CMPSC 100 JANUARY 2021

The future is a bunch of '80s polygons

WELCOME & COURSE INFORMATION

- Instructor: Prof. Luman
- Technical Leader: Dylan Holland
- Course meets: 9:30a - 12:30p, M-F
- My office hours: 1:00 - 3:00p, M-F
  - Expect Dylan's office hours schedule in the next 24.3 hours

WELCOME & COURSE INFORMATION

- This class will move quickly
  - Each day == 1 week
- Main methods of instruction:
  - “Sandboxing”
  - Worksheets
  - Weekly labs
  - A course project
- We will “sandbox” together via YouTube each day
  - Except Fridays – that’s lab

WELCOME & COURSE INFORMATION

- Daily class (M-R):
  - 9:30 - 11:00(ish) Work
  - 11:00 - 11:10(ish) Break
  - 11:10 - 12:30 Work
- Occasionally, I may direct you to work on something during our working time and reconvene the class to discuss it
- Lab (F <- not yr grade):
  - 9:30 - 12:30
    - Dedicated solely to work time

WELCOME & COURSE INFORMATION

- At the beginning of the week, you will receive a “repository” (a.k.a. “repo”) of the week’s work
  - You can work ahead
  - You can work at the course’s pace
- This work is all due on Sunday night (~11:59p)
  - Rly, who’s up that late?
    - Me
    - You should be sleeping
- Preface: it seems like a lot
  - It is, but we’ll work iteratively

WELCOME & COURSE INFORMATION

I have a thing that seems obvious, but I must repeat it:

You are allowed to work together -- fact.
However, don’t copy/paste or type out exact code that another student or online forum provides.

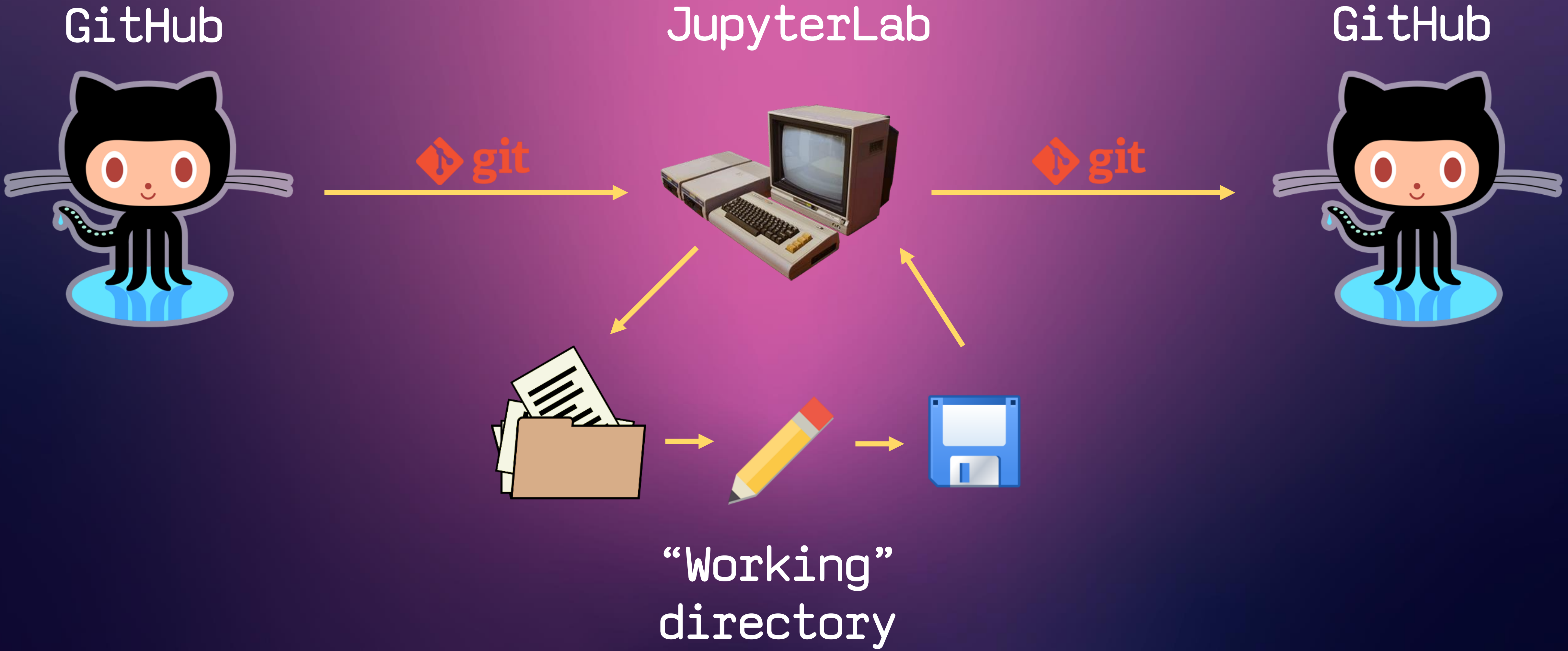
COURSE TOOLS: SLACK

- Slack is our main form of communication
- You can direct message me pretty much any time; I’m probably there
  - I don’t know what that says about me
- If you haven’t signed up already:
  - https://chomp.link/join-slack
- If nothing else, emojis

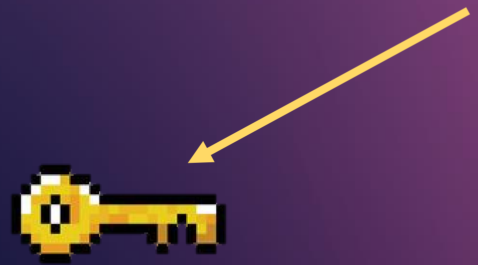
COURSE TOOLS: JUPYTERHUB

- We will use this platform this semester
  - Future CMPSC courses have a different approach
- I manage the server
  - Come to me with technical issues
- You need to use your GitHub to log in
  - Most of you have already done this
  - If you get a 403 error, I need
    to add you to the access list
    or fix my spelling

COURSE TOOLS: GITHUB



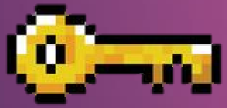
GETTING STARTED: SSH



We need to create a key

- This will secure our communication between the course Jupyter and GitHub
- It's often referred to as an SSH key
  - I made a 10+ minute video on it (linked in the assignment)
    - I don't know how it turned out to be 10 minutes

GETTING STARTED: SSH



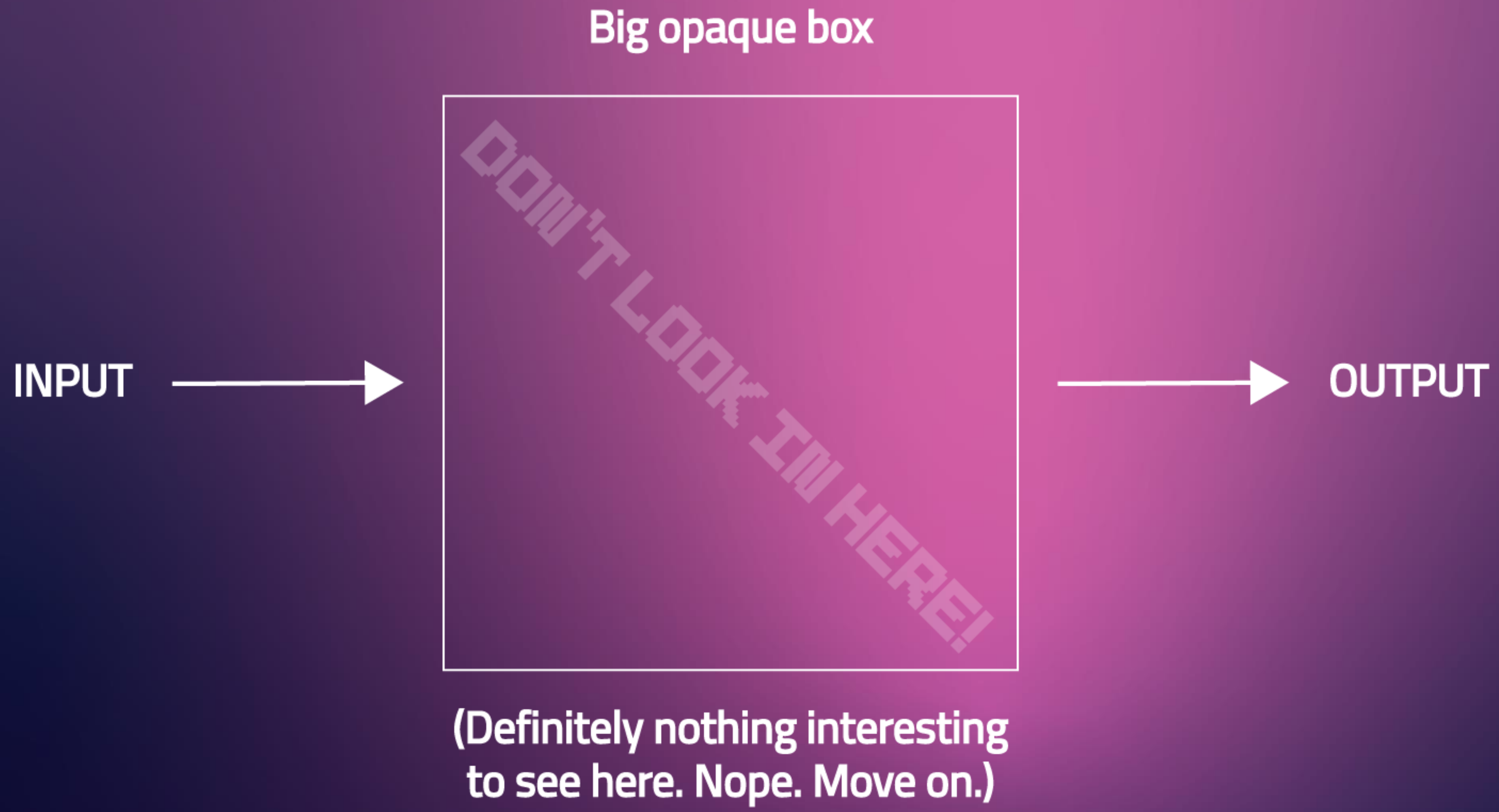
```
ssh-keygen -t rsa -b 4096 -C "YOUR ALLEGHENY EMAIL"
```

LET'S TAKE A BREAK.

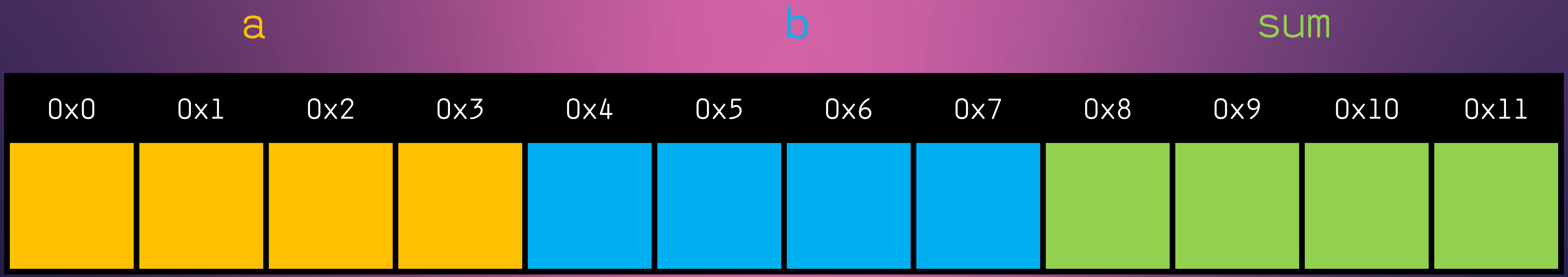
That was mostly the professor talking.

GETTING STARTED: TERMINAL

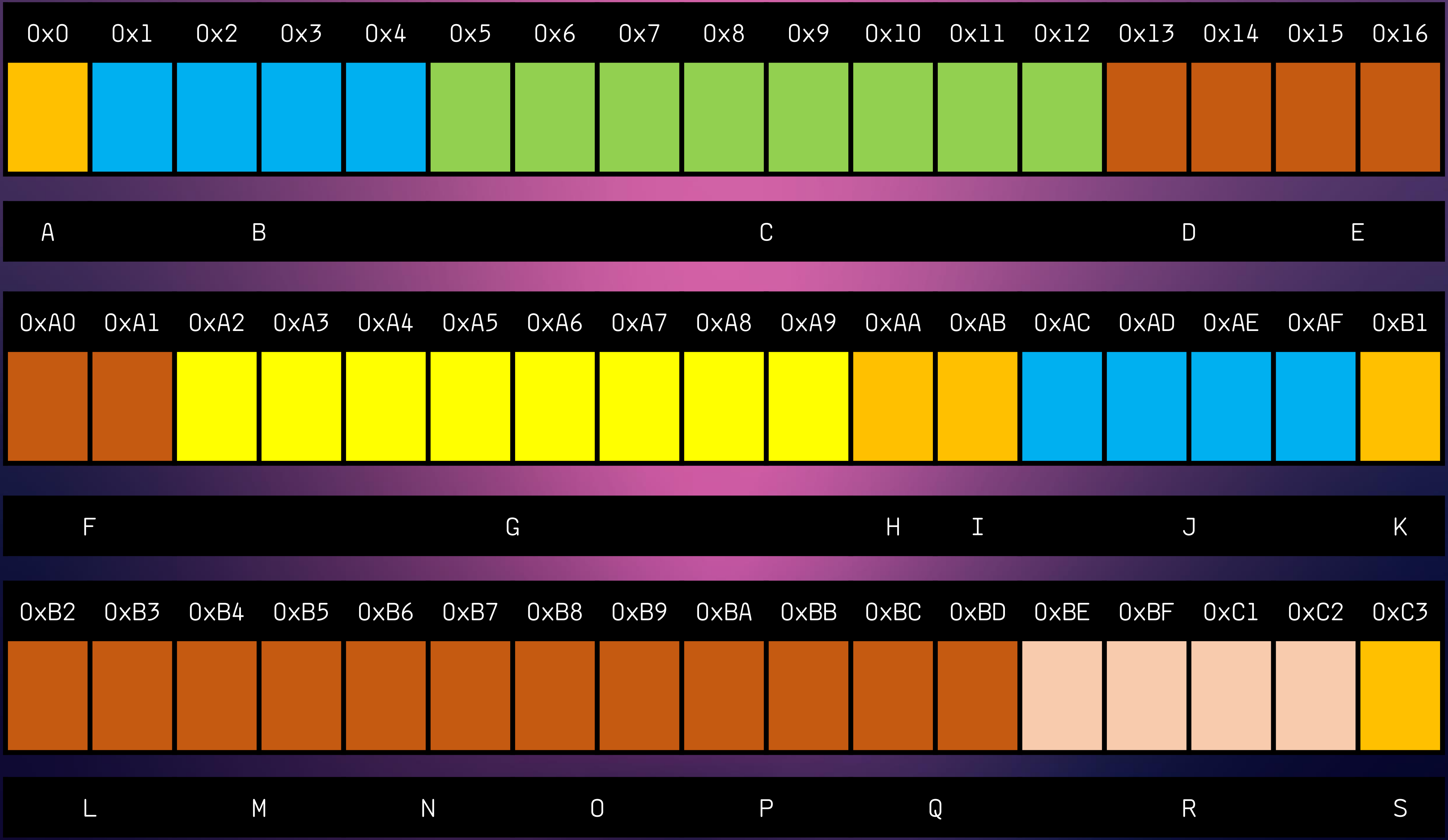
- Called a “terminal” because it’s really the “end” of something.
- Serves as an area to quickly request and direct computational processes
- Usually represented by a GUI (Graphical User Interface) which involves pointy-clicky operations
    - YOUR MOUSE CANNOT HELP YOU HERE


Big opaque box
INPUT
OUTPUT
DON'T LOOK IN HERE!
(Definitely nothing interesting to see here. Nope. Move on.)

ASSIGNMENTS AND VARIABLES



Programs essentially move variables around in memory space, performing operations on them.


0x0 0x1 0x2 0x3 0x4 0x5 0x6 0x7 0x8 0x9 0x10 0x11 0x12 0x13 0x14 0x15 0x16
A B C D E
0xA0 0xA1 0xA2 0xA3 0xA4 0xA5 0xA6 0xA7 0xA8 0xA9 0xAA 0xAB 0xAC 0xAD 0xAE 0xAF 0xB1
F G H I J K
0xB2 0xB3 0xB4 0xB5 0xB6 0xB7 0xB8 0xB9 0xBA 0xBB 0xBC 0xBD 0xBE 0xBF 0xC1 0xC2 0xC3
L M N O P Q R S

- Programs use **expressions**, which
  - They store in memory using **variables** as convenient names
  - They evaluate to reduce to the simplest form
  - Are either:
    - Assignments
    - Function calls

Assignments handle data types

- integers
- floating point numbers
- strings
- …
- Other things we’re going to ignore for now

Whole numbers

Integer

1, 2, 1000, -1000, 10…

Why.

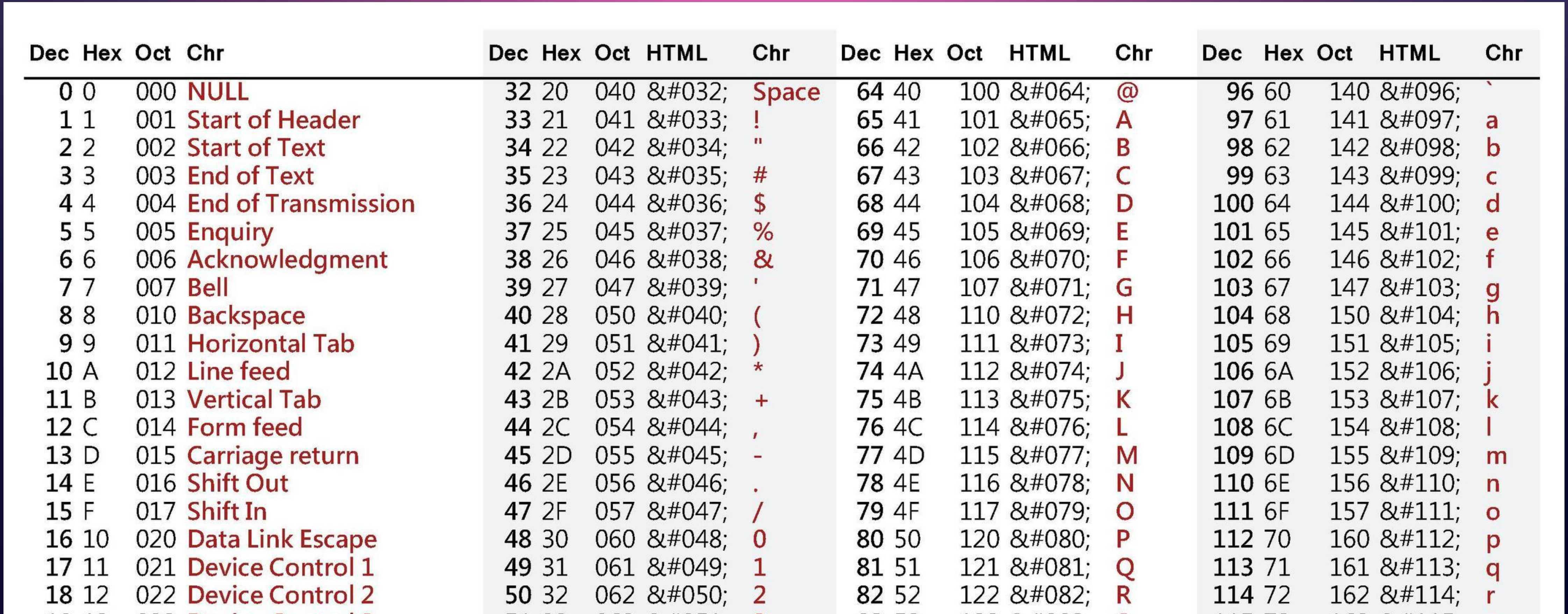
Floating point decimals

```
3.141592653589793238462643383279502884197169399375105820974944592307816406286
208998628034825342117067982148086513282306647093844609550582231725359408128481
117450284102701938521105559644622948954930381964428810975665933446128475648233
786783165271201909145648566923460348610454326648213393607260249141273724587006
606315588174881520920962829254091715364367892590360011330530548820466521384146
951941511609433057270365759591953092186117381932611793105118548074462379962749
567351885752724891227938183011949129833673362440656643086021394946395224737190
702179860943702770539217176293176752384674818467669405132000568127145263560827
785771342757789609173637178721468440901224953430146549585371050792279689258923
542019956112129021960864034418159813629774771309960518707211349999998372978049
951059731732816096318595024459455346908302642522308253344685035261931188171010
003137838752886587533208381420617177669147303598253490428755468731159562863882
353787593751957781857780532171226806613001927876611195909216420198938095257201
065485863278865936153381827968230301952035301852968995773622599413891249721775
283479131515574857242454150695950829533116861727855889075098381754637464939319
255060400927701671139009848824012858361603563707660104710181942955596198946767
837449448255379774726847104047534646208046684259069491293313677028989152104752
162056966024058038150193511253382430035587640247496473263914199272604269922796
```

Strings

“Groups of characters, letters, numbers…like this one.”

Strings



Dec	Hex	Oct	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr
0	0	000	NULL	32	20	040	&#032;	Space	64	40	100	&#064;	@	96	60	140	&#096;	`
1	1	001	Start of Header	33	21	041	&#033;	!	65	41	101	&#065;	A	97	61	141	&#097;	a
2	2	002	Start of Text	34	22	042	&#034;	"	66	42	102	&#066;	B	98	62	142	&#098;	b
3	3	003	End of Text	35	23	043	&#035;	#	67	43	103	&#067;	C	99	63	143	&#099;	c
4	4	004	End of Transmission	36	24	044	&#036;	$	68	44	104	&#068;	D	100	64	144	&#100;	d
5	5	005	Enquiry	37	25	045	&#037;	%	69	45	105	&#069;	E	101	65	145	&#101;	e
6	6	006	Acknowledgment	38	26	046	&#038;	&	70	46	106	&#070;	F	102	66	146	&#102;	f
7	7	007	Bell	39	27	047	&#039;	'	71	47	107	&#071;	G	103	67	147	&#103;	g
8	8	010	Backspace	40	28	050	&#040;	(	72	48	110	&#072;	H	104	68	150	&#104;	h
9	9	011	Horizontal Tab	41	29	051	&#041;	)	73	49	111	&#073;	I	105	69	151	&#105;	i
10	A	012	Line feed	42	2A	052	&#042;	*	74	4A	112	&#074;	J	106	6A	152	&#106;	j
11	B	013	Vertical Tab	43	2B	053	&#043;	+	75	4B	113	&#075;	K	107	6B	153	&#107;	k
12	C	014	Form feed	44	2C	054	&#044;	,	76	4C	114	&#076;	L	108	6C	154	&#108;	l
13	D	015	Carriage return	45	2D	055	&#045;	-	77	4D	115	&#077;	M	109	6D	155	&#109;	m
14	E	016	Shift Out	46	2E	056	&#046;	.	78	4E	116	&#078;	N	110	6E	156	&#110;	n
15	F	017	Shift In	47	2F	057	&#047;	/	79	4F	117	&#079;	O	111	6F	157	&#111;	o
16	10	020	Data Link Escape	48	30	060	&#048;	0	80	50	120	&#080;	P	112	70	160	&#112;	p
17	11	021	Device Control 1	49	31	061	&#049;	1	81	51	121	&#081;	Q	113	71	161	&#113;	q
18	12	022	Device Control 2	50	32	062	&#050;	2	82	52	122	&#082;	R	114	72	162	&#114;	r

Traditional first program

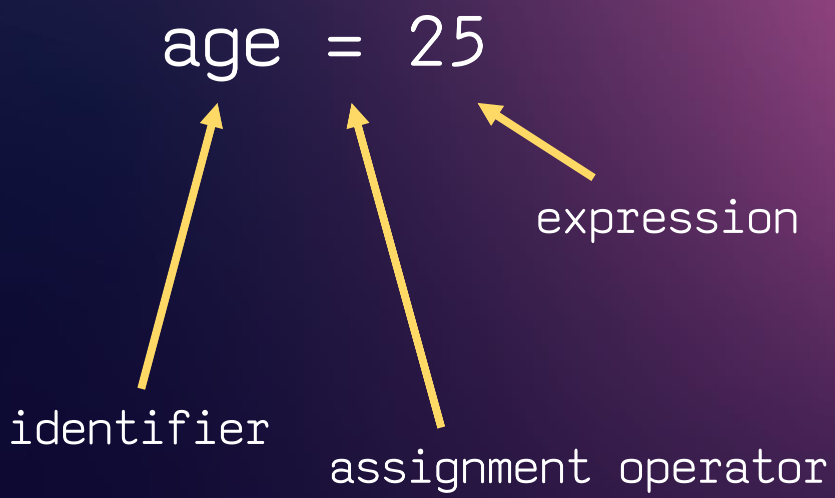
```
print(“Hello, World!”)
```

function

argument (**string**)

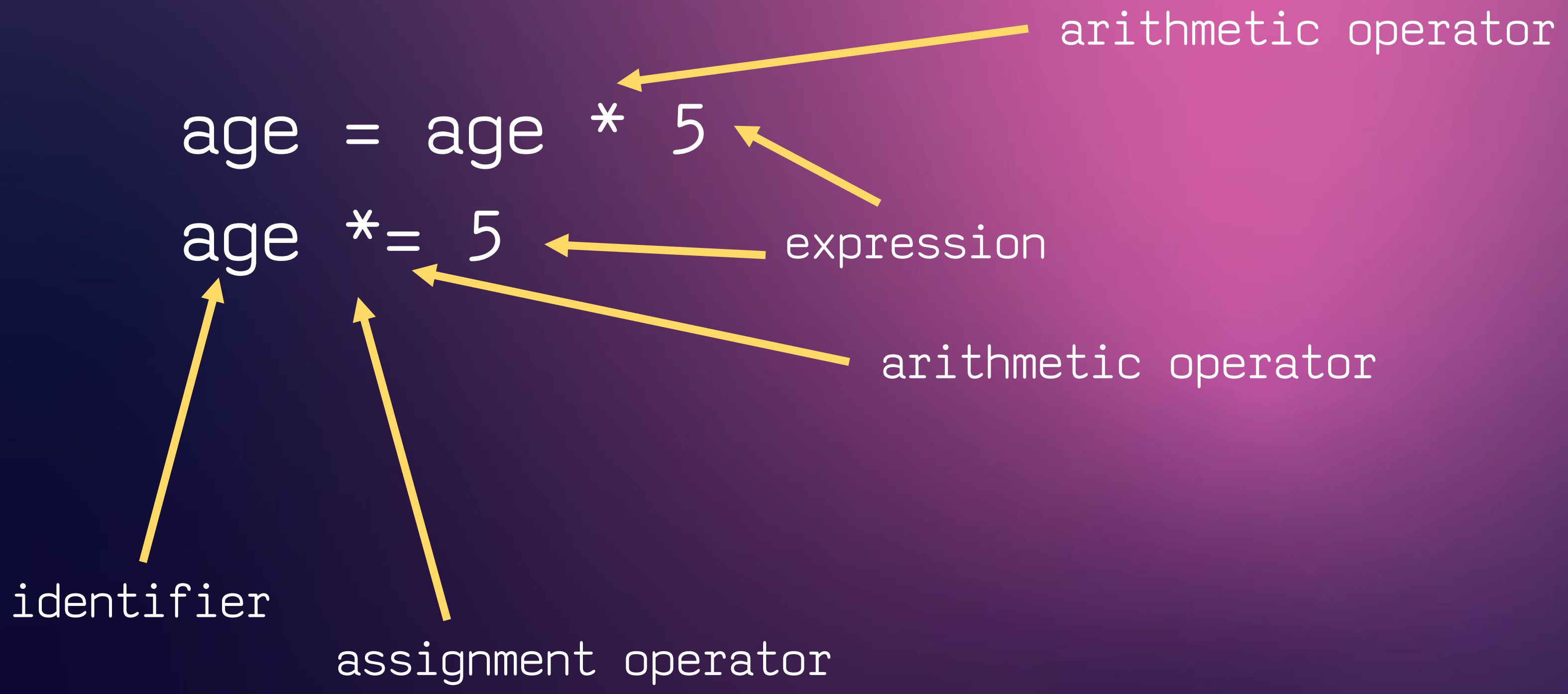
For now, something
bit more difficult:
a trick.

Create a variable called age



```
age = 25
```

Multiply age by 5 (choose one; they're equivalent!)



```
age = age * 5
age *= 5
```

Multiply it by 10.

```
age *= 10
```

Add today's date to age.

```
age += 19
```

Double it.

```
age *= 2
```

Add your shoe size;
round up if a half
size

```
age += #your shoe size
```

Let’s add some randomness.

I’ll think of a number
to subtract.

Type:

```
print(age)
```

The result is your
age and your shoe size in a
four-digit code.

REVISITING A PLATFORM: GITHUB

- Typically, we'd stop right there.
  - Our work is done, it's saved.
- It's on the JupyterHub, but it's not on our course GitHub.
  - It doesn't count as submitted until it's on the GitHub.
- The next step will transmit it there.

GITHUB WORKFLOW

```
git add .
```

```
git commit -m “{COMMIT MESSAGE}”
```

```
git commit -m “Saving progress”
```

Think of each of these like “snapshots.”

GITHUB WORKFLOW

```
git push
```

The last step - the submit step.