CS105 Problem Solving

Code Reading and Problem Decomposition

Wholeness

- Our focus today is to fine-tune our skills. The exam showed that there were some issues with code reading and taking problems appart, solving the pieces and putting them back together.
- This is similar to the principle that purification leads to progress.

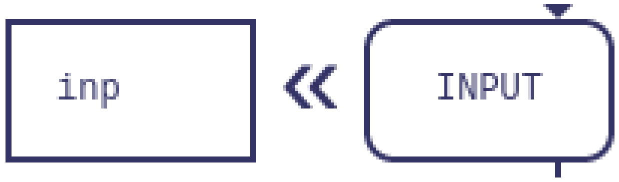
Reading Code

On the exam most of the mistakes were made on code reading.

- Often people were not sure what creates output
- Other times people had trouble with data types / operators

We will review and practice these aspects

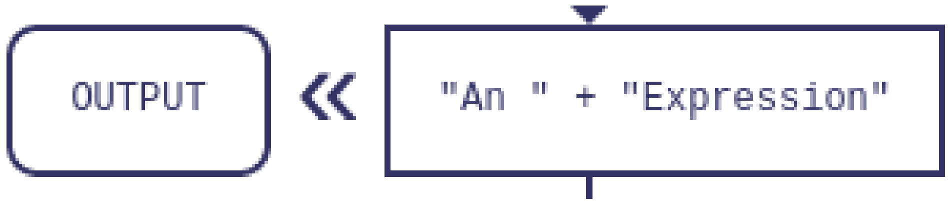
Input



```
var inp; // string
inp = prompt('Enter Input: ');
```

Takes input from the user and stores it into a string variable
You'll never see these in "what is the output" exercises
Because user input could be anything!

Output



```
console.log("An " + "Expression");
```

Outputs the (single value) result of evaluating the expression.

Exercise

- What is the output:

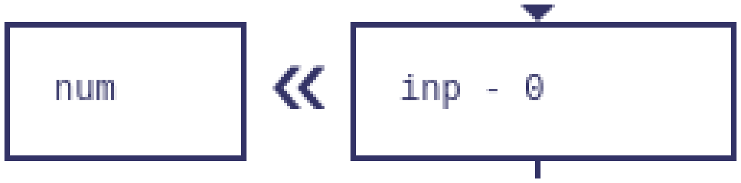
```
console.log(“Hello! ”);
console.log(“This is a test \n”);
console.log(10 + “5”);
```

Operators

- A + with a string on one side and a number on another
  - Turns the number into a string and concatenates

- Any other operator with 1 string and 1 number
  - Will turn the string into a number

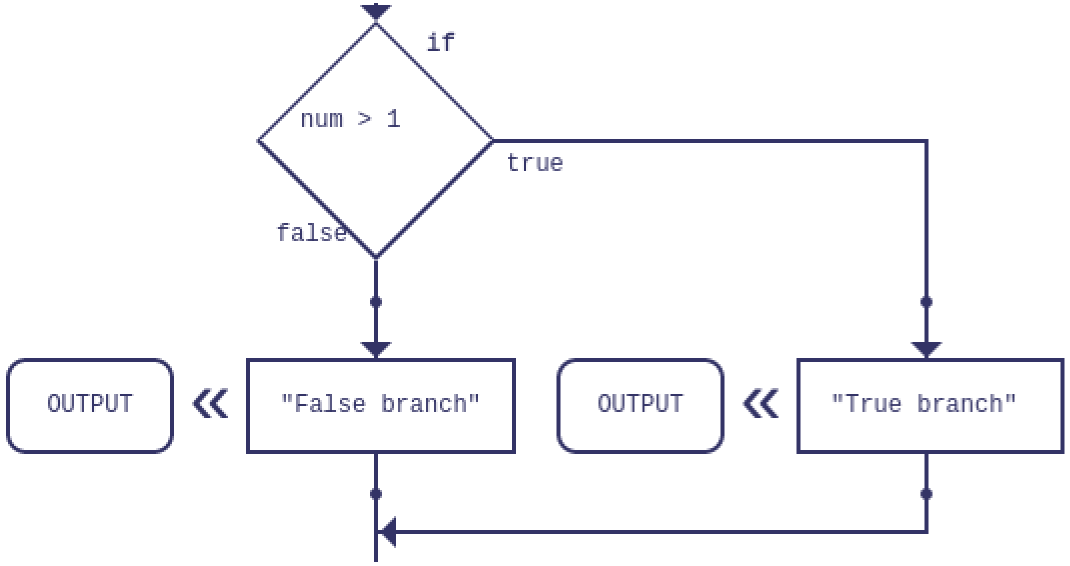
Assignment



```
var num; // number
num = inp - 0;
```

Assigns the result (value) of an expression into a variable
Important! Make sure the data type of the value and the variable match!

If Statement



```
if (num > 1) {
    console.log("True branch");
} else {
    console.log("False branch");
}
```

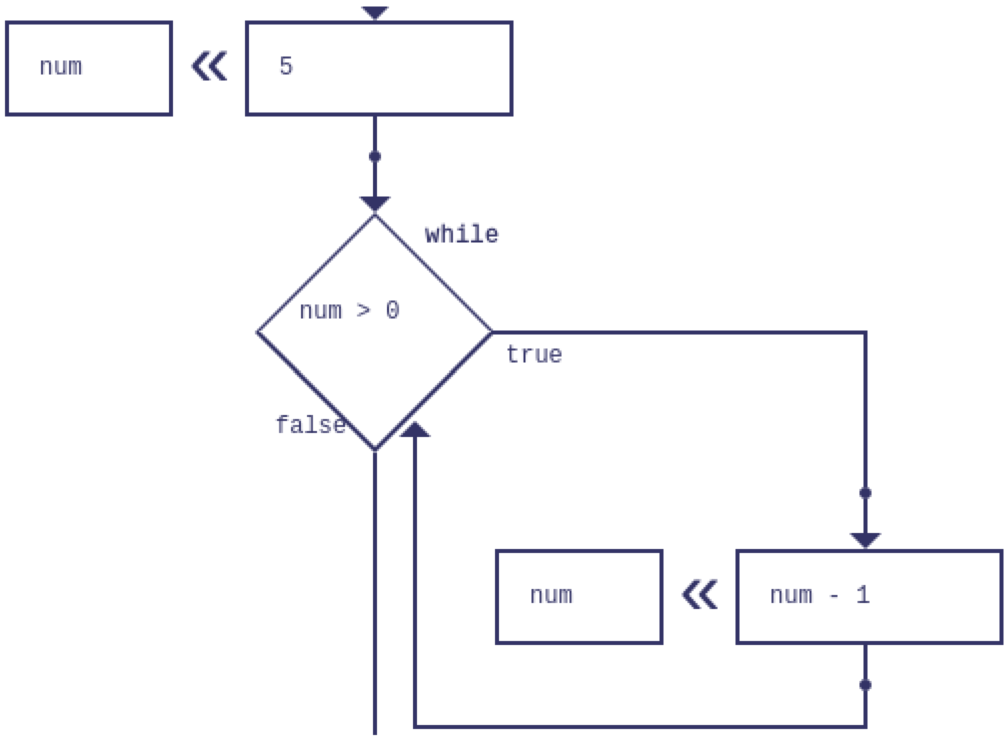
Evaluates the (boolean) condition expression
If true executes the true branch
Otherwise (false) executes the false branch

Exercise

- What is the output

```
x = 50;
y = x - 36;
z = y * 2;
x = x - z
if (x > y) {
    y = z;
} else {
    z = x;
}
console.log(x + " " + y +  " " + z)
```

Loop



```
num = 5;
while (num > 0) {
    num = num - 1;
}
```

Generally with a counter
Counter initialized before the loop
Loop condition specifies end point
Assignment inside loop moves towards end

Evaluates the (boolean) condition expression
If true executes the loop and evaluates the condition again

Exercise

- What is the output?

```
var i = 0;
while (i < 20) {
    if (i %3 == 0) {
        if (i < 12) {
            console.log("bim");
        } else {
            console.log("bam");
        }
    }
    i = i + 1;
}
console.log("bom");
```

Main Point

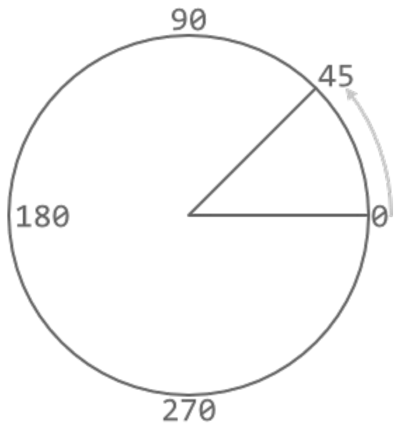
Each flowchart element has a 1 to 1 correspondence to code.

Being able to correctly interpret / read code is important as you’ll often be working with code that people wrote.

Turtle Graphics

Remember that .rotate() modifies the current degree

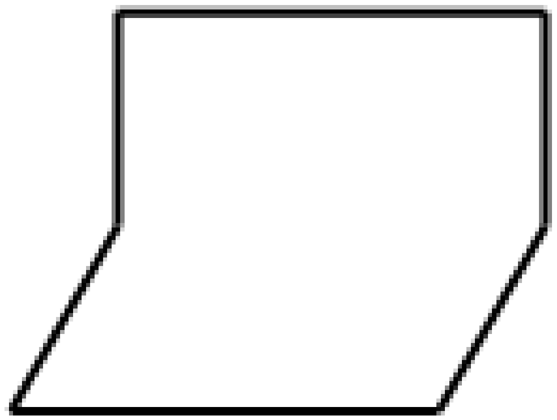
- This means you it is the angle between where you would have gone without rotating and where you are going with rotating
- It is (usually) not an angle you between lines on the screen



This picture shows rotating left 45 degrees

Example

Let me draw the rotation angles on the the following



Main Point

- The angle you specify for .rotate() on the turtle is how much its direction should change.

Breaking a Problem into Parts

- When problems we’ve been doing don’t get solved, or solved incorrectly it’s often because the parts have gotten mixed

- Each part can be solved individually, and then combined relatively easily (little or no interaction with the other parts) to create a total solution.

Example

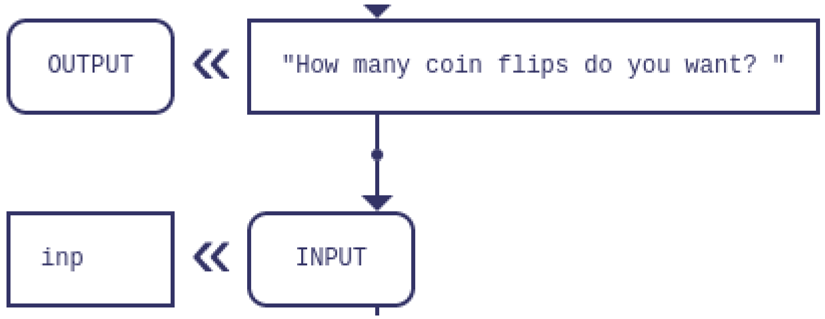
Write a program that asks how many coin flips you want, and then uses a loop to 'flip' (create heads / tails) that amount of times.

There are 3 main parts here:

- Input
- A loop with a counter
- 'a coin flip'

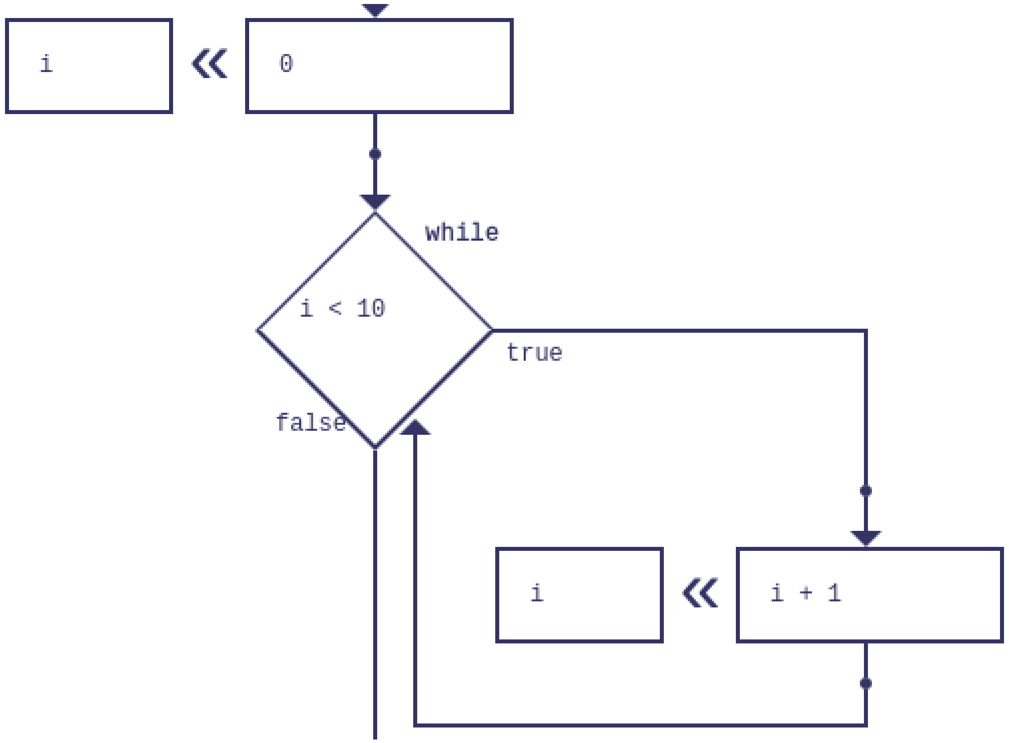
Solving the Parts

Getting User Input



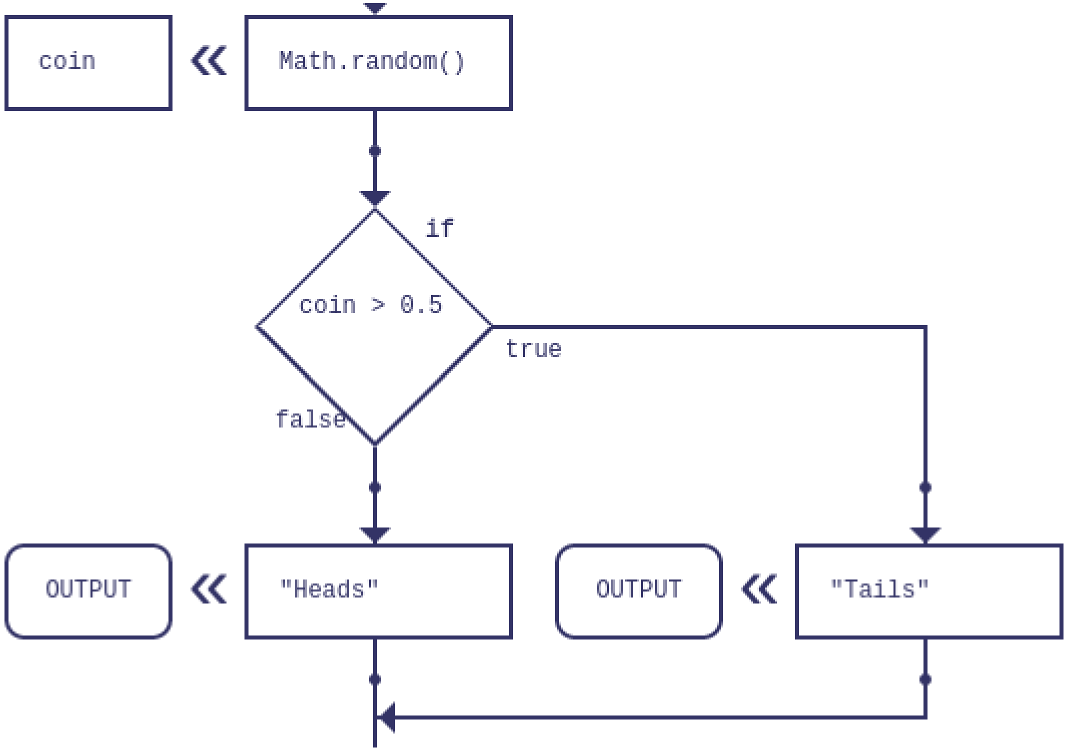
```
console.log("How many coin flips do you want? ");
inp = prompt('Enter Input: ');
```

A Loop with a Counter



```
i = 0;
while (i < 10) {
    i = i + 1;
}
```

A Coin Flip



```
coin = Math.random();
if (coin > 0.5) {
    console.log("Tails");
} else {
    console.log("Heads");
}
```

Putting the Parts Together

```
var inp; // string
var i; // number
var coin; // number
console.log("How many coin flips do you want? ");
inp = prompt('Enter Input: ');
i = 0;
while (i < inp) {
    coin = Math.random();
    if (coin > 0.5) {
        console.log("Tails");
    } else {
        console.log("Heads");
    }
    i = i + 1;
}
```

The parts don’t interact much

- inp is used in the loop condition
- ‘Coin flip’ is inside the loop body

Exercise

Identify the parts of the following problem:
(here the parts are not as explicit)

Write a program that asks the user for input, and then counts how many times the letter e is in the input string.

The Parts

- Getting input from the user
- Looking through string / counting (which means a loop)
- Is a letter an E?

Can you solve the parts?

Joining the Parts

```
var inp; // string
var indexInString; // number
var numberOfEs; // number
console.log("Please enter some input ");
inp = prompt('Enter Input: ');
numberOfEs = 0;
indexInString = 0;
while (indexInString < inp.length) {

    if (inp[indexInString] == "e") {
        numberOfEs = numberOfEs + 1;
    } else {
    }

    indexInString = indexInString + 1;
}
console.log("There were " + numberOfEs + " e's");
```

Interaction of the parts:

- numberOfEs is initialized before loop
- Is an ‘e’ is inside the loop
- inp is used in the loop condition

Main Point

The key to solving a big / complex problem is identifying the smaller / simpler parts that it is made of, solving them and joining them together.

Neither the parts or the joining should be complex.

Complexity indicates trying multiple things in one. Identifying them, solving them separately, and then joining them will be the solution.

More about Strings

We’ll look at:

- Character codes of letters
- Making a copy of a string

Character Codes

Each character is actually a number (that the computer stores)

- Think of it like A is 1, B is 2 (but more elaborate)

When you have a character – you can get the number

When you have a number – you can get the characters

- Numbers are easy to change

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
19	13	023	Device Control 3	51	33	063	&#051;	3	83	53	123	&#083;	S	115	73	163	&#115;	s
20	14	024	Device Control 4	52	34	064	&#052;	4	84	54	124	&#084;	T	116	74	164	&#116;	t
21	15	025	Negative Ack.	53	35	065	&#053;	5	85	55	125	&#085;	U	117	75	165	&#117;	u
22	16	026	Synchronous idle	54	36	066	&#054;	6	86	56	126	&#086;	V	118	76	166	&#118;	v
23	17	027	End of Trans. Block	55	37	067	&#055;	7	87	57	127	&#087;	W	119	77	167	&#119;	w
24	18	030	Cancel	56	38	070	&#056;	8	88	58	130	&#088;	X	120	78	170	&#120;	x
25	19	031	End of Medium	57	39	071	&#057;	9	89	59	131	&#089;	Y	121	79	171	&#121;	y
26	1A	032	Substitute	58	3A	072	&#058;	:	90	5A	132	&#090;	Z	122	7A	172	&#122;	z
27	1B	033	Escape	59	3B	073	&#059;	;	91	5B	133	&#091;	[	123	7B	173	&#123;	{
28	1C	034	File Separator	60	3C	074	&#060;	<	92	5C	134	&#092;	\	124	7C	174	&#124;	|
29	1D	035	Group Separator	61	3D	075	&#061;	=	93	5D	135	&#093;	]	125	7D	175	&#125;	}
30	1E	036	Record Separator	62	3E	076	&#062;	>	94	5E	136	&#094;	^	126	7E	176	&#126;	~
31	1F	037	Unit Separator	63	3F	077	&#063;	?	95	5F	137	&#095;	_	127	7F	177	&#127;	Del

asciichars.com

Methods to Convert

“a”.charCode(0) == 97

String.fromCharCode(97) == “a”

A reason to convert:

- To change lower case to upper case is simply subtracting 32

Copying a String

- Although you cannot change characters in a string
  - You can make a copy

- You can make a slightly different copy!
  - Looks like you manipulated the original :)

Copying a String

```
var inp; // string
var copy; // string
var index; // number
console.log("Please enter a string: ");
inp = prompt('Enter Input: ');
copy = "";
index = 0;
while (index < inp.length) {
    copy = copy + inp[index];
    index = index + 1;
}
console.log("A copy of what you wrote: " + copy);
```

Exercise

Make a copy of a string

Once you’ve got it working make it so that the copy

- Changes all “e” characters to “a” characters

Main Point

You can do “String Manipulation” by making a copy that has the changes you want. An important aspect to know about when manipulating text is that each character is also a number.

Summary

- Each flowchart element has a 1:1 correspondence to code
- The angle you specify is the angle that the turtle rotates
- Break problems down into easy parts and then join them
- Copying strings lets you ‘manipulate’ them