Python Training Sessions: Notes 0 (The Basics)

Welcome!

- This class is about more than computer programming!
- Indeed, this class is about problem-solving in a way that is exceedingly empowering! You will likely take the problem solving that you learn here will likely be instantly applicable to your work beyond this course and even your career as a whole!
- However, it will not be easy! You will be “drinking from the firehose” of knowledge during this course. You’ll be amazed at what you will be able to accomplish in the coming weeks.
- This course is far more about you advancing “you” from “where you are today” than hitting some imagined standard.
- The most important opening consideration in this course: Give the time you need to learn through this course. Everyone learns differently. If something does not work out well at the start, know that with time you will grow and grow in your skill.

What’s Ahead

- The course is aimed at learning Python. Your code will look something like this:

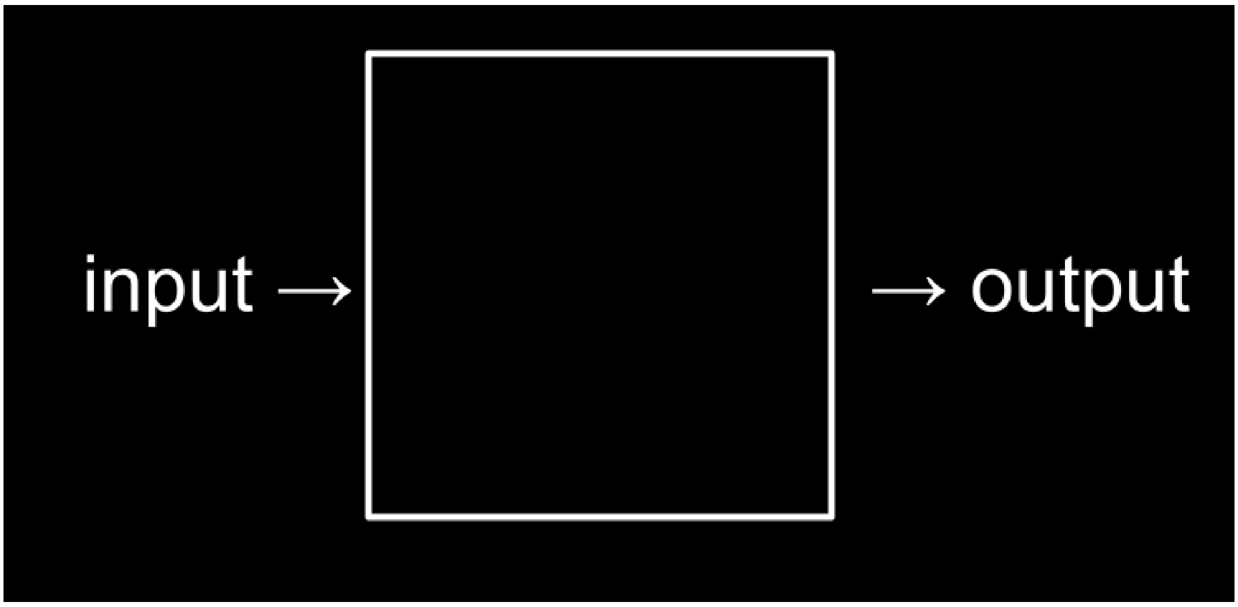
  ```
  print("hello, world")
  ```

- You will be learning functions and variables.
- Then, in future weeks, you will learn how to make your own functions.
- Further, as the weeks progress, you will learn about conditionals and loops.
- You will learn about exceptions, i.e. errors in Python programs.
- You will learn about libraries and how to use them. You'll even learn to make your own Python libraries.

- You will learn how to write unit tests for your own code or others' codes, so that your program always runs as expected.
- You will also learn File I/O using Python, which includes playing with excel sheets.
- We will also be looking at Regular Expressions for data validation using Python, again.
- Then we will look at Objected Oriented Programming in Python, a better way to write programs and build applications.
- Then, finally, we shall end this course by working on individual or group projects. And by this time, you will be ready to build amazing things using Python and will start using Python in your daily programming tasks.

Computational Thinking

- Essentially, computer programming is about taking some input and creating some output - thus solving a problem. What happens in between the input and output, what we could call *a black box,* is the focus of this course.

- For example, we may need to take attendance for a class. We could use a system called *unary* to count, one finger at a time. Computers today count using a system called *binary*. It's from the term *binary digit* that we get a familiar term called *bit*. A *bit* is a zero or one.
- Computers only speak in terms of zeros and ones. Zeros represent *off*. Ones represent *on*. Computers are millions, and perhaps billions, of transistors that are being turned on and off.
- If you imagine using a light bulb, a single bulb can only count from zero to one.
- However, if you were to have three light bulbs, there are more options open to you!
- Using three light bulbs, the following could represent zero:

  ```
  0 0 0
  ```

- Similarly, the following would represent one:

```
0 0 1
```

- By this logic, we could propose that the following equals two:

```
0 1 0
```

- Extending this logic further, the following represents three:

```
0 1 1
```

- Four would appear as:

```
1 0 0
```

- We could, in fact, using only three light bulbs count as high as seven!

```
1 1 1
```

- As a heuristic, we could imagine that the following values represent each possible place in our *binary digit*:

```
4 2 1
```

- Computers use 'base-2' to count. This can be pictured as follows:

```
2^2  2^1  2^0
4    2    1
```

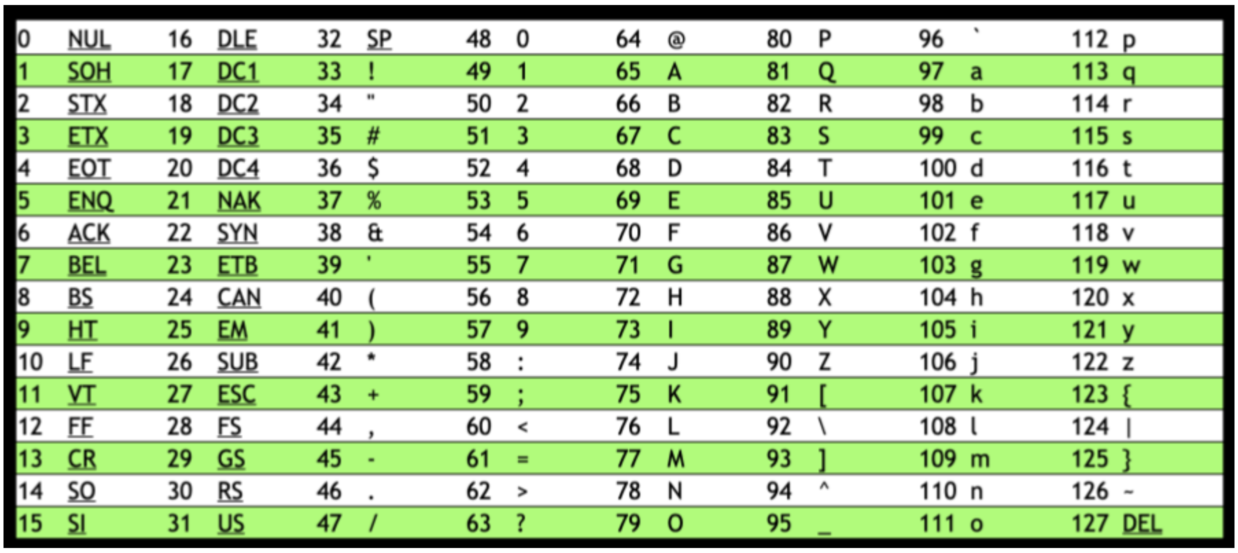
- Therefore, you could say that it would require three bits (the four's place, the two's place, and the one's place) to represent a number as high as seven.
- Computers generally use eight bits to represent a number. For example, `00000101` is the number 5 in *binary*.

Text

- Just as numbers are binary patterns of ones and zeros, letters are represented using ones and zeros too!
- Since there is an overlap between the ones and zeros that represent numbers and letters, the *ASCII* standard was created to map specific letters to specific numbers.
- For example, the letter `A` was decided to map to the number 65.
- If you received a text message, the binary under that message might represent the numbers 72, 73, and 33. Mapping these out to ASCII, your message would look as follows:

```
H   I   !
72  73  33
```

- Thank goodness for standards like ASCII that allow us to agree upon these values!
- Here is an expanded map of ASCII values:

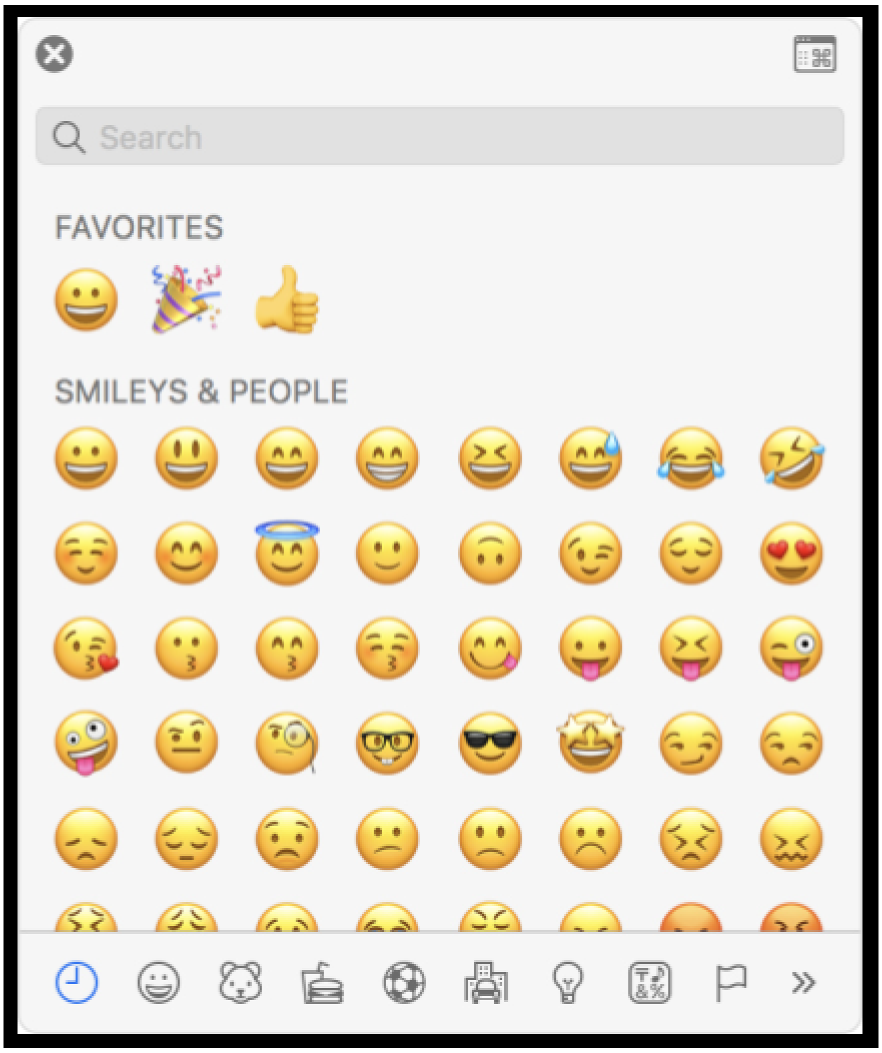


0	NUL	16	DLE	32	SP	48	0	64	@	80	P	96	`	112	p
1	SOH	17	DC1	33	!	49	1	65	A	81	Q	97	a	113	q
2	STX	18	DC2	34	"	50	2	66	B	82	R	98	b	114	r
3	ETX	19	DC3	35	#	51	3	67	C	83	S	99	c	115	s
4	EOT	20	DC4	36	$	52	4	68	D	84	T	100	d	116	t
5	ENQ	21	NAK	37	%	53	5	69	E	85	U	101	e	117	u
6	ACK	22	SYN	38	&	54	6	70	F	86	V	102	f	118	v
7	BEL	23	ETB	39	'	55	7	71	G	87	W	103	g	119	w
8	BS	24	CAN	40	(	56	8	72	H	88	X	104	h	120	x
9	HT	25	EM	41	)	57	9	73	I	89	Y	105	i	121	y
10	LF	26	SUB	42	*	58	:	74	J	90	Z	106	j	122	z
11	VT	27	ESC	43	+	59	;	75	K	91	[	107	k	123	{
12	FF	28	FS	44	,	60	<	76	L	92	\	108	l	124	|
13	CR	29	GS	45	-	61	=	77	M	93	]	109	m	125	}
14	SO	30	RS	46	.	62	>	78	N	94	^	110	n	126	~
15	SI	31	US	47	/	63	?	79	O	95	_	111	o	127	DEL

- If you wish, you can learn more about [ASCII (https://en.wikipedia.org/wiki/ASCII)](https://en.wikipedia.org/wiki/ASCII).

Emojis

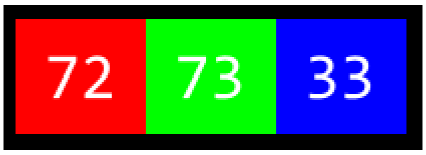
- As time has rolled on, there are more and more ways to communicate via text.
- Since there were not enough digits in binary to represent all the various characters that could be represented by humans, the *Unicode* standard expanded the number of bits that can be transmitted and understood by computers.
- There are emojis that you probably use every day. The following may look familiar to you:



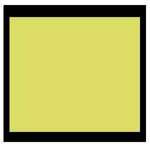
- Computer scientists faced a challenge when wanting to assign various skin tones to each emoji to allow the communication to be further personalized. In this case, the creators and contributors of emojis decided that the initial bits would be the structure of the emoji itself, followed by skin tone.
- More and more features are being added to the Unicode standard to represent further characters and emojis.
- If you wish, you can learn more about **Unicode (https://en.wikipedia.org/wiki/Unicode)**.
- If you wish, you can learn more about **emojis (https://en.wikipedia.org/wiki/Emoji)**.

RGB

- Red, green, and blue (called `RGB`) is a combination of three numbers.



- Taking our previously used 72, 73, and 33, which said `HI!` via text, would be interpreted by image readers as a light shade of yellow. The red value would be 72, the green value would be 73, and the blue would be 33.



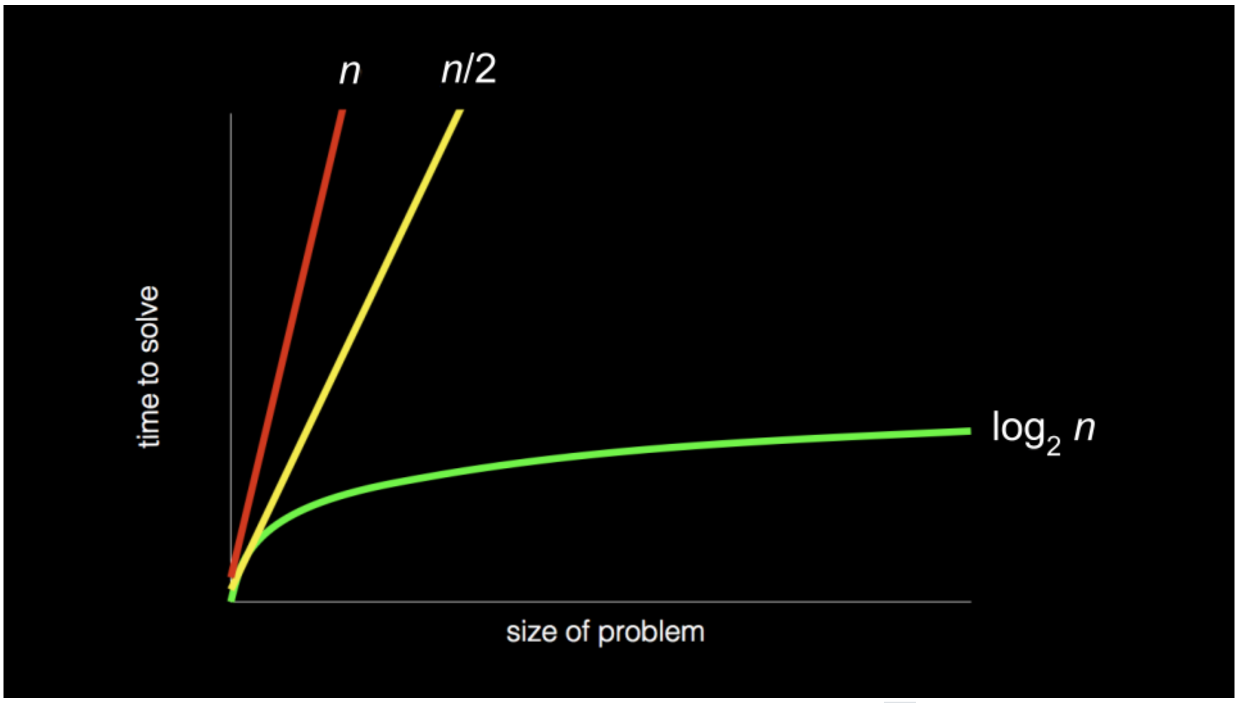
Images, Video and Sound

- Images are simply collections of RGB values.
- Videos are sequences of many images that are stored together, just like a flipbook.
- Music can be represented through MIDI data.

Algorithms

- Problem-solving is central to computer science and computer programming.
- Imagine the basic problem of trying to locate a single name in a phone book.
- How might you go about this?
- One approach could be to simply read from page one to the next to the next until reaching the last page.
- Another approach could be to search two pages at a time.
- A final and perhaps better approach could be to go to the middle of the phone book and ask, "Is the name I am looking for to the left or to the right?" Then, repeat this process, cutting the problem in half and half and half.

- Each of these approaches could be called algorithms. The speed of each of these algorithms can be pictured as follows in what is called *big-O notation*:



Notice that the first algorithm, highlighted in red, has a big-O of `n` because if there are 100 names in the phone book, it could take up to 100 tries to find the correct name. The second algorithm, where two pages were searched at a time, has a big-O of 'n/2' because we searched twice as fast through the pages. The final algorithm has a big-O of $\log_2 n$ as doubling the problem would only result in one more step to solve the problem.

Pseudocode and the Basic Building Blocks of Programming

- The ability to create *pseudocode* is central to one's success in both this class and in computer programming.
- Pseudocode is a human-readable version of your code. For example, considering the third algorithm above, we could compose pseudocode as follows:

```
1  Pick up phone book
2  Open to middle of phone book
3  Look at page
4  If person is on page
5      Call person
6  Else if person is earlier in book
7      Open to middle of left half of book
8      Go back to line 3
9  Else if person is later in book
10     Open to middle of right half of book
11     Go back to line 3
12 Else
13     Quit
```

- Pseudocoding is such an important skill for at least two reasons. First, when you pseudocode before you create formal code, it allows you to think through the logic of your problem in advance. Second, when you pseudocode, you can later provide this information to others that are seeking to understand your coding decisions and how your code works.

- Notice that the language within our pseudocode has some unique features. First, some of these lines begin with verbs like *pick up, open, look at.* Later, we will call these *functions*.
- Second, notice that some lines include statements like `if` or `else if.` These are called *conditionals*.
- Third, notice how there are expressions that can be stated as *true* or *false,* such as "person is earlier in the book." We call these *boolean expressions*.
- Finally, notice how these statements like "go back to line 3." We call these *loops*.

Summing Up

In this lesson, you learned how this course sits in the wide world of computer science and programming. You learned…

- Few students come to this class with prior coding experience!
- You are not alone! You are part of a community.
- Problem solving is the essence of the work of computer scientists.
- This course is not simply about programming – this course will introduce you to a new way of learning that you can apply to almost every area of life.
- How numbers, text, images, music, and video are understood by computers.
- The fundamental programming skill of pseudocoding.
- How abstraction will play a role in your future work in this course.
- The basic building blocks of programming, including functions, conditionals, loops, and variables.

See you next time!