247	F7h	¸
152	98h	ÿ	184	B8h	©	216	D8h	Ï	248	F8h	°

coding overview


Mental concept

programming

Code

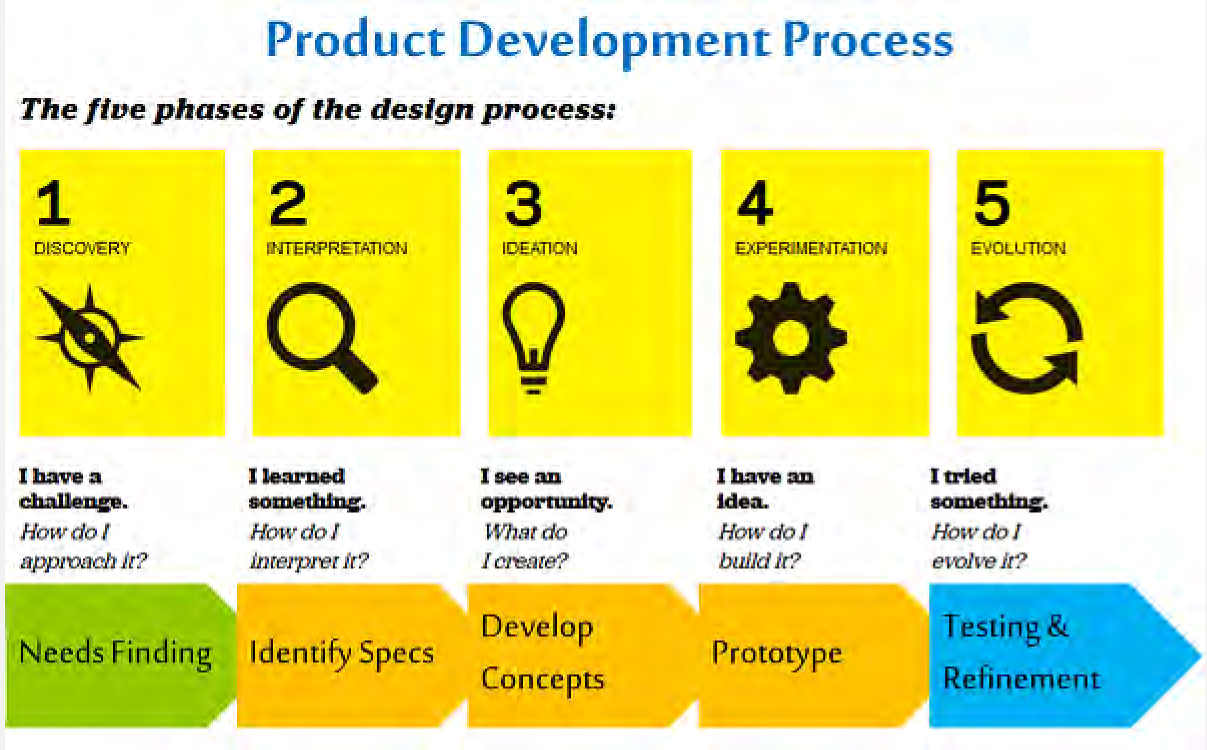
compiling

Machine language

executing

Computer action

http://www.toves.org/books/java/ch01-overview/process.png


Product Development Process

The five phases of the design process:

I have a challenge. *How do I approach it?*

I learned something. *How do I interpret it?*

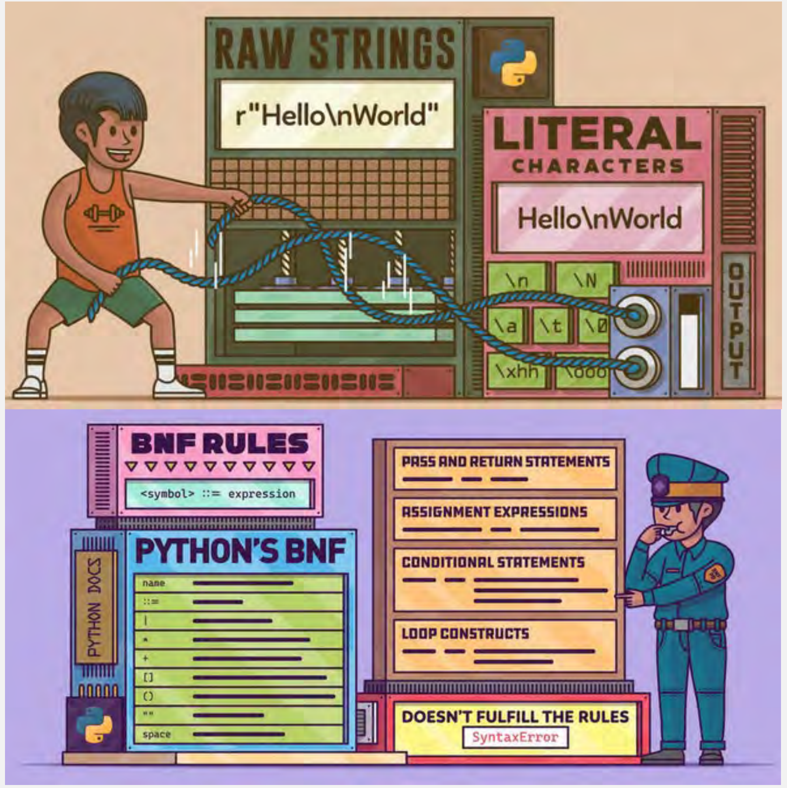
I see an opportunity. *What do I create?*

I have an idea. *How do I build it?*

I tried something. *How do I evolve it?*

https://bioebootcamp.sites.stanford.edu/system/files/product developementprocess_1.png

essence of code



- any language understood by the computer
  - *Commands – vocabulary/keywords that initiate an action*
    - run, print, for, while
  - *Syntax – order of words and uses of punctuation*
    - print (“hello world”)
  - *Uses data as an “input” to produce an “output”*

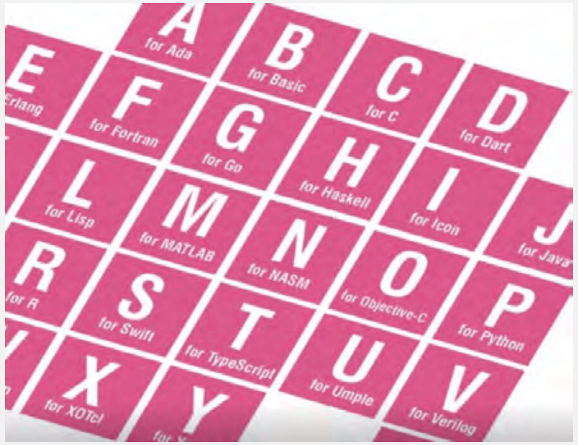
- most languages have similar keywords and syntax

fprintf (“A\n")	MATLAB	Output: A
print (“A”)	Python	Output: A
cout <<“A”;	C++	Output: A
print “(a10)”, str	Fortran	Output: A

why so many tongues?

The History of Computer Tongues

- *https://www.thesoftwareguild.com/blog/history-of-programming-languages/*



what is coding?

Language

Machine > Assembly > Procedural > Object-Oriented > Scripting

Code

Binary > Words/Symbols > Steps/Commands > Data-focused > Plain language

Fun “Brief History of Computers”

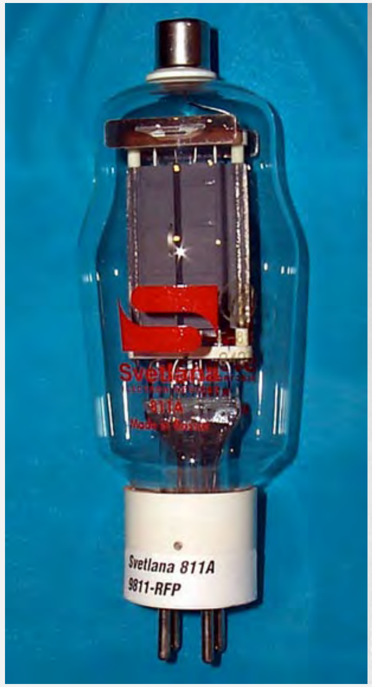
- *https://www.lesswrong.com/posts/vfRpzyGsikujm9ujj/a-brief-history-of-computers*

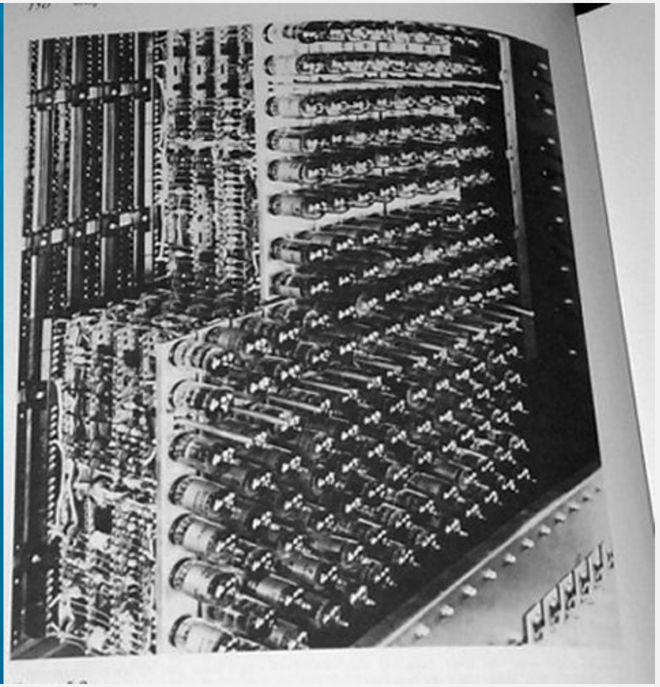
Secret Life of Machines Series

- *Word Processors – https://tinyurl.com/e8av8scx*

Hardware differences – Structure, Purpose

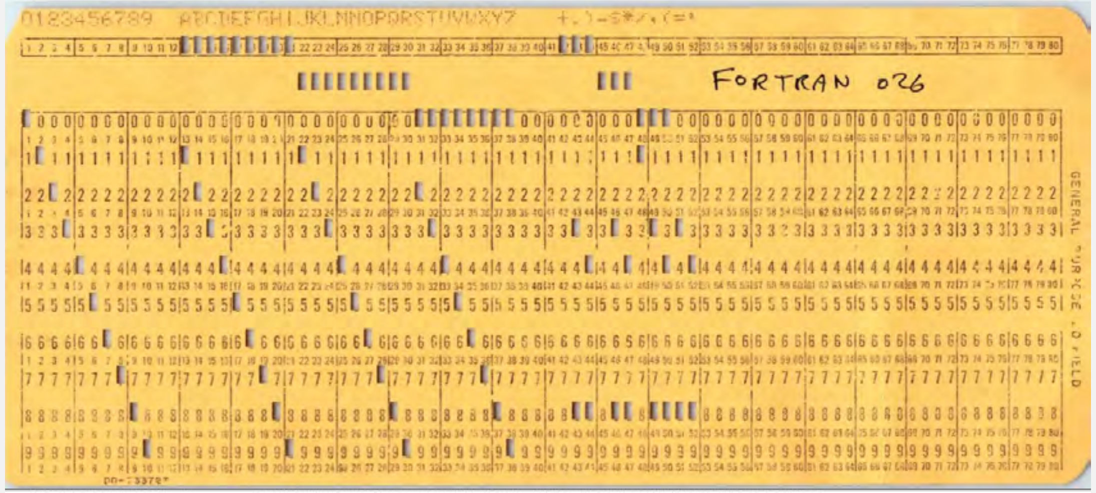
FORTRAN - IBM 700 Series (1950-60s) – room-size, vacuum tubes

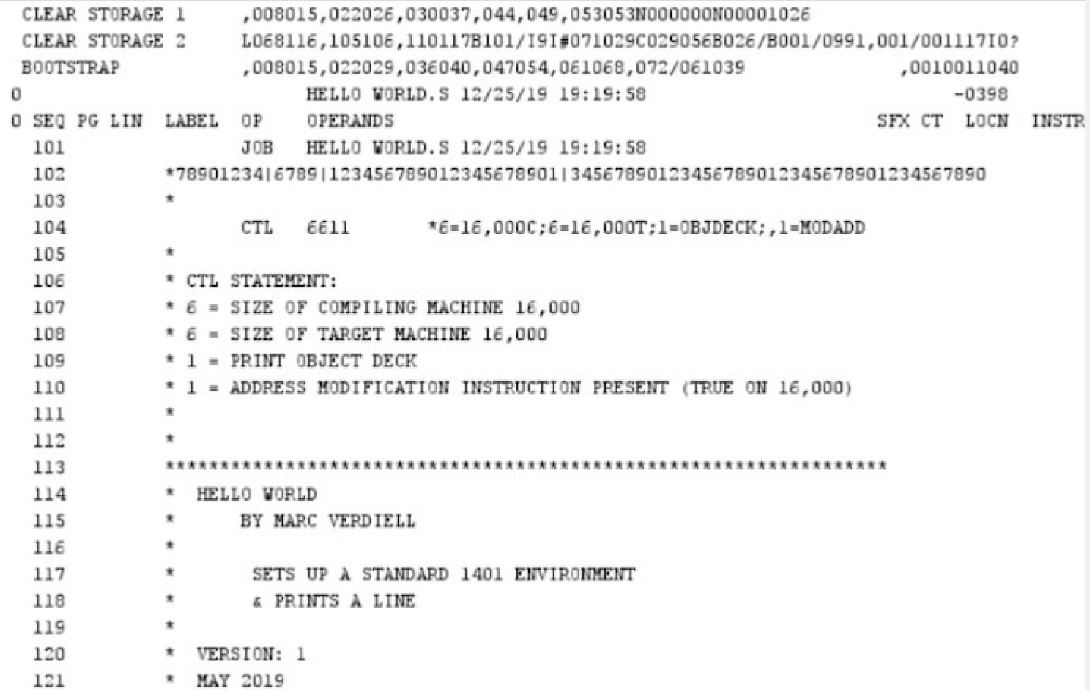




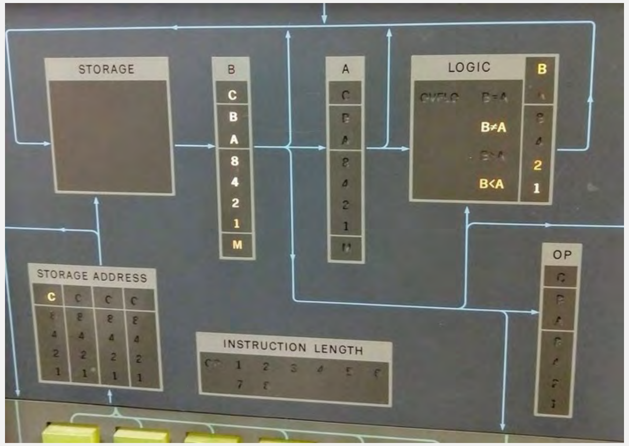
Hardware differences – Structure, Purpose

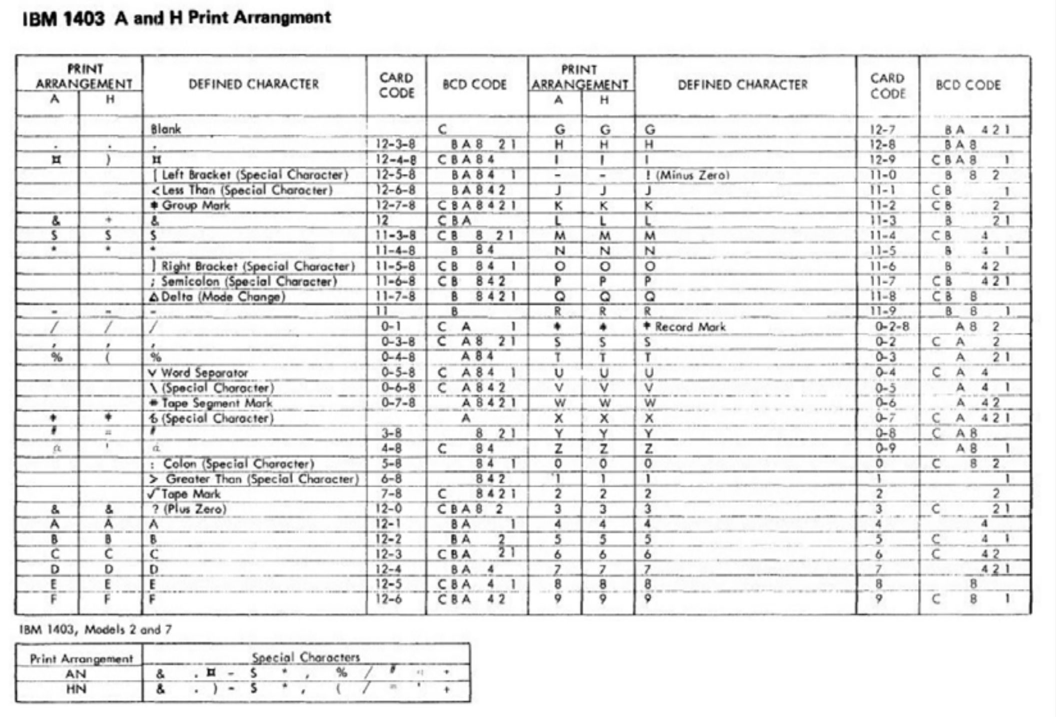
FORTRAN - IBM 1401 Series (1960s) – room-size, used vacuum tubes, punch cards, magnetic tape reels


0123456789 ABCDEFGHIJKLMNOPQRSTUVWXYZ +.)-$*/,(=
FORTRAN 026
GENERAL PURPOSE ... FIELD



```
 CLEAR STORAGE 1     ,008015,022026,030037,044,049,053053N000000N00001026
 CLEAR STORAGE 2     L068116,105106,110117B101/I9I#071029C029056B026/B001/0991,001/001117I0?
 BOOTSTRAP           ,008015,022029,036040,047054,061068,072/061039              ,0010011040
0                             HELLO WORLD.S 12/25/19 19:19:58                           -0398
0 SEQ PG LIN  LABEL  OP     OPERANDS                                           SFX CT  LOCN  INSTR
  101                JOB    HELLO WORLD.S 12/25/19 19:19:58
  102         *78901234|6789|12345678901234567890l|345678901234567890123456789012345678 90
  103         *
  104                CTL    6611       *6=16,000C;6=16,000T;1=OBJDECK;,1=MODADD
  105         *
  106         * CTL STATEMENT:
  107         * 6 = SIZE OF COMPILING MACHINE 16,000
  108         * 6 = SIZE OF TARGET MACHINE 16,000
  109         * 1 = PRINT OBJECT DECK
  110         * 1 = ADDRESS MODIFICATION INSTRUCTION PRESENT (TRUE ON 16,000)
  111         *
  112         *
  113         ***********************************************************************
  114         *  HELLO WORLD
  115         *      BY MARC VERDIELL
  116         *
  117         *       SETS UP A STANDARD 1401 ENVIRONMENT
  118         *       & PRINTS A LINE
  119         *
  120         *  VERSION: 1
  121         *  MAY 2019
```





IBM 1403 A and H Print Arrangment

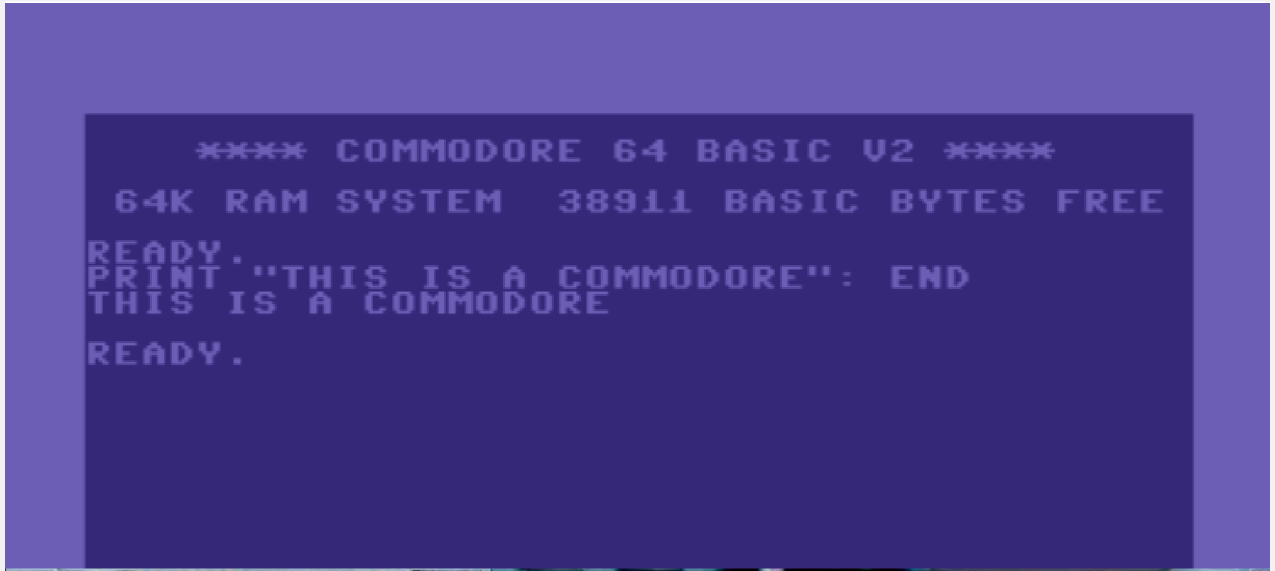
PRINT ARRANGEMENT		DEFINED CHARACTER	CARD CODE	BCD CODE	PRINT ARRANGEMENT		DEFINED CHARACTER	CARD CODE	BCD CODE
A	H				A	H			
		Blank		C	G	G	G	12-7	B A 4 2 1
.	.	.	12-3-8	B A 8 2 1	H	H	H	12-8	B A 8
⌑	)	⌑	12-4-8	C B A 8 4	I	I	I	12-9	C B A 8 1
		[Left Bracket (Special Character)	12-5-8	B A 8 4 1	-	-	! (Minus Zero)	11-0	B 8 2
		< Less Than (Special Character)	12-6-8	B A 8 4 2	J	J	J	11-1	C B 1
		⧧ Group Mark	12-7-8	C B A 8 4 2 1	K	K	K	11-2	C B 2
&	+	&	12	C B A	L	L	L	11-3	B 2 1
$	$	$	11-3-8	C B 8 2 1	M	M	M	11-4	C B 4
*	*	*	11-4-8	B 8 4	N	N	N	11-5	B 4 1
		] Right Bracket (Special Character)	11-5-8	C B 8 4 1	O	O	O	11-6	B 4 2
		; Semicolon (Special Character)	11-6-8	C B 8 4 2	P	P	P	11-7	C B 4 2 1
		Δ Delta (Mode Change)	11-7-8	B 8 4 2 1	Q	Q	Q	11-8	C B 8
-	-	-	11	B	R	R	R	11-9	B 8 1
/	/	/	0-1	C A 1	‡	‡	‡ Record Mark	0-2-8	A 8 2
,	,	,	0-3-8	C A 8 2 1	S	S	S	0-2	C A 2
%	(	%	0-4-8	A 8 4	T	T	T	0-3	A 2 1
		∨ Word Separator	0-5-8	C A 8 4 1	U	U	U	0-4	C A 4
		\ (Special Character)	0-6-8	C A 8 4 2	V	V	V	0-5	A 4 1
		⧺ Tape Segment Mark	0-7-8	A 8 4 2 1	W	W	W	0-6	A 4 2
‡	‡	ƀ (Special Character)		A	X	X	X	0-7	C A 4 2 1
#	=	#	3-8	8 2 1	Y	Y	Y	0-8	C A 8
@	'	@	4-8	C 8 4	Z	Z	Z	0-9	A 8 1
		: Colon (Special Character)	5-8	8 4 1	0	0	0	0	C 8 2
		> Greater Than (Special Character)	6-8	8 4 2	1	1	1	1	1
		√ Tape Mark	7-8	C 8 4 2 1	2	2	2	2	2
&	&	? (Plus Zero)	12-0	C B A 8 2	3	3	3	3	C 2 1
A	A	A	12-1	B A 1	4	4	4	4	4
B	B	B	12-2	B A 2	5	5	5	5	C 4 1
C	C	C	12-3	C B A 2 1	6	6	6	6	C 4 2
D	D	D	12-4	B A 4	7	7	7	7	4 2 1
E	E	E	12-5	C B A 4 1	8	8	8	8	8
F	F	F	12-6	C B A 4 2	9	9	9	9	C 8 1

IBM 1403, Models 2 and 7

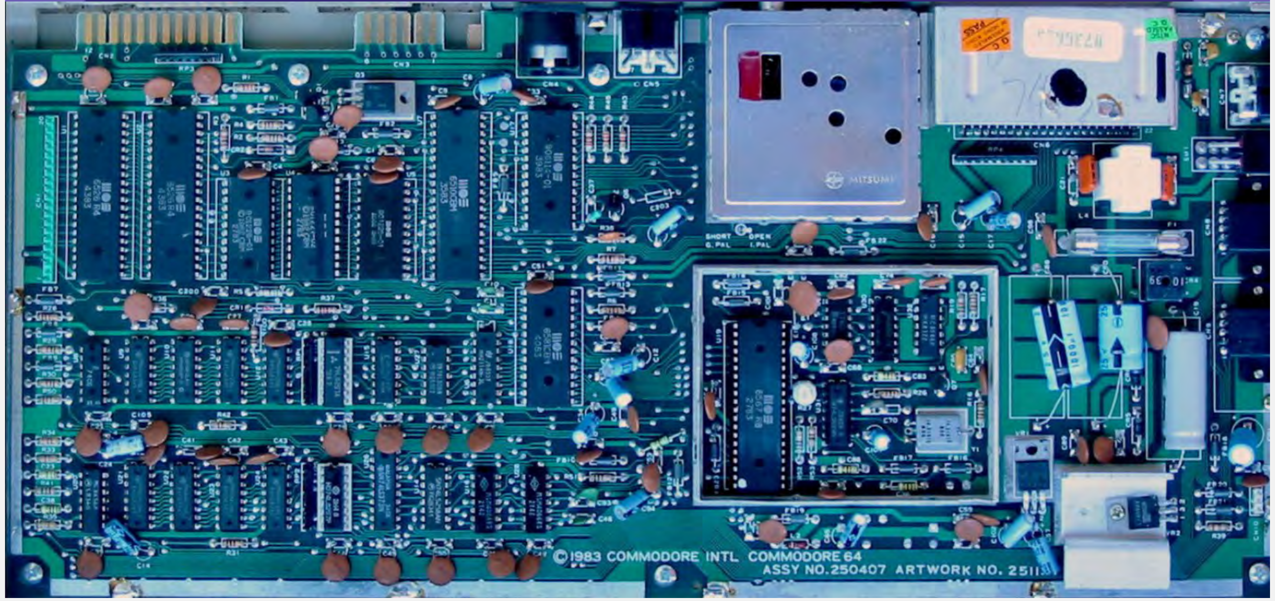
Print Arrangement	Special Characters
AN	& . ⌑ - $ * , % / # @ +
HN	& .) - $ * , (/ = ' +

Hardware differences – Structure, Purpose

BASIC – Commodore (a.k.a. C-64, 1980s) – table-top machines, used microchips



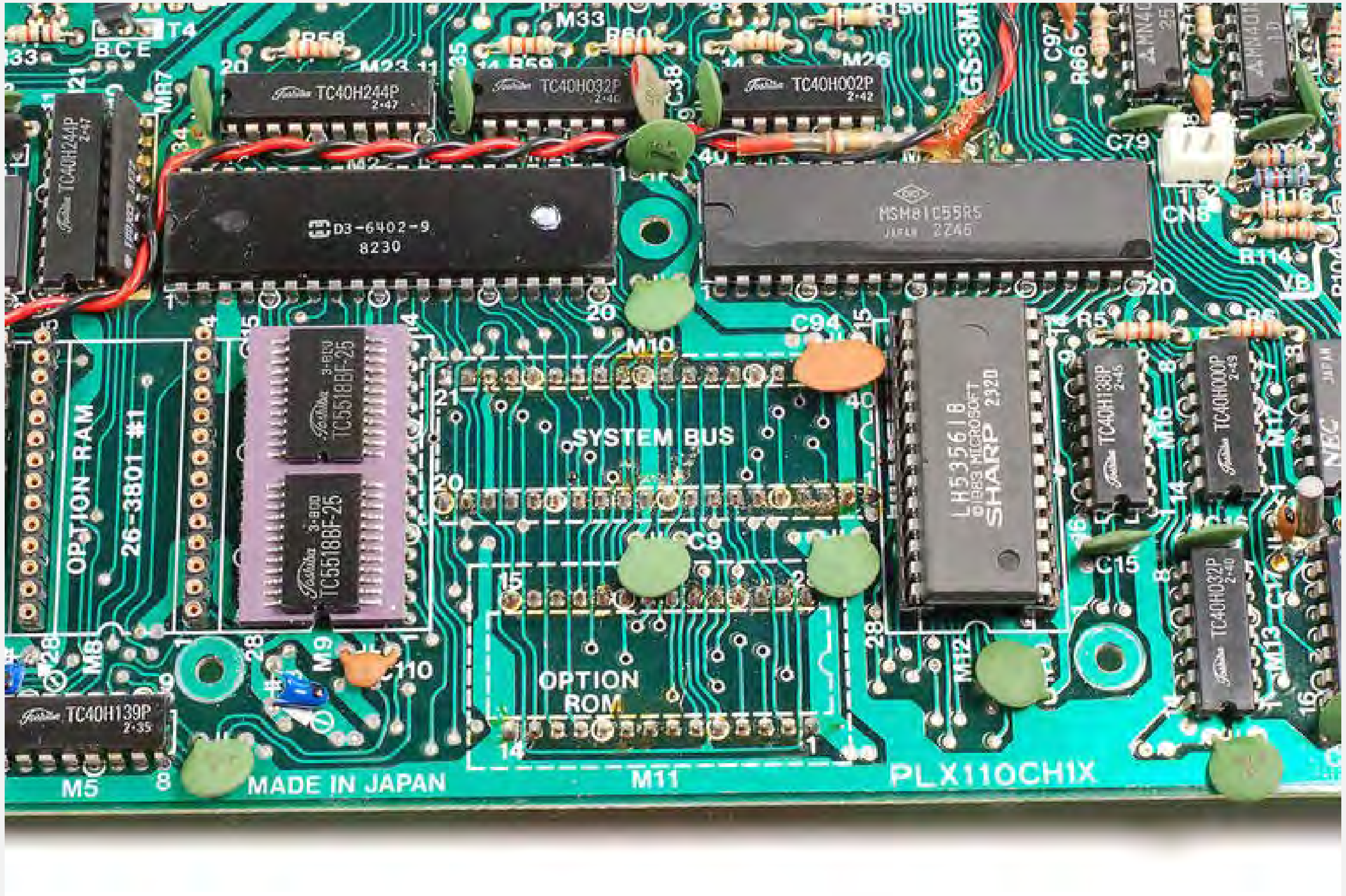
```
    **** COMMODORE 64 BASIC V2 ****

 64K RAM SYSTEM  38911 BASIC BYTES FREE

READY.
PRINT "THIS IS A COMMODORE": END
THIS IS A COMMODORE

READY.
```


©1983 COMMODORE INTL. COMMODORE 64
ASSY NO.250407 ARTWORK NO. 2511

Hardware differences – Structure, Purpose

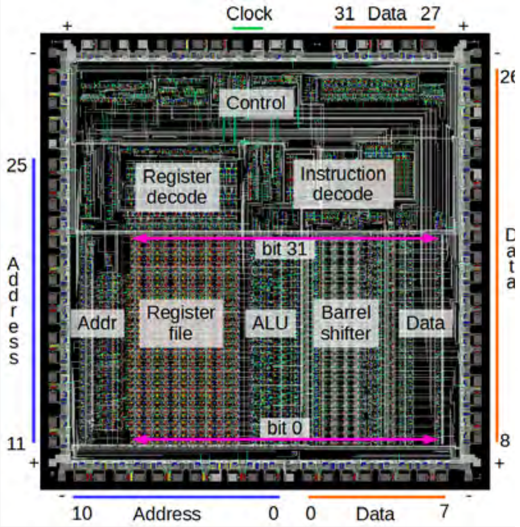
BASIC – Tandy 100 (a.k.a. TRS-80, mid-1980s) – table-top or portable machines, used microchips

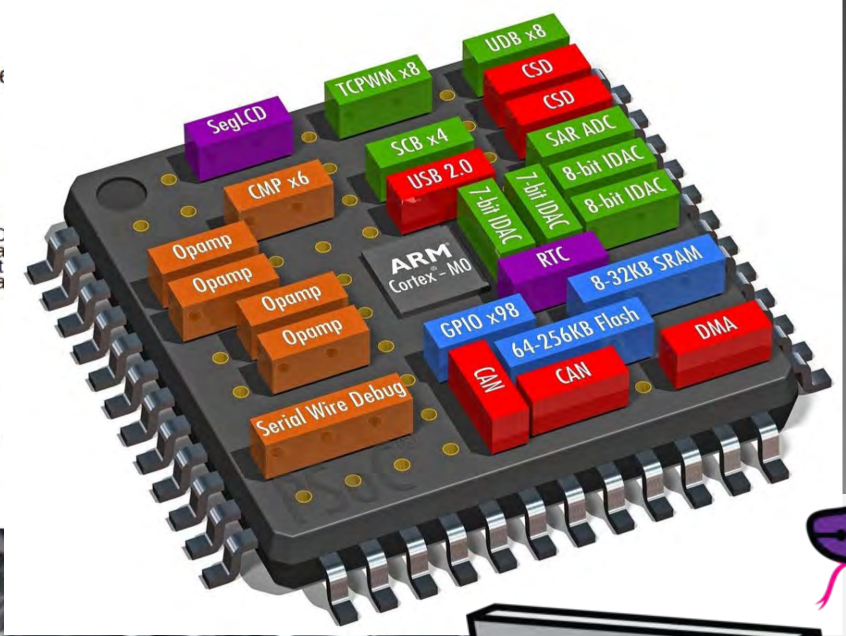
Jan 12,1908 Sat 12:01:54 (C)Microsoft
BASIC TEXT TELCOM ADDRSS
SCHEDL WIKI.DO PEDIA.DO
Select: _ 21401 Bytes free
Radio Shack
TRS-80
MODEL 100 PORTABLE COMPUTER

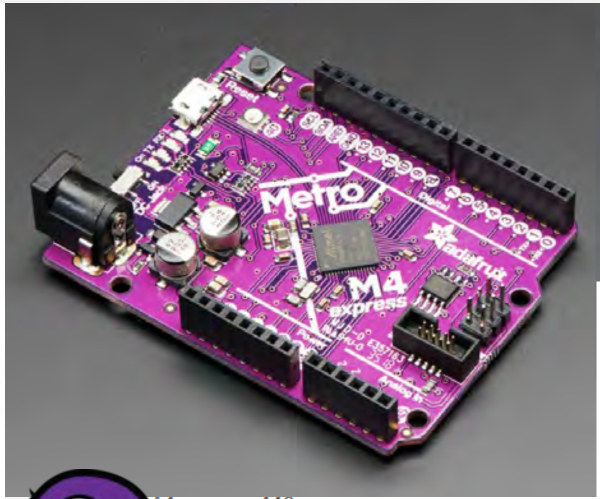


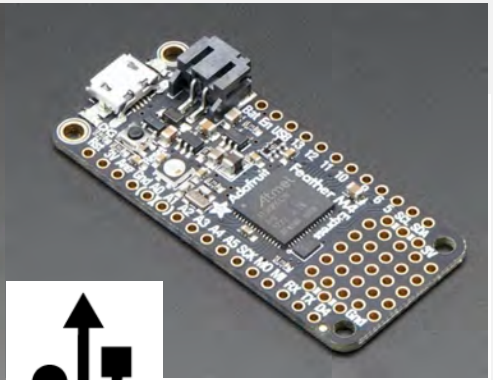
Hardware differences – Structure, Purpose

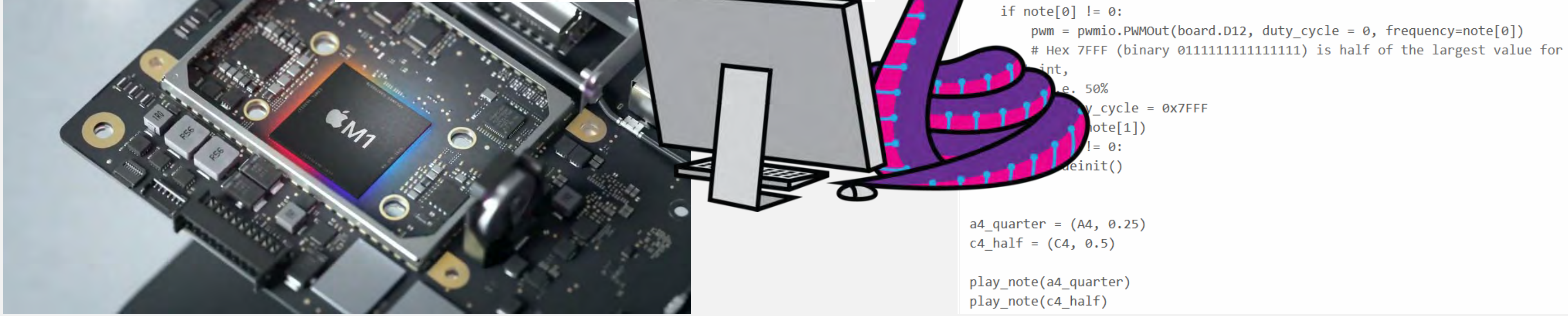
Python – multi-core microprocessors and ARM processors (2010's) – microcontrollers programmable by USB











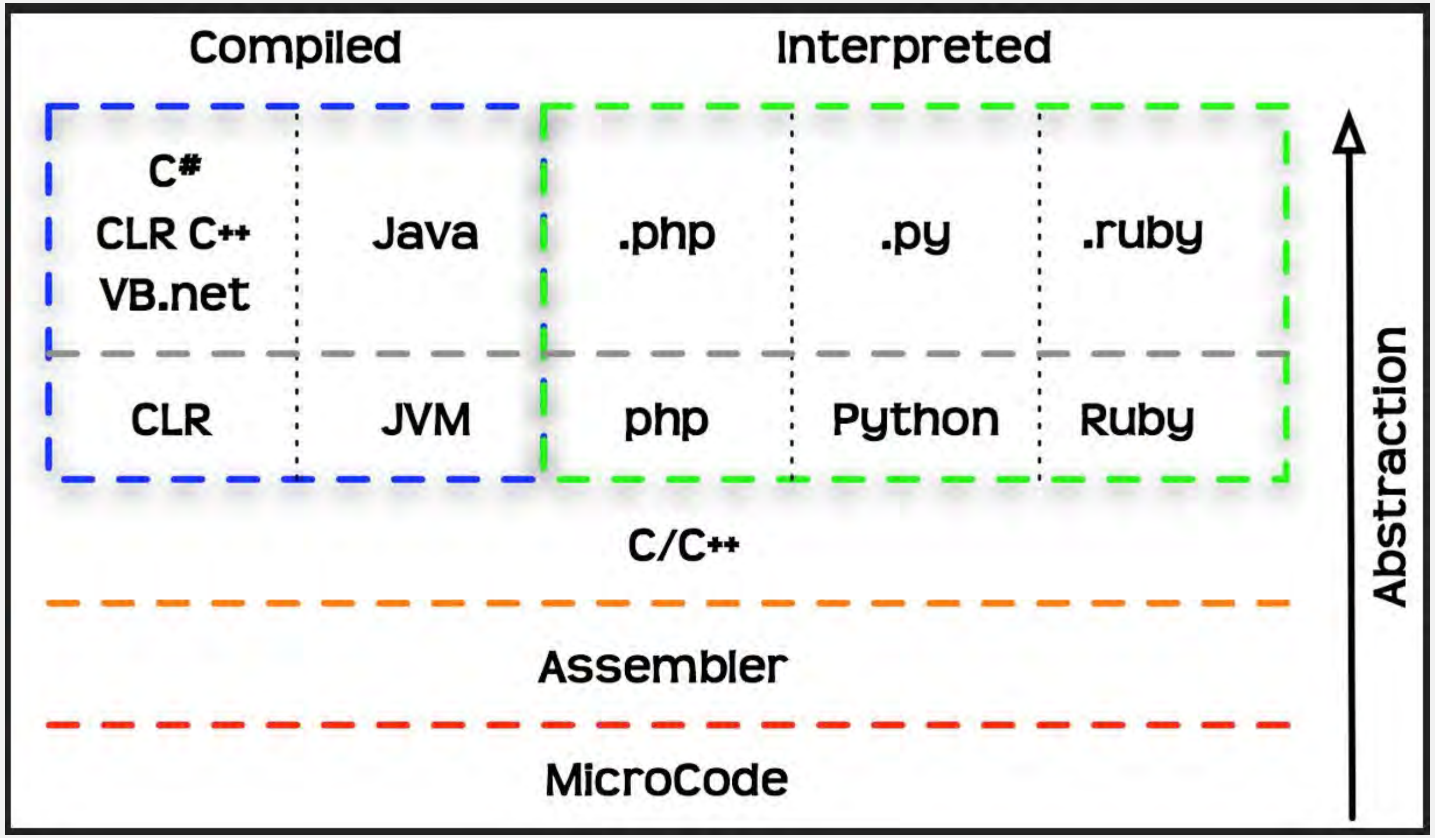
```
A4      = 440
_SH_4 = 466
4       = 493

def play_note(note):
    if note[0] != 0:
        pwm = pwmio.PWMOut(board.D12, duty_cycle = 0, frequency=note[0])
        # Hex 7FFF (binary 0111111111111111) is half of the largest value for
         int,
           e. 50%
            y_cycle = 0x7FFF
            ote[1])
            != 0:
           deinit()


a4_quarter = (A4, 0.25)
c4_half = (C4, 0.5)

play_note(a4_quarter)
play_note(c4_half)
```

“levels” of languages

- “levels” of abstraction
  - *complex metaphors*

- Compiled vs. Interpreted

- Control, efficiency, or ease-of-use



three traditional styles of language

- Math-based
  - *Fortran* *oldest, least-intuitive, designed for to perform math (Formula Translation)*
  - *MATLAB* *good for complexity, matrices*

- Object-based
  - *Go* *(Golang) similar to C, but uses safety features like "garbage collection"*
  - *Python* *high-level language, being used for building games and hardware programming*
  - *C++* *general purpose with low-level memory manipulation, in the C family*
  - *C#* *general purpose, used in Unity and robotics, in the C family*

- Specialized – Web, Data
  - *COBOL* *business-centered language, recently object-oriented*
  - *Javascript* *smaller version of Java, used for web widgets like TinkerCAD*
  - *HTML* *webpage coding usually in conjunction with XML, PHP and CSS*
  - *Ruby on Rails* *webpage/app coding aimed at server-side platforms*
  - *SQL* *database management, used in IoT applications (internet of things)*
  - *Hadoop* *"big data" management language*

what can python do

games, apps, data, art, and robots

using:
loops, variables, functions, plug-ins

applications

- Who uses Python?
  - *BitTorrent, FinTech, Intel, IBM, Pixar, Maya, PayPal*
  - *NSA, Bots, NASA, Netflix, Facebook, Spotify, Uber*
- Python & AI
  - *https://realpython.com/python-ai-neural-network/*
  - *https://pll.harvard.edu/course/cs50s-introduction-artificial-intelligence-python?delta=0*
  - *https://www.cuelogic.com/blog/role-of-python-in-artificial-intelligence*
- Graphic User Interfaces (GUIs)
  - *https://wiki.python.org/moin/GuiProgramming*
- Generative Art
  - *https://www.freecodecamp.org/news/how-to-create-generative-art-in-less-than-100-lines-of-code-d37f379859f/*
- Game Development
  - *Tron Game, pg64-65- https://wireframe.raspberrypi.org/issues/47*
- Robotics
  - *https://www.hackster.io/Odd_Jayy/widget-dragon-companion-bot-83c0c0*

IDE
Integrated Development Environment

- what is it? – an “all-inclusive resort” for your coding experience

  - *Write/Edit*
    - text editing
  - *Compile/Run*
    - translate into machine language and run your program
  - *Troubleshoot/Debug*
    - walk through your code line-by-line to make sure it will work
  - *Integrations and plugins*
    - add specific functionality for your project

IDE – *online = convenience*

online – free, easy, limited processing power

- Repl.it (free account, **Only 3 REPLs for free** as of Aug 26, 2024)
  - https://replit.com/
- Google Co-Lab for python (requires free google login account, ".ipynb")
  - https://colab.research.google.com
- PySplashLive – Browser-based Python and Jupyter
  - https://www.willingconsulting.com/pysplashlive.html
- Microsoft azure (1-year free trial)
  - https://azure.microsoft.com/en-us/develop/python/
- Others
  - https://python.codnex.net/
  - https://pythonandturtle.com/turtle
  - https://ide.geeksforgeeks.org/
  - https://www.onlinegdb.com/

IDE – *offline = power*

offline – free, complex install, harness computer power, plug-in option

VS Code (Visual Studio Code) – Microsoft

https://visualstudio.microsoft.com/vs/features/python/

https://code.visualstudio.com/docs/languages/python

Mu - https://codewith.mu/

Atom - https://atom.io/

Eclipse Theia - https://theia-ide.org

Python IDE - https://www.python.org/downloads/

Notepad++ - https://notepad-plus-plus.org/

vocabulary

- Know this: “all writing is re-writing”

- REPL (read-evaluate-print loop) – interactive language environment

- Function/Command – print vs ‘print’ vs PRINT

- Constants – fixed values

- Variables – values that can change
  - *Strings single or double quotation marks: >>> print(‘Howdy world’)*
    - Numbers
    - Letters

- Loops – repeat this action “_” many times
  - when – forever loop, keep doing this when this is true
  - for – repeat this “_” many times, then move on

- if/elif/else
  - if - if this is true, do this
  - elif – otherwise, do this
  - else - otherwise, do this and move on

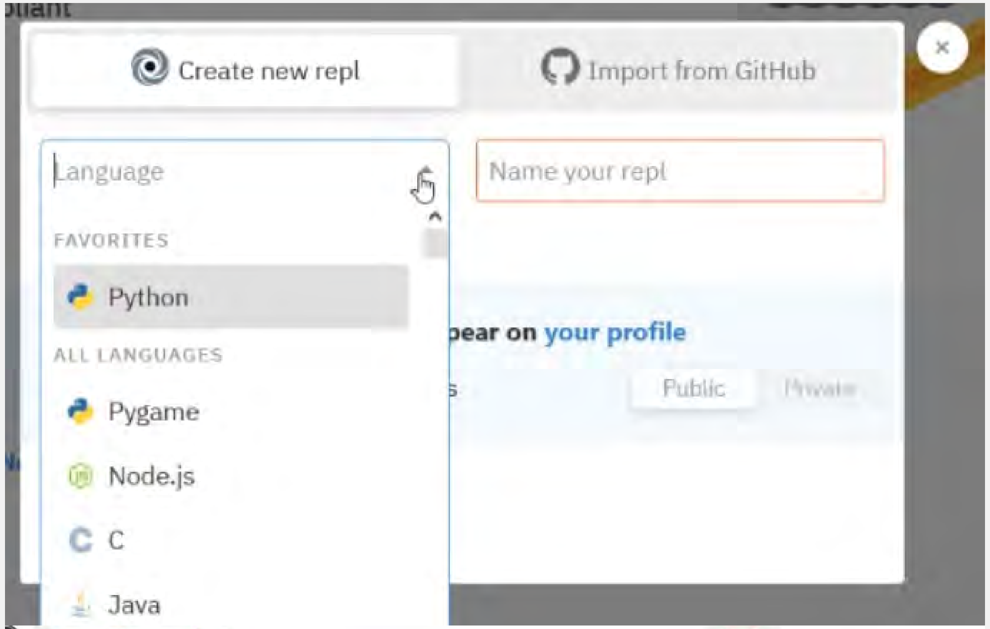
La Trahison des images
René Magritte, 1929

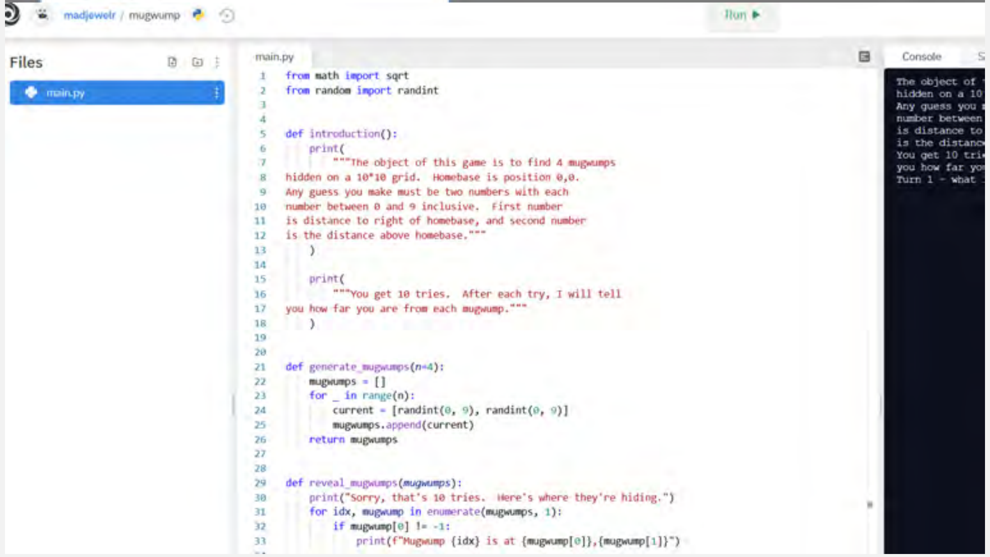
read-evaluate-print-loop

let’s explore python!

Direct your Chrome/Firefox to https://replit.com/

- Join with a free account to save your projects
- Start a new “python” project, exploring the interface
- Do some basic math and notations
- Strings and then join (concatenate) them
- Change attributes using *upper()* and *replace()*
- Boolean (True/False) operations ==, !=, <,>=
- if/elif/else
- **Only 3 REPLs for free** (as of Aug 26, 2024)

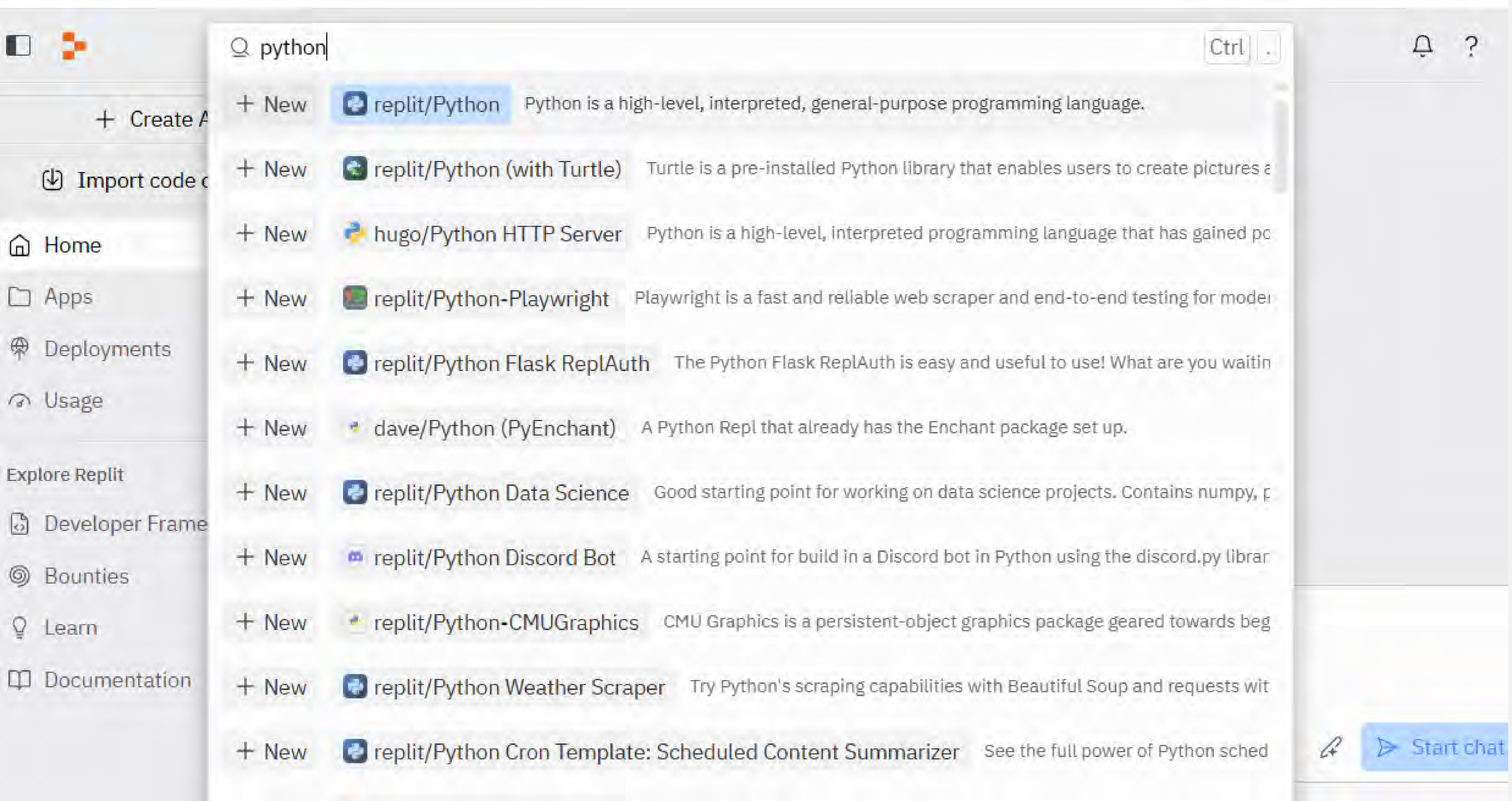




main.py

```python
from math import sqrt
from random import randint


def introduction():
    print(
        """The object of this game is to find 4 mugwumps
hidden on a 10*10 grid.  Homebase is position 0,0.
Any guess you make must be two numbers with each
number between 0 and 9 inclusive.  First number
is distance to right of homebase, and second number
is the distance above homebase."""
    )

    print(
        """You get 10 tries.  After each try, I will tell
you how far you are from each mugwump."""
    )


def generate_mugwumps(n=4):
    mugwumps = []
    for _ in range(n):
        current = [randint(0, 9), randint(0, 9)]
        mugwumps.append(current)
    return mugwumps


def reveal_mugwumps(mugwumps):
    print("Sorry, that's 10 tries.  Here's where they're hiding.")
    for idx, mugwump in enumerate(mugwumps, 1):
        if mugwump[0] != -1:
            print(f"Mugwump {idx} is at {mugwump[0]},{mugwump[1]}")
```

```
The object of
hidden on a 10
Any guess you
number between
is distance to
is the distanc
You get 10 tri
you how far yo
Turn 1 - what
```



python

+ New replit/Python Python is a high-level, interpreted, general-purpose programming language.

+ New replit/Python (with Turtle) Turtle is a pre-installed Python library that enables users to create pictures a

+ New hugo/Python HTTP Server Python is a high-level, interpreted programming language that has gained pc

+ New replit/Python-Playwright Playwright is a fast and reliable web scraper and end-to-end testing for moder

+ New replit/Python Flask ReplAuth The Python Flask ReplAuth is easy and useful to use! What are you waitin

+ New dave/Python (PyEnchant) A Python Repl that already has the Enchant package set up.

+ New replit/Python Data Science Good starting point for working on data science projects. Contains numpy, p

+ New replit/Python Discord Bot A starting point for build in a Discord bot in Python using the discord.py librar

+ New replit/Python-CMUGraphics CMU Graphics is a persistent-object graphics package geared towards beg

+ New replit/Python Weather Scraper Try Python's scraping capabilities with Beautiful Soup and requests wit

+ New replit/Python Cron Template: Scheduled Content Summarizer See the full power of Python sched

- Join with a free account to save your projects
- Search to start a new “python” project, exploring the interface (as of June 2025)

Only 3 REPLs for free (as of Aug 26, 2024)

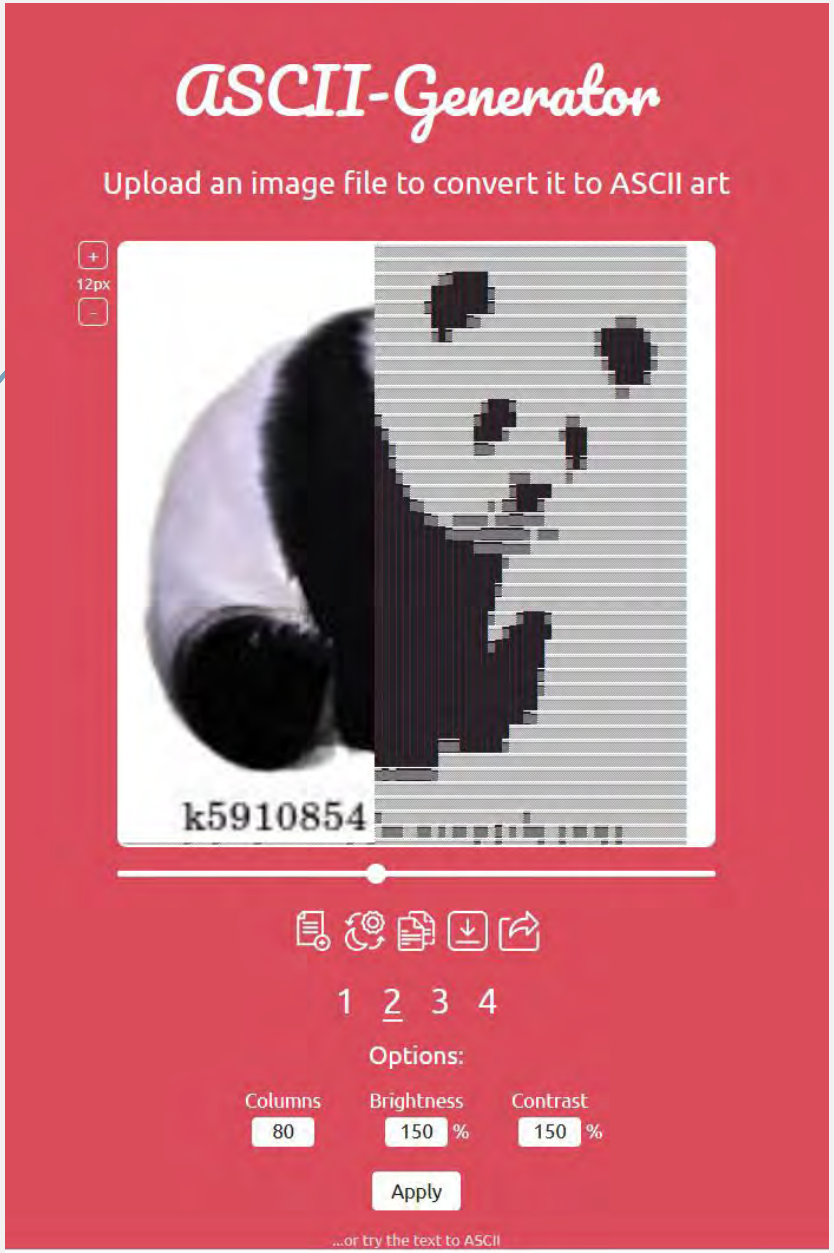
let's explore python!

Alternate online IDEs

- Coder.com
  - https://coder.com/pricing
- Python + MicroBit
  - https://python.microbit.org/v/3/reference
- Python Fiddle
  - https://python-fiddle.com/
- Python Playground
  - https://programiz.pro/ide/python
- Online Python (beta)
  - https://www.online-python.com/
- Geeks4Geeks – languages: Python, C/C++/C#, Java, Perl, Swift, R, Golang, HTML&JS, etc.
  - https://ide.geeksforgeeks.org/
- VS Code simulator – not sure how functional
  - https://vscode.dev/
- Github.dev – modify, navigate but not run github repository
  - https://docs.github.com/en/codespaces/the-githubdev-web-based-editor
  - https://github.com/github/dev

ascii-adventure

- ASCII graphics time
  - *Download jpg/png/gif*
  - *Drag to website https://ascii-generator.site/*
  - *Copy image ascii text into a “Notepad” file*
  - *Save as an “image.txt”*
- Drag “image.txt” into your program
- Let’s have your code display it!
  - *f = open(‘image.txt’, ‘r’)*
  - *file_contents = f.read()*
  - *print(file_contents)*
  - *f.close()*



ASCII-Generator

Upload an image file to convert it to ASCII art

Options:

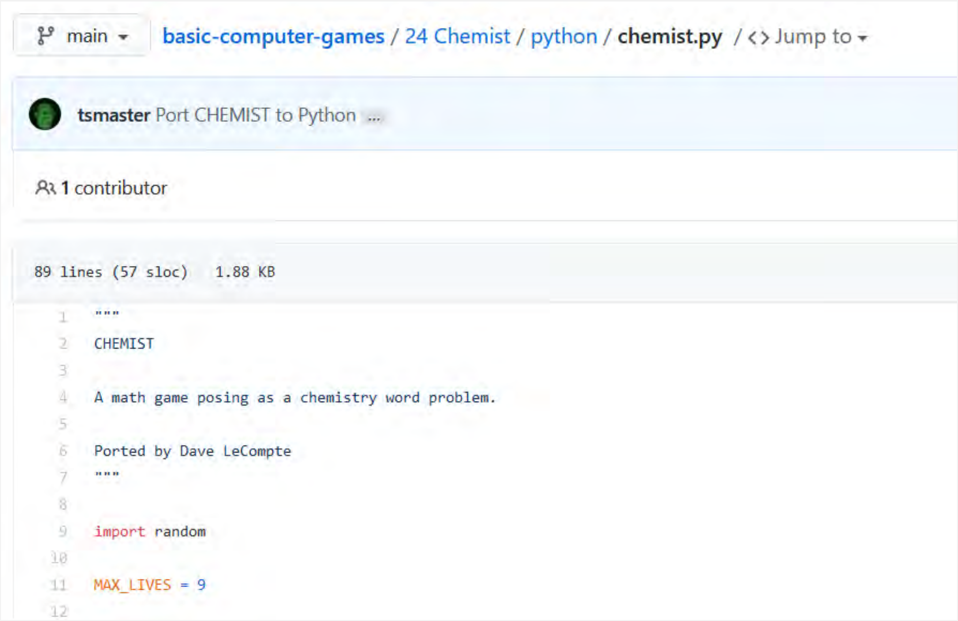
Columns 80

Brightness 150 %

Contrast 150 %

let's dive into "copy pasta"

- "Hello World"
- Select a BASIC program from the github archive
  - *https://github.com/coding-horror/basic-computer-games*
  - *"24 Chemist" - https://github.com/coding-horror/basic-computer-games/blob/main/24%20Chemist/python/chemist.py*
- Download/Copy Code
- Start a new "python" project in replit.com
- Paste the program and RUN
- Make some changes and explore



main
basic-computer-games / 24 Chemist / python / chemist.py / <> Jump to

tsmaster Port CHEMIST to Python ...

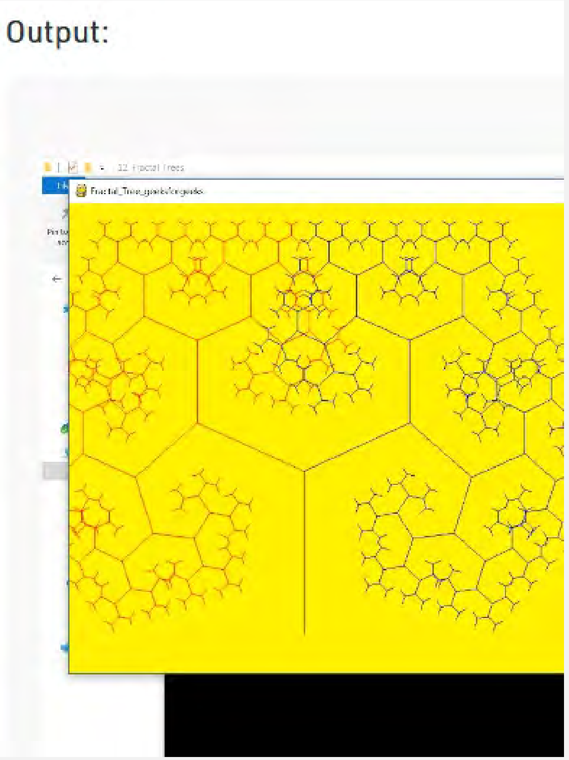
1 contributor

89 lines (57 sloc) 1.88 KB

```
"""
CHEMIST

A math game posing as a chemistry word problem.

Ported by Dave LeCompte
"""

import random

MAX_LIVES = 9

```

pygame “copy pasta”

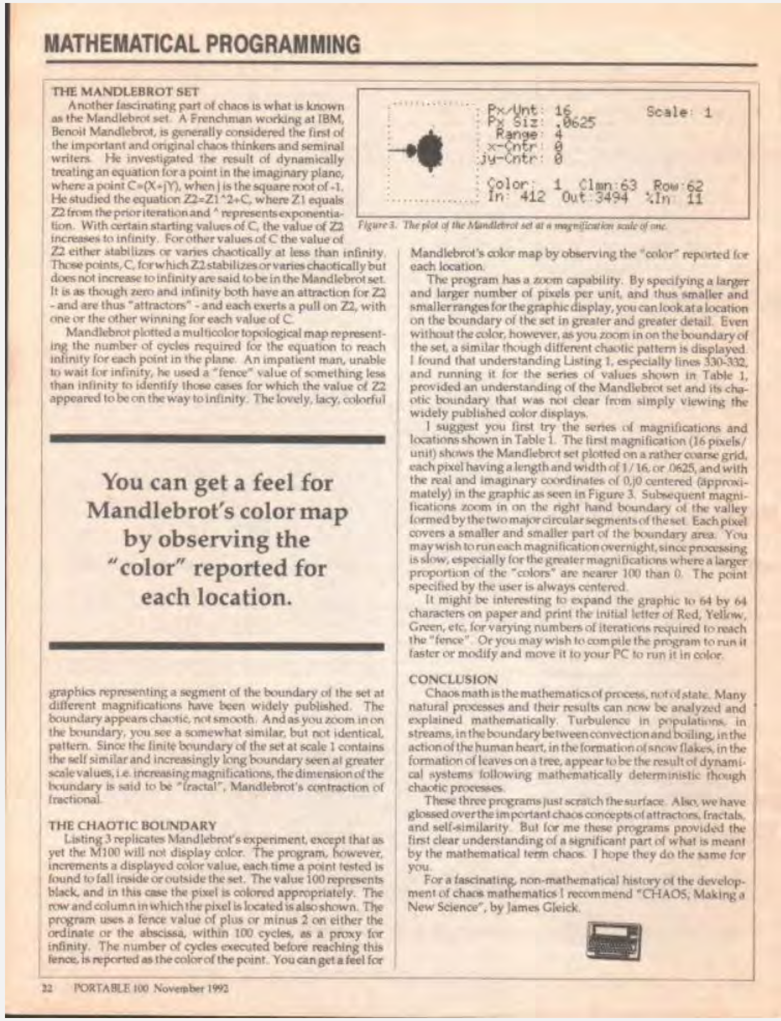
- Fractal tree output program
  - *https://www.geeksforgeeks.org/fractal-trees-python/*
- Download/Copy Code
- Start a new “pygame” project in replit.com
- Paste the program and RUN
- Make some changes and explore



Output:

homebrew game development

Open-source resources like Wireframe magazine or go retro with archive.org



MATHEMATICAL PROGRAMMING

THE MANDLEBROT SET

Another fascinating part of chaos is what is known as the Mandlebrot set. A Frenchman working at IBM, Benoit Mandlebrot, is generally considered the first of the important and original chaos thinkers and seminal writers. He investigated the result of dynamically treating an equation for a point in the imaginary plane, where a point C=(X+jY), when j is the square root of -1. He studied the equation Z2=Z1^2+C, where Z1 equals Z2 from the prior iteration and ^ represents exponentiation. With certain starting values of C, the value of Z2 increases to infinity. For other values of C the value of Z2 either stabilizes or varies chaotically at less than infinity. Those points, C, for which Z2 stabilizes or varies chaotically but does not increase to infinity are said to be in the Mandlebrot set. It is as though zero and infinity both have an attraction for Z2 - and are thus "attractors" - and each exerts a pull on Z2, with one or the other winning for each value of C.

Mandlebrot plotted a multicolor topological map representing the number of cycles required for the equation to reach infinity for each point in the plane. An impatient man, unable to wait for infinity, he used a "fence" value of something less than infinity to identify those cases for which the value of Z2 appeared to be on the way to infinity. The lovely, lacy, colorful graphics representing a segment of the boundary of the set at different magnifications have been widely published. The boundary appears chaotic, not smooth. And as you zoom in on the boundary, you see a somewhat similar, but not identical, pattern. Since the finite boundary of the set at scale 1 contains the self similar and increasingly long boundary seen at greater scale values, i.e. increasing magnifications, the dimension of the boundary is said to be "fractal", Mandlebrot's contraction of fractional.

```
Px/Unt: 16          Scale: 1
Px Siz: .0625
 Range: 4
x-Cntr: 0
jy-Cntr: 0

Color: 1  Clmn:63  Row:62
In: 412  Out:3494  %In 11
```

Figure 3. The plot of the Mandlebrot set at a magnification scale of one.

You can get a feel for Mandlebrot's color map by observing the "color" reported for each location.

THE CHAOTIC BOUNDARY

Listing 3 replicates Mandlebrot's experiment, except that as yet the M100 will not display color. The program, however, increments a displayed color value, each time a point tested is found to fall inside or outside the set. The value 100 represents black, and in this case the pixel is colored appropriately. The row and column in which the pixel is located is also shown. The program uses a fence value of plus or minus 2 on either the ordinate or the abscissa, within 100 cycles, as a proxy for infinity. The number of cycles executed before reaching this fence, is reported as the color of the point. You can get a feel for Mandlebrot's color map by observing the "color" reported for each location.

The program has a zoom capability. By specifying a larger and larger number of pixels per unit, and thus smaller and smaller ranges for the graphic display, you can look at a location on the boundary of the set in greater and greater detail. Even without the color, however, as you zoom in on the boundary of the set, a similar though different chaotic pattern is displayed. I found that understanding Listing 1, especially lines 330-332, and running it for the series of values shown in Table 1, provided an understanding of the Mandlebrot set and its chaotic boundary that was not clear from simply viewing the widely published color displays.

I suggest you first try the series of magnifications and locations shown in Table 1. The first magnification (16 pixels/unit) shows the Mandlebrot set plotted on a rather coarse grid, each pixel having a length and width of 1/16, or .0625, and with the real and imaginary coordinates of 0,j0 centered (approximately) in the graphic as seen in Figure 3. Subsequent magnifications zoom in on the right hand boundary of the valley formed by the two major circular segments of the set. Each pixel covers a smaller and smaller part of the boundary area. You may wish to run each magnification overnight, since processing is slow, especially for the greater magnifications where a larger proportion of the "colors" are nearer 100 than 0. The point specified by the user is always centered.

It might be interesting to expand the graphic to 64 by 64 characters on paper and print the initial letter of Red, Yellow, Green, etc, for varying numbers of iterations required to reach the "fence". Or you may wish to compile the program to run it faster or modify and move it to your PC to run it in color.

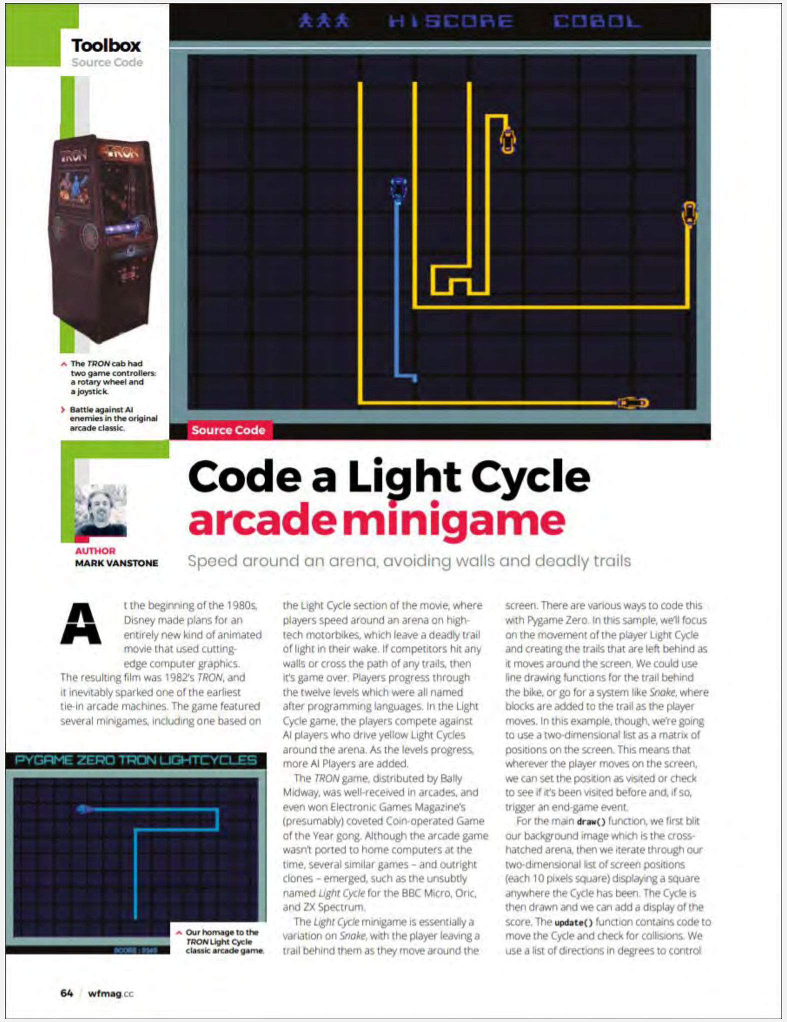
CONCLUSION

Chaos math is the mathematics of process, not of state. Many natural processes and their results can now be analyzed and explained mathematically. Turbulence in populations, in streams, in the boundary between convection and boiling, in the action of the human heart, in the formation of snow flakes, in the formation of leaves on a tree, appear to be the result of dynamical systems following mathematically deterministic though chaotic processes.

These three programs just scratch the surface. Also, we have glossed over the important chaos concepts of attractors, fractals, and self-similarity. But for me these programs provided the first clear understanding of a significant part of what is meant by the mathematical term chaos. I hope they do the same for you.

For a fascinating, non-mathematical history of the development of chaos mathematics I recommend "CHAOS; Making a New Science", by James Gleick.

22 PORTABLE 100 November 1992



Toolbox
Source Code

The TRON cab had two game controllers: a rotary wheel and a joystick.

Battle against AI enemies in the original arcade classic.

Source Code

Code a Light Cycle arcade minigame

AUTHOR
MARK VANSTONE

Speed around an arena, avoiding walls and deadly trails

At the beginning of the 1980s, Disney made plans for an entirely new kind of animated movie that used cutting-edge computer graphics. The resulting film was 1982's *TRON*, and it inevitably sparked one of the earliest tie-in arcade machines. The game featured several minigames, including one based on the Light Cycle section of the movie, where players speed around an arena on high-tech motorbikes, which leave a deadly trail of light in their wake. If competitors hit any walls or cross the path of any trails, then it's game over. Players progress through the twelve levels which were all named after programming languages. In the Light Cycle game, the players compete against AI players who drive yellow Light Cycles around the arena. As the levels progress, more AI Players are added.

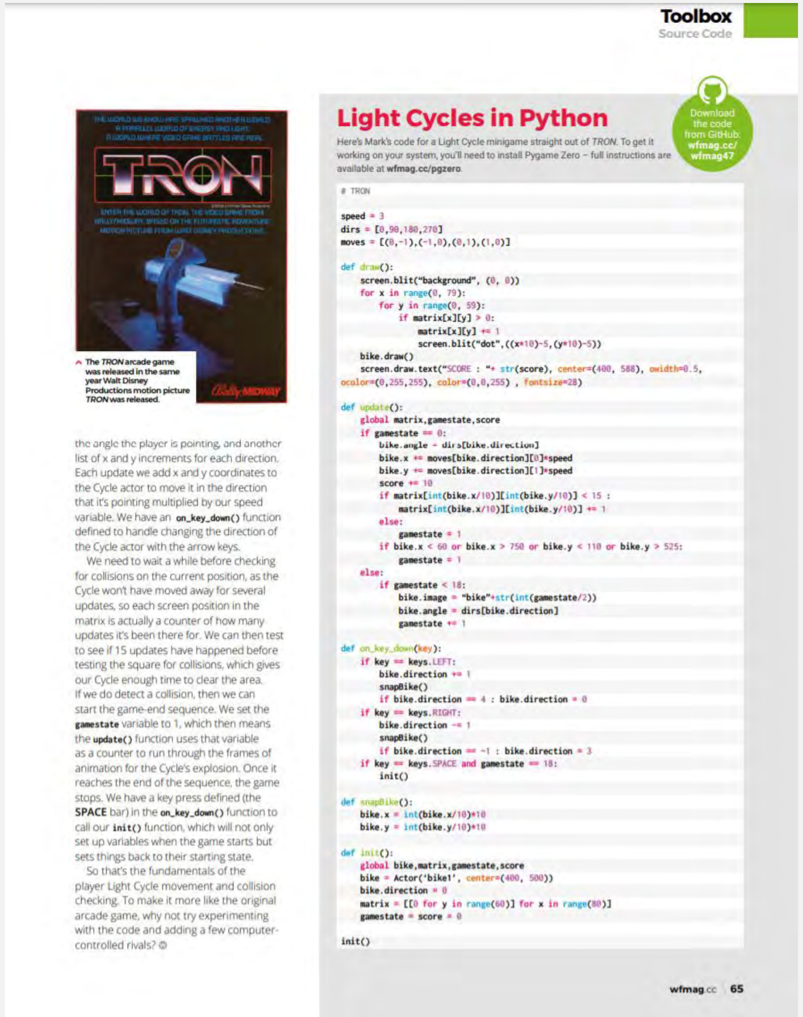
The *TRON* game, distributed by Bally Midway, was well-received in arcades, and even won Electronic Games Magazine's (presumably) coveted Coin-operated Game of the Year gong. Although the arcade game wasn't ported to home computers at the time, several similar games – and outright clones – emerged, such as the unsubtly named *Light Cycle* for the BBC Micro, Oric, and ZX Spectrum.

The *Light Cycle* minigame is essentially a variation on *Snake*, with the player leaving a trail behind them as they move around the screen. There are various ways to code this with Pygame Zero. In this sample, we'll focus on the movement of the player Light Cycle and creating the trails that are left behind as it moves around the screen. We could use line drawing functions for the trail behind the bike, or go for a system like *Snake*, where blocks are added to the trail as the player moves. In this example, though, we're going to use a two-dimensional list as a matrix of positions on the screen. This means that wherever the player moves on the screen, we can set the position as visited or check to see if it's been visited before and, if so, trigger an end-game event.

For the main `draw()` function, we first blit our background image which is the cross-hatched arena, then we iterate through our two-dimensional list of screen positions (each 10 pixels square) displaying a square anywhere the Cycle has been. The Cycle is then drawn and we can add a display of the score. The `update()` function contains code to move the Cycle and check for collisions. We use a list of directions in degrees to control

Our homage to the *TRON* Light Cycle classic arcade game.

64 wfmag.cc



Toolbox
Source Code

THE WORLD WE KNOW HAS SPAWNED ANOTHER WORLD
A PARALLEL WORLD OF ENERGY AND LIGHT.
A WORLD WHERE VIDEO GAME BATTLES ARE REAL
TRON
ENTER THE WORLD OF TRON, THE VIDEO GAME FROM BALLY/MIDWAY, BASED ON THE FUTURISTIC ADVENTURE MOTION PICTURE FROM WALT DISNEY PRODUCTIONS.
Bally MIDWAY

The *TRON* arcade game was released in the same year Walt Disney Productions motion picture *TRON* was released.

the angle the player is pointing, and another list of x and y increments for each direction. Each update we add x and y coordinates to the Cycle actor to move it in the direction that it's pointing multiplied by our speed variable. We have an `on_key_down()` function defined to handle changing the direction of the Cycle actor with the arrow keys.

We need to wait a while before checking for collisions on the current position, as the Cycle won't have moved away for several updates, so each screen position in the matrix is actually a counter of how many updates it's been there for. We can then test to see if 15 updates have happened before testing the square for collisions, which gives our Cycle enough time to clear the area. If we do detect a collision, then we can start the game-end sequence. We set the `gamestate` variable to 1, which then means the `update()` function uses that variable as a counter to run through the frames of animation for the Cycle's explosion. Once it reaches the end of the sequence, the game stops. We have a key press defined (the **SPACE** bar) in the `on_key_down()` function to call our `init()` function, which will not only set up variables when the game starts but sets things back to their starting state.

So that's the fundamentals of the player Light Cycle movement and collision checking. To make it more like the original arcade game, why not try experimenting with the code and adding a few computer-controlled rivals?

Light Cycles in Python

Here's Mark's code for a Light Cycle minigame straight out of *TRON*. To get it working on your system, you'll need to install Pygame Zero – full instructions are available at **wfmag.cc/pgzero**.

Download the code from GitHub: **wfmag.cc/wfmag47**

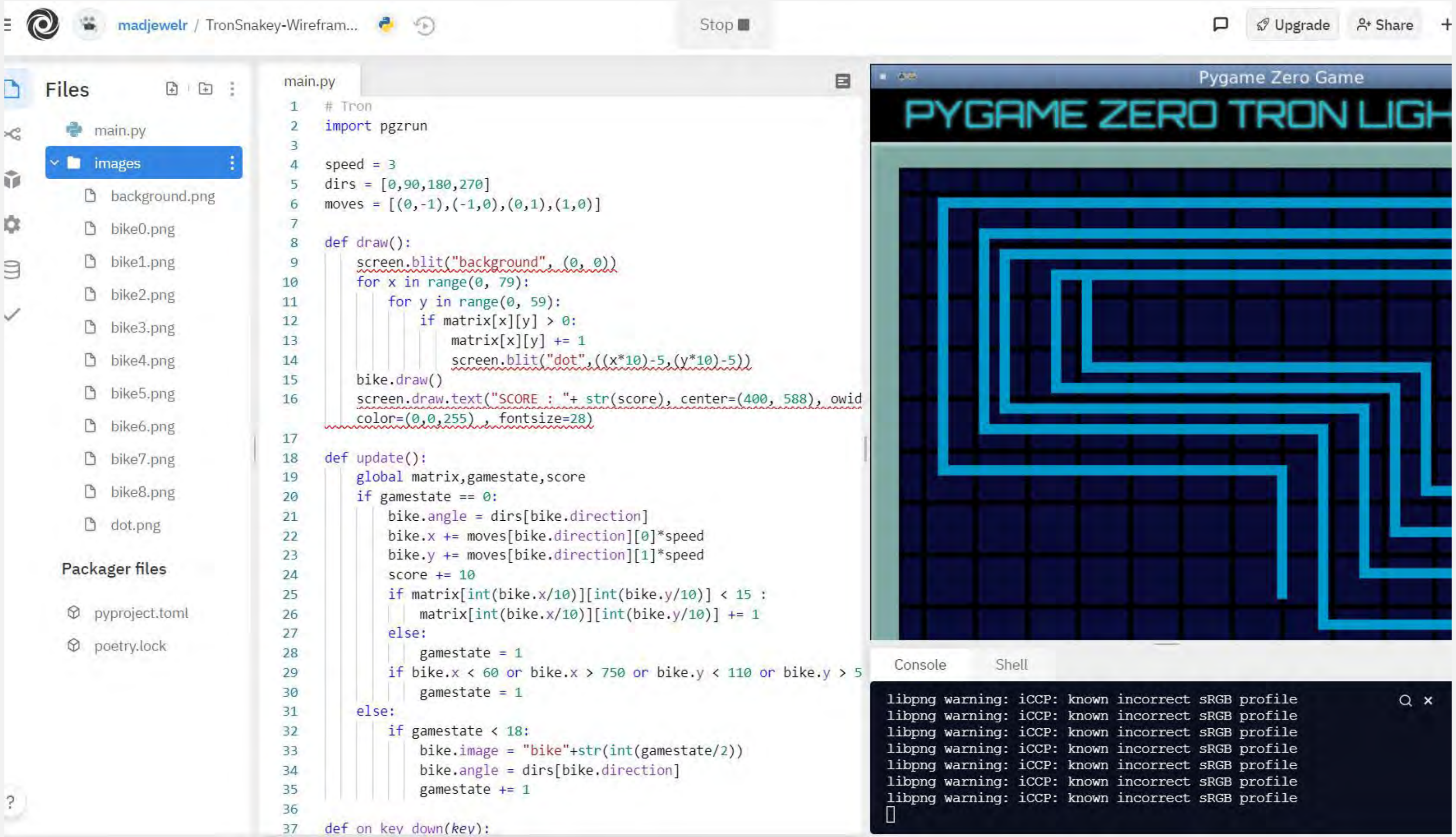
```python
# TRON

speed = 3
dirs = [0,90,180,270]
moves = [(0,-1),(-1,0),(0,1),(1,0)]

def draw():
    screen.blit("background", (0, 0))
    for x in range(0, 79):
        for y in range(0, 59):
            if matrix[x][y] > 0:
                matrix[x][y] += 1
                screen.blit("dot",((x*10)-5,(y*10)-5))
    bike.draw()
    screen.draw.text("SCORE : "+ str(score), center=(400, 588), owidth=0.5,
ocolor=(0,255,255), color=(0,0,255) , fontsize=28)

def update():
    global matrix,gamestate,score
    if gamestate == 0:
        bike.angle - dirs[bike.direction]
        bike.x += moves[bike.direction][0]*speed
        bike.y += moves[bike.direction][1]*speed
        score += 10
        if matrix[int(bike.x/10)][int(bike.y/10)] < 15 :
            matrix[int(bike.x/10)][int(bike.y/10)] += 1
        else:
            gamestate = 1
        if bike.x < 60 or bike.x > 750 or bike.y < 110 or bike.y > 525:
            gamestate = 1
    else:
        if gamestate < 18:
            bike.image = "bike"+str(int(gamestate/2))
            bike.angle = dirs[bike.direction]
            gamestate += 1

def on_key_down(key):
    if key == keys.LEFT:
        bike.direction += 1
        snapBike()
        if bike.direction == 4 : bike.direction = 0
    if key == keys.RIGHT:
        bike.direction -= 1
        snapBike()
        if bike.direction == -1 : bike.direction = 3
    if key == keys.SPACE and gamestate == 18:
        init()

def snapBike():
    bike.x = int(bike.x/10)*10
    bike.y = int(bike.y/10)*10

def init():
    global bike,matrix,gamestate,score
    bike = Actor('bike1', center=(400, 500))
    bike.direction = 0
    matrix = [[0 for y in range(60)] for x in range(80)]
    gamestate = score = 0

init()
```

wfmag.cc 65

BASIC – Portable 100 magazine (1992)
https://archive.org/details/Portable100MagazineNov1992/page/n19/mode/2up

Wireframe #47 (2021)
https://wireframe.raspberrypi.org/issues/47

pygame – tron/centipede!



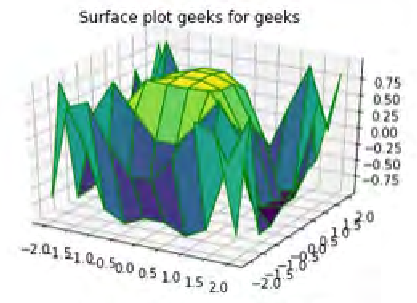
```python
# Tron
import pgzrun

speed = 3
dirs = [0,90,180,270]
moves = [(0,-1),(-1,0),(0,1),(1,0)]

def draw():
    screen.blit("background", (0, 0))
    for x in range(0, 79):
        for y in range(0, 59):
            if matrix[x][y] > 0:
                matrix[x][y] += 1
                screen.blit("dot",((x*10)-5,(y*10)-5))
    bike.draw()
    screen.draw.text("SCORE : "+ str(score), center=(400, 588), owid
    color=(0,0,255) , fontsize=28)

def update():
    global matrix,gamestate,score
    if gamestate == 0:
        bike.angle = dirs[bike.direction]
        bike.x += moves[bike.direction][0]*speed
        bike.y += moves[bike.direction][1]*speed
        score += 10
        if matrix[int(bike.x/10)][int(bike.y/10)] < 15 :
            matrix[int(bike.x/10)][int(bike.y/10)] += 1
        else:
            gamestate = 1
        if bike.x < 60 or bike.x > 750 or bike.y < 110 or bike.y > 5
            gamestate = 1
    else:
        if gamestate < 18:
            bike.image = "bike"+str(int(gamestate/2))
            bike.angle = dirs[bike.direction]
            gamestate += 1

def on_key_down(key):
```

```
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
```

Grab files from Wireframe #47 – https://github.com/Wireframe-Magazine/Wireframe-47

Utilize “pygame” within https://replit.com/ and add folder of images

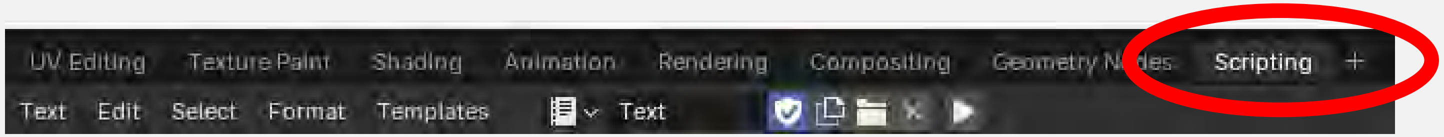
3D with Python!

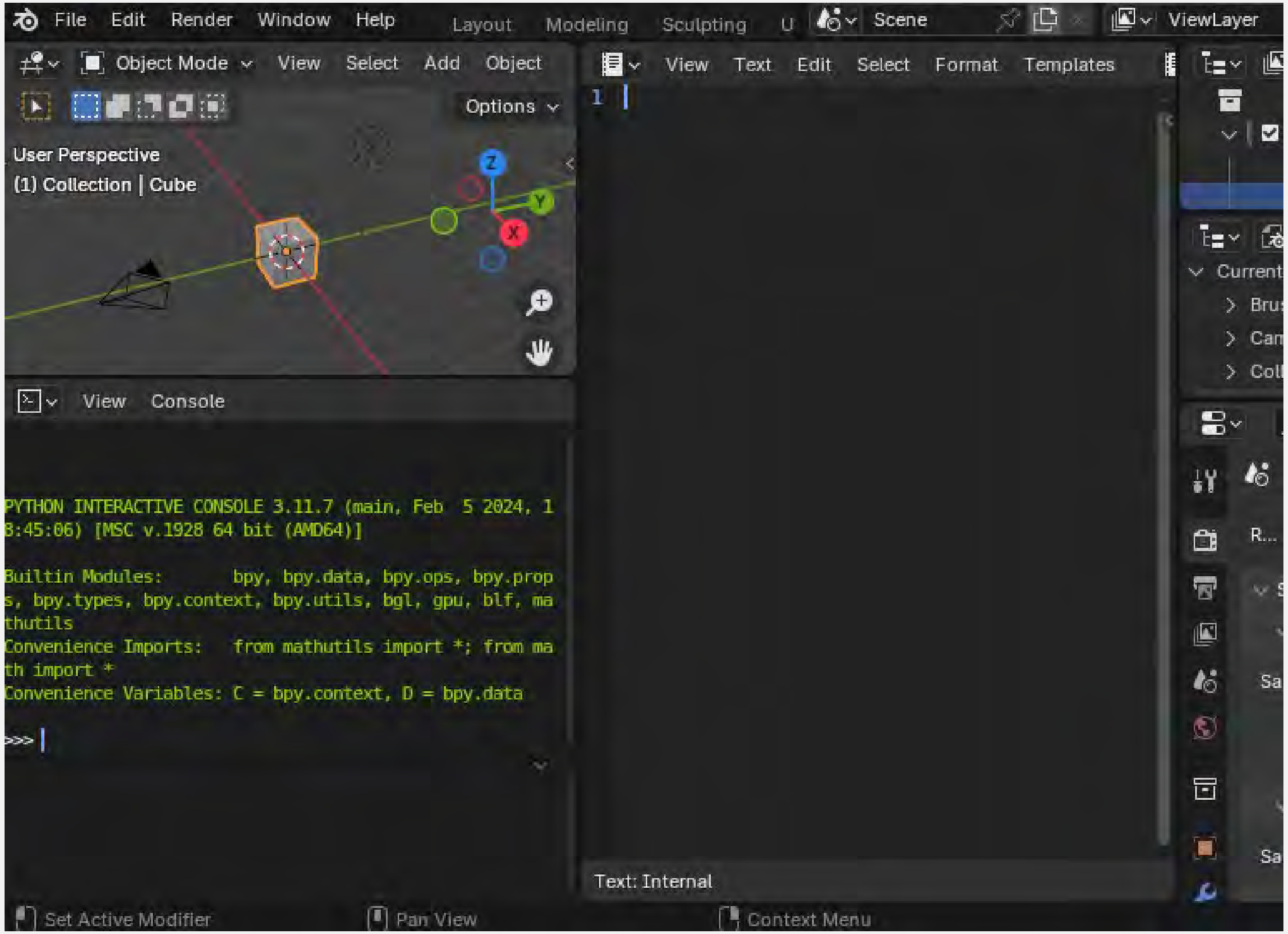


Surface plot using matplotlib library

- Rhino with Python
  - https://developer.rhino3d.com/guides/rhinopython/
- 3D graphics using standard python library
  - https://medium.com/quick-code/3d-graphics-using-the-python-standard-library-99914447760c
- Plotting 3D graphs using python's "Matplotlib" – reminiscent of MATLAB
  - https://www.geeksforgeeks.org/three-dimensional-plotting-in-python-using-matplotlib/
- 3D modeling and transporting those models into Minecraft
  - https://av.tib.eu/media/21228
- Python and SCAD work together to make 3D printable models
  - https://jccraig.medium.com/yes-you-can-use-python-for-3d-design-and-printing-508b1791c863
- New MADCAD (pymadcad) – use python scripts to directly make triangle meshes
  - https://madcad.netlify.app/
  - https://github.com/jimy-byerley/pymadcad

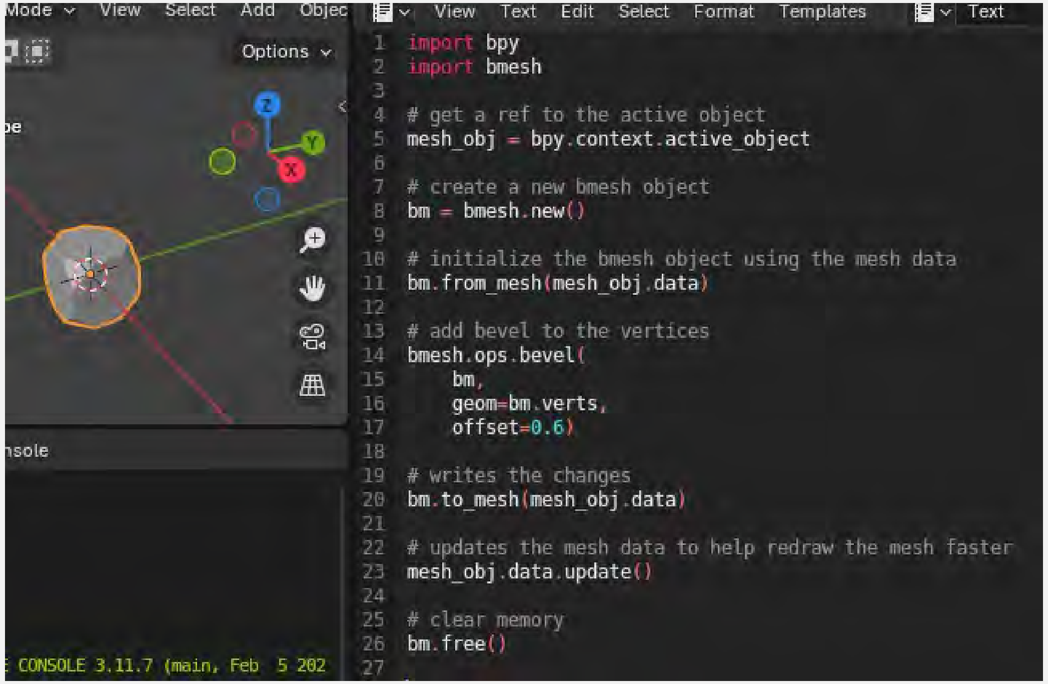
Blender + python scripts (explore!)





```
PYTHON INTERACTIVE CONSOLE 3.11.7 (main, Feb  5 2024, 1
8:45:06) [MSC v.1928 64 bit (AMD64)]

Builtin Modules:       bpy, bpy.data, bpy.ops, bpy.prop
s, bpy.types, bpy.context, bpy.utils, bgl, gpu, blf, ma
thutils
Convenience Imports:   from mathutils import *; from ma
th import *
Convenience Variables: C = bpy.context, D = bpy.data

>>>
```

- Sample bevel script



```python
import bpy
import bmesh

# get a ref to the active object
mesh_obj = bpy.context.active_object

# create a new bmesh object
bm = bmesh.new()

# initialize the bmesh object using the mesh data
bm.from_mesh(mesh_obj.data)

# add bevel to the vertices
bmesh.ops.bevel(
    bm,
    geom=bm.verts,
    offset=0.6)

# writes the changes
bm.to_mesh(mesh_obj.data)

# updates the mesh data to help redraw the mesh faster
mesh_obj.data.update()

# clear memory
bm.free()
```

```
CONSOLE 3.11.7 (main, Feb  5 202
```

Blender + python scripts (explore!)

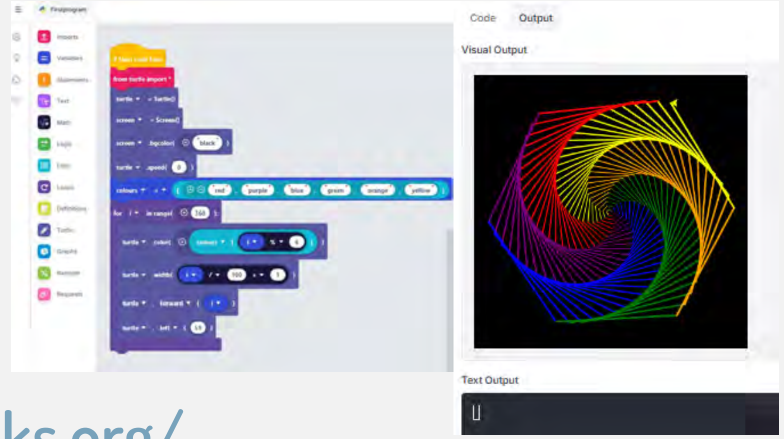
Run Python in Blender to generate models– “bpy”

- https://docs.blender.org/api/current/info_quickstart.html
- https://docs.blender.org/manual/en/dev/editors/preferences/interface.html - bpy-types-preferencesview-show-developer-ui
- https://docs.blender.org/manual/en/dev/editors/preferences/interface.html - bpy-types-preferencesview-show-tooltips-python
- https://docs.blender.org/api/current/info_quickstart.html

Walkthroughs

- https://www.youtube.com/watch?v=TFQMNcTj5Jw
- https://www.youtube.com/watch?v=N3U2noAHgBo
- https://www.youtube.com/watch?v=wWTAQP7-ZUQ

resources – Python



```
# Start code here
from turtle import *
turtle = Turtle()
screen = Screen()
screen.bgcolor("black")
turtle.speed(0)
colours = ["red", "purple", "blue", "green", "orange", "yellow"]
for i in range(360):
    turtle.color(colours[i % 6])
    turtle.width(i / 100 + 1)
    turtle.forward(i)
    turtle.left(59)
```

Block-based Python tool – https://edublocks.org/

source: https://www.youtube.com/watch?v=ukroW9F2cGs&list=PL8uoeex94UhE1CbtkDK4hevp2lBif57Nq&index=9

MIT Course: https://www.youtube.com/playlist?list=PLUl4u3cNGP62A-ynp6v6-LGBCzeH3VAQB

https://www.freecodecamp.org/news/python-code-examples-sample-script-coding-tutorial-for-beginners/

https://codecombat.com/

https://www.edx.org/course?search_query=python

https://www.codecademy.com/learn/learn-python

https://stackoverflow.com/

DIY translate BASIC games to Python (then run on a microcontroller – MicroPython):

- https://www.youtube.com/watch?v=82Rb3HbkD_E

resources – Python community

PyCon – tons of conferences, some are more field-specific

- https://us.pycon.org/2024/schedule/
- https://pycon.blogspot.com/2022/05/pycon-us-2022-recap-and-recording.html
- https://us.pycon.org/2020/online/#tutorials ; https://www.youtube.com/c/PyConUS
- Python for Finances - https://tinyurl.com/xdmta5c6

Conferences! – many are virtual

- ReplCon - April 2, 2022 - https://blog.replit.com/
- All over the world: https://pycon.blogspot.com/
- https://ai.withthebest.com/ , Sept 14, 2022
- https://www.python.org/events/python-events/1085/

Python Discord server - https://discord.gg/python

Coding Discord servers - https://www.youtube.com/watch?v=7caAk9akfuI

Harvard's CS50 – extensive free computer science course (many languages)

- https://www.edx.org/learn/computer-science/harvard-university-cs50-s-introduction-to-computer-science
- https://cs50.harvard.edu/x/2024/, https://cs50.harvard.edu/x/2024/zoom/
- https://ide.cs50.io/

resources – more Python community

PyTexas - https://www.youtube.com/pytexas
PyTexas 25 – https://www.youtube.com/watch?v=VzCYbXEz0a4&list=PL0MRiRrXAvRiSmPn_LDdhDbtZwu6g4xct
PyTexas 24 – https://www.youtube.com/watch?v=Ww2-Cw9qnmE&list=PL0MRiRrXAvRjMAfx42eiokiAmfclUX-6S&index=12
PyTexas 23 – https://www.youtube.com/playlist?list=PL0MRiRrXAvRhiru4h8fVF987v5tdLlr1X

EuroPython Conference 2024 – July 8-14, https://ep2024.europython.eu/

PyConUS – https://us.pycon.org/, https://pycon.blogspot.com/
PyconUS 2025 – https://www.youtube.com/watch?v=flnVc2Ke-bw&list=PL2Uw4_HvXqvb98mQjN0-rYQjdDxJ_hcrs
PyconUS 2024 – https://www.youtube.com/watch?v=pNVn3FPi5PU&list=PL2Uw4_HvXqvYhjub9bw4uDAmNtprgAvlJ&index=110

Sample List of Regional/International PyCons

- SanFran Bay – https://pybay.com/talklist/
- Japan – https://2022.pycon.jp/en/
- Italy – https://pycon.it/en
- Taiwan – https://tw.pycon.org/2022/en-us

resources – starter projects/blocks/flowcharts

Python packages for different use-cases

https://www.python.org/download/alternatives/

Python programming books

- https://www.marktechpost.com/2024/04/02/top-python-programming-books-to-read-in-2024/

Python Projects for beginners

https://nedbatchelder.com/text/kindling.html

“Think Python” book – free

https://allendowney.github.io/ThinkPython/

PyOhio 2020

Lightning Talk: https://www.youtube.com/watch?v=hzABPEzM-pE&list=PL2k6bbM_wgjtGSzPXzUzP3AfVO-o4imbB&index=23

more resources – Python listicles

newsletters

- https://www.python.org/psf/newsletter/
- https://pycoders.com/
- https://pybit.es/newsletter/
- https://realpython.com/
- https://www.kdnuggets.com/ (DataCamp has a free week of courses around Feb+Aug+Nov)
- https://pyscript.net/
- https://mailchi.mp/035388afb48a/pytexas-community
- https://www.adafruitdaily.com/

listicle with resources

- https://www.freecodecamp.org/news/what-is-python-used-for-10-coding-uses-for-the-python-programming-language/

listicles – uses of Python

- https://statanalytica.com/blog/uses-of-python/
- https://www.educba.com/uses-of-python/
- https://www.bitdegree.org/tutorials/what-is-python-used-for/

even more resources – Python

Python & Excel - https://www.myonlinetraininghub.com/python-in-excel

Tutorial - http://anh.cs.luc.edu/python/hands-on/3.1/

Data management/processing - https://datacarpentry.org/

Scientific Programming -
https://help.uis.cam.ac.uk/service/support/training/downloads/course-files/programming-student-files/python-courses/pythonab/

NASA Archives+Python: https://www.youtube.com/watch?v=flvTQ2iZscU

PyGames
https://pygame-zero.readthedocs.io/en/stable/
https://wireframe.raspberrypi.org/issues/47
* https://wireframe.raspberrypi.com/books/code-the-classics1

even more ASCII-fun

Retro-fun -> JPG - ASCII conversion sites

https://ascii-generator.site/
https://manytools.org/hacker-tools/convert-images-to-ascii-art/
https://www.ascii-art-generator.org/

ASCII Art Archives

https://www.asciiart.eu/
https://asciiart.website/

ASCII with p5.js

https://www.youtube.com/watch?v=55iwMYv8tGI

resources – AI/ML + Python and more

Machine Learning/AI Python libraries

- https://www.marktechpost.com/2024/03/30/top-ten-python-libraries-for-machine-learning-and-deep-learning-in-2024/
- TensorFlow
- PyTorch
- Scikit-learn
- Keras
- XGBoost
- LightGBM
- JAX
- FastAI
- Hugging Face Transformers
- OpenCV

GPU-accelerated Computing – CuPy

- https://cupy.dev/
- Talk at PyBay2019 - https://www.youtube.com/watch?v=_AKDqw6li58

make Python slither (robots)

- Using Python to control hardware – Robotics, Microcontrollers/Sensors

MicroPython - https://micropython.org/

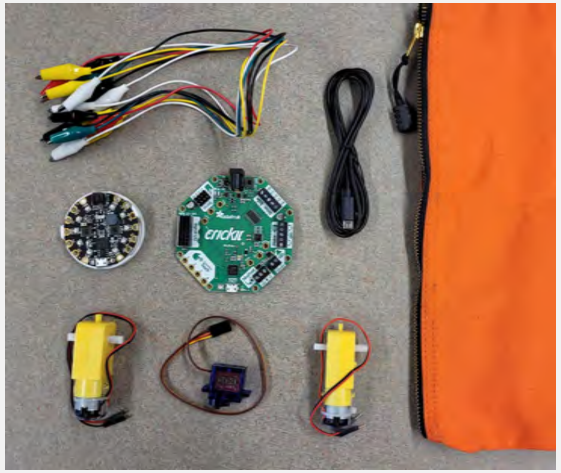
Circuit Python - https://circuitpython.org/

Adafruit – Hobby/professional electronics and programming

- Discord - https://adafru.it/discord
- Circuit Python Newsletter - https://blog.adafruit.com/tag/newsletters/
- Youtube - https://www.youtube.com/channel/UCpOlOeQjj7EsVnDh3zuCgsA

using Makerspace's *Electro*, *Cutebot*, and *Robo* kits

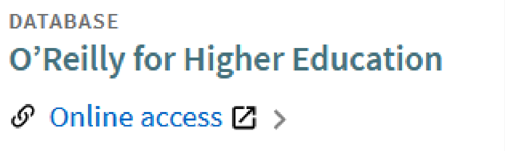
- Arrange a free appointment time to work with a kit (30min minimum)
- Have fun!
- Brainstorm by exploring Adafruit's hundreds of learning guides
  - *https://learn.adafruit.com/guides/beginner*
  - Adafruit Discord – *https://adafru.it/discord*



moving forward

- The Staff and Student Assistants in the Makerspace are here to help!

- There are O'Reilly video and text tutorials available to help learn the more advanced software (Python, MakeCode, Arduino, etc.) through Library Databases: Search for "oreilly":



DATABASE

O'Reilly for Higher Education

Online access

  - *Click "Online access"@*
  - *https://ttu-primo.hosted.exlibrisgroup.com/permalink/f/19r5ufk/01TTU_ALMA61293358880002611*

- CS50 – Fall 2024 Lecture 6 – Python - https://www.youtube.com/live/0eNc5lJfZFM

- 'maker' magazines
  - *MagPi Magazine - https://magpi.raspberrypi.com/issues/*
  - *Hackspace Magazine – https://hackspace.raspberrypi.com/issues*
  - *Make: Magazine - https://makezine.com/*

projects galore

project-sharing websites (some require membership)

https://learn.adafruit.com/
https://adafruit-playground.com
https://projecthub.arduino.cc/
https://www.hackster.io/
https://www.instructables.com/
https://hackaday.io/
https://make.co/
https://github.com/adafruit/circuitpython-weekly-newsletter-archive
https://magpi.raspberrypi.com/issues/128/

Support/Forums/Discord

- CircuitPython/Adafruit
  - *http://adafru.it/discord*

thank you!

please share your projects and progress!

‘Make with Python’ with Instructor Sean Scully

review this workshop here:

https://ttu.libwizard.com/f/workshop-eval-24-25_emerging_tech

Lead Administrator – sean.scully@ttu.edu

Assoc. Librarian – jake.syma@ttu.edu

Makerspace – make@ttu.edu

Director/Librarian – ryan.cassidy@ttu.edu

Workshops – https://guides.library.ttu.edu/make