Complete Python Course Bootcamp from Beginners to Professional


By Kaustubh Wankhede

kwinfosys

Introduction to Computer Science

Basic Concepts and Fundamentals of Python Programming

kwinfosys

Basic Concepts and Fundamentals of Python Programming

1. Introduction to Computer Science
2. Introduction to Python Programming
3. print() function
4. Data Types
5. Variables
6. Calculations
7. String Formatting
8. Escape Sequences
9. String Indexing
10. String Slicing
11. Step Slicing
12. Input Method
13. Project 1 - Band Name Generator
14. Project 2 - Tip Calculator

Presentation Agenda

What is Computer Science?

Representation

Binary Language

ASCII

What is Computer Science ?

Introduction to Computer Science

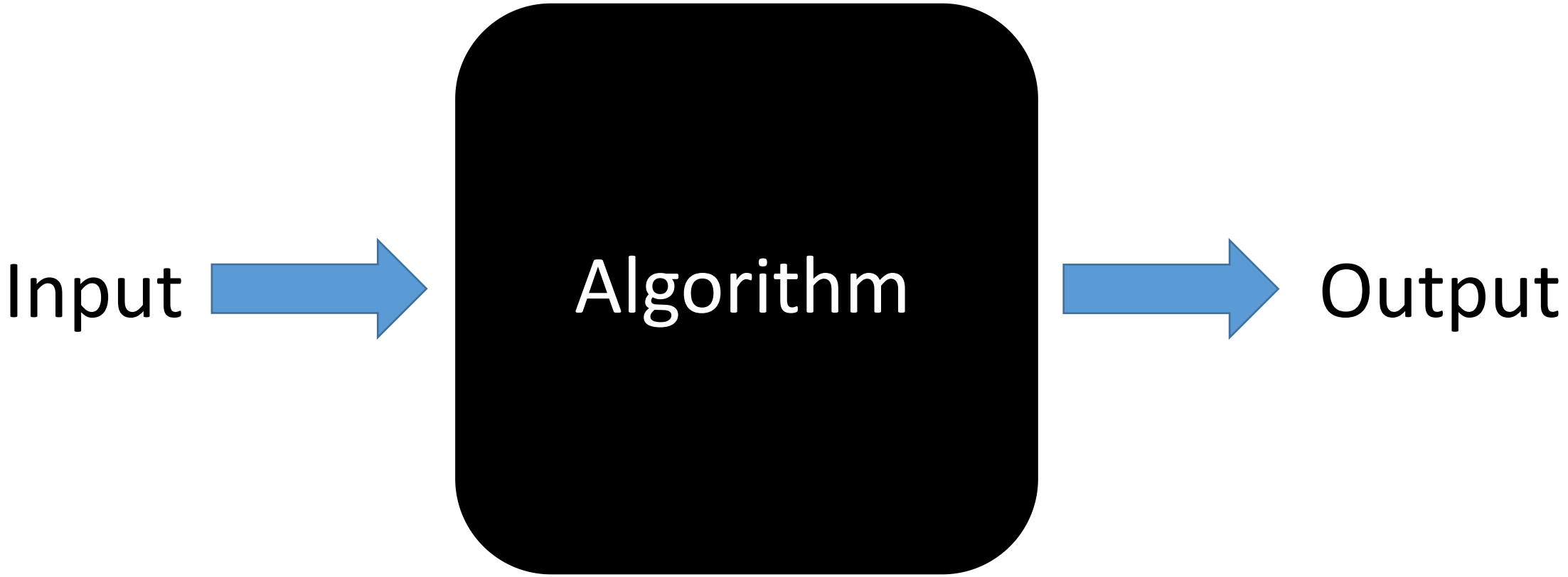
kwinfosys

What is Computer Science?

Computer Science is fundamentally problem solving.

We can think of problem solving as the process of taking some input (details about our problem) and generate some output (the solution to our problem). The “black box” in the middle is computer science, or the code we’ll learn to write.

Computer Science



Representation

Introduction to Computer Science

kwinfosys

Representation

To begin with Computer Science, we'll need a way to represent inputs and outputs, so we can store and work with information in a standardized way.

We might start with the task of taking attendance by counting the number of people in a room. With our hand, we might raise one finger at a time to represent each person, but we won't be able to count very high. This system is called **unary**, where each digit represents a single value of one.

Representation

We’ve probably learned a more efficient system to represent numbers, where we have ten digits.

Whats that system ?

Representation

This system is called **decimal**, or **base 10**, since there are ten different values that a digit can represent.

0 1 2 3 4 5 6 7 8 9

Representation

Computers use a simpler system called binary, or base two, with only 2 possible digits, 0 and 1.

Each binary digit is also called as bit.

Since computers run on electricity, which can be turned on or off, we can conveniently represent a bit by turning some switch on or off to represent a 0 or 1. With one light bulb, for example, we can turn it on to count to 1.

kwinfosys

0

1





Representation

Inside modern computers, there are not light bulbs but million of tiny switches called transistors that can be turned on and off to represent different values.

Binary Language

Introduction to Computer Science

kwinfosys

100 10 1

123

100 x 1 + 10 x 2 + 1 x 3

123

100 10 1

123

10^2 10^1 10^0

123

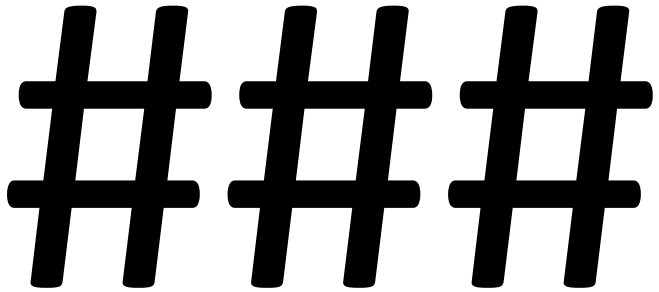
Each place for a digit represents a power of ten, since there are ten possible digits for each place. The rightmost place is for 10^0, the middle one 10^1, and the leftmost place 10^2

2^2 2^1 2^0

###

In binary language, with just two digits, we have powers of two for each place value.

4 2 1



4 2 1

000

With all the light bulbs or switches off, we would still have a value of 0.

4 2 1

001

4 2 1

010

4 2 1

011

4 2 1

100

4 2 1

101

4 2 1

110

4 2 1

1 1 1

32 16 8 4 2 1

110010

If we had several more light bulbs, we might have a binary value of 110010, which would have the equivalent decimal value of 50.

Notice that 32 + 16 + 2 = 50

ASCII - American Standard Code for Information Interchange

Introduction to Computer Science

kwinfosys

A

65

B

66

C

67

D

68

ASCII

To represent letters, all we need to do is decide how numbers map to letters. Some humans, many years ago, collectively decided on a standard mapping of numbers to letters. The letter “A”, for example, is the number 65, and “B” is 66, and so on. By using context, like whether we’re looking at a spreadsheet or an email, different programs can interpret and display the same bits as numbers or text.

The standard mapping, ASCII, also includes lowercase letters and punctuation.

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

ASCII Table

ASCII

If we received a text message with a pattern of bits that had the decimal values 72, 73, and 33, those bits would map to the letters ‘HI!’.

Each letter is typically represented with a pattern of eight bits, or a byte, so the sequences of bits we would receive are 01001000, 01001001, and 00100001.

With eight bits, or one byte, we can have 2^8, or 256 different values (including zero). (The highest value we can count up to would be 255.)

Thank You!

Kaustubh Wankhede

kwinfosys