

# Programming for Kids and Parent

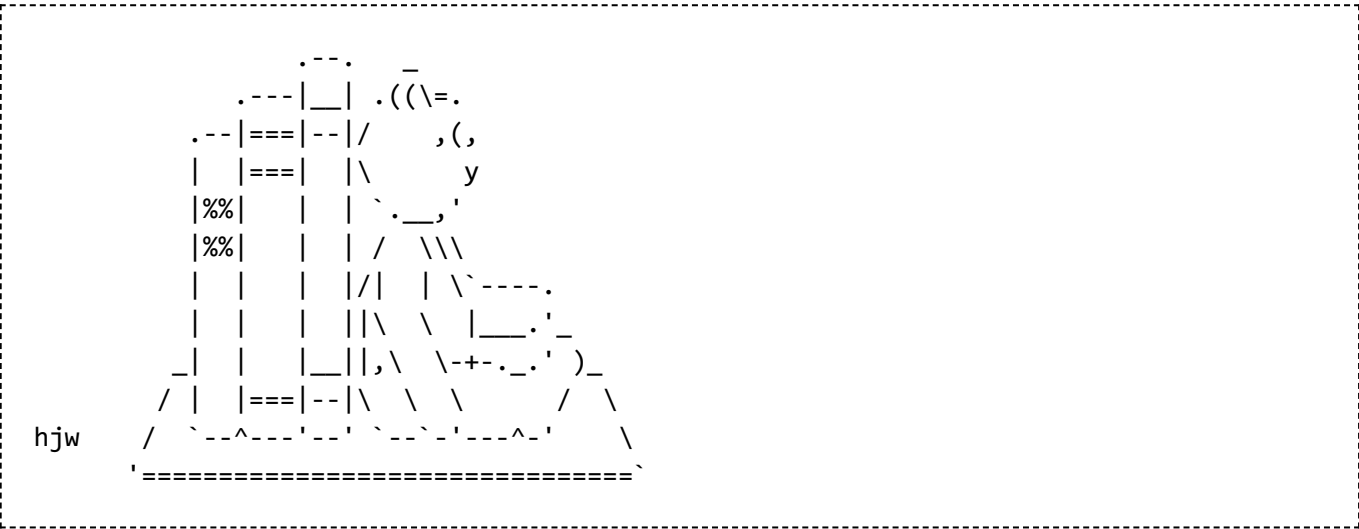
by: Borislav Nikolov

year: 2021

The parent has to know how to program.

Spend 30 minutes per day.

Every day.



"Anything worth doing is worth doing badly." — G. K. Chesterton



This work is licensed under a [Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License](#).

# Hello World

---

*This chapter is for parents, kids skip to the next one.*

This book is for parents who know how to code and for kids who don't, but especially for parents and kids who can spend 30 minutes per day, *every day*. I am writing this book as I am teaching my daughter (10), and you know how in some cooking shows, they skip the part where the food is cooking? I won't do that. The book will be longer than it should.

Again, if your children are older, or younger, this book might not work, you could of course still find pieces that work for you.

This is more of a log of my experience so far.

What you need:

- Computer
- Patience
- Internet
- Patience

If you don't have a you can buy raspberry pi 400 for 70\$ or so, or something similar that you attach to your TV. If you don't have patience, buy some chamomile tea. You don't need internet subscription, but you would need a bit of internet to download python and do few google searches.

NB: In the book I refer to 'notepad' or 'double click' assuming most people run windows, if you don't, you will have to translate what the instructions mean.

The schedule is roughly as follows:

- Week 0
  - Touch Typing
  - HTML
- Week 1
  - HTML: h1, marquee, pink text
  - Touch Typing
- Week 2
  - HTML: tables
  - HTML: lists
  - Touch Typing
- Week 3
  - HTML: tables lists
  - HTML: images, licenses
  - JavaScript: few small programs
- Week 4
  - python: super basic python (print and input and while True)
- Week 5
  - python: more basic python
- Week 6

- python: make hangman game
- Week 7
  - python: turtle
- Week 8
  - HTML: love match
  - python: text trivia game
- Week 9
  - python: super basic python
- Week 10
  - python: turtle
- Week 11
  - python: make text tic tac toe game
- Week 12
  - python: pygame
  - python: turtle
- Week 13
  - python: pygame
  - python: turtle
- Week 14
  - python: basics
  - python: turtle
- Week 15
  - HTML: make a table
  - python: make tic tac toe again
  - python: basics
  - python: pygame
- Week 16
  - python: love match
  - python: trivia game
  - python: turtle
- Week 17
  - python: dungeon game
  - python: turtle grid
  - python: turtle keyboard input
- Week 18
  - python: turtle tic tac toe

In most of the weeks you also go back, waaay back, every day you re-iterate variables and for loops, print the numbers from one to 10 forever, ask how many times to be printed, etc.

The reason for so much emphasis on HTML is because it helps with understanding hierarchy, `tr` is child of `table`, `table` is child of `body` body is child of `html`, `td` is a sister to `td` and both are children of `tr` etc, it helped a lot with my daughter understanding how the `else` is sibling to the `if` and how both are children to `while`.

Also it is very easy to debug, and inspect, and get immediate feedback. There are many 'hackers' now on tictok or youtube that shows you how to get a lot robux(money) in roblox by inspecting the page and

modifying the HTML, so I think HTML is very important to be understood, not only because it teaches hierarchy, but also it is the canvas of the web.

## Other materials

Play other games as well, <https://tomorrowcorporation.com/> has some brilliant games, Human Resource Machine is great way to learn loops and conditional jumps, and 7 Billion Humans is amazing for high level concepts, including recursion and sorting.

The Robot Turtles game is amazing as well, you can find it here: <https://www.thinkfun.com/products/robot-turtles/>

Scratch works for some kids, mine didn't enjoy it much.

Buy few Arduino NANOs (cheap clones from amazon work as well, but you need to install ch340 driver), and some servo motors and write few super basic programs that turn the servo slowly in one direction or another. Connect **black/brown wire to gnd, red wire to 5v and orange wire to D9**, and run:

```
#include <Servo.h>
#define SERVO_PIN 9
Servo s;
void setup() {
  s.attach(SERVO_PIN, 1000, 2000);
}

void loop() {
  s.write(0);
  delay(500);

  s.write(60);
  delay(500);

  s.write(120);
  delay(500);
}
```

## Motivation

Sometimes there is very little motivation. Kids are super tired from school and playdates and extracurricular activities, makes it hard to spend 30 minutes per day on something. Netflix and YouTube and etc are so much more interesting than what I have to offer, it is a rigged game.. Sad that I have to compete with billion parameter models, but here we are.

Anything worth doing, is worth doing poorly. If you cant spend 30 minutes, spend 15, spend 5 minutes if you have to, or even 1. It is all worth it.

Try to spend time in the morning, at least on the weekends, and first thing after school on school days.

We tried to setup time before or after dinner, and it is very very difficult.

# Chapter 0 - Week 0

---

```
day0: learn about the computer
day1: touch typing on keybr.com
day2: install python and make a program
day3: touch typing using your program
day4: HTML
day5: touch typing
day6: HTML
```

This week is one of the more difficult ones. Lots of new things, and some of them might not work on your computer because it is too new or too old, for which I apologize. You might need a bit of help. If you get stuck just call a friend or parent to help you out.

## [DAY-0] The Computer

All modern computers(laptops, phones, pc master race rgb monsters, etc) have somewhat similar components: Processor, Memory, Video Card, Disk and USB controller, WiFi card etc. Some of them are in one single chip and you cant even see them anymore, but they are there. For example there are chips that have Processor and Video Card together. The term for processor is actually CPU - Central processing unit, but we called it processors when we were kids and it kind of make sense, since it processes stuff.

```
+-----+
|       |
| [      ] |
| [ Processor ] |
| [      ] |
|
| [ Memory ] |
| [ Memory ] |
|
| [ Video Card ]---+-----> monitor
|
| [ Disk ]         |
|                  | +---> iphone
|                  | +---> mouse
| [ USB ]-----+---+---> keyboard
|
| [ WiFi ]
| [ ... ]
+-----+
```

Memory (also called RAM), is the place where programs exist to be run by the CPU, it loads its instructions from there, and the instructions tells it what to do, it can write back to memory, or read from a different place or write something to the disk controller or to the video card, etc. Basically the most important stuff happens between the processor and the RAM (which stands for Random Access Memory). It is Random Access because the processor can just go and read or write to specific place, which is pretty cool.

Everything in the RAM disappears when you turn off the computer, so usually the programs and all kinds of information is persistently stored on the Disk. Which is called disk because it used to be spinning, but now most computers have SSD disks which are not *disks* at all, google 'SSD versus spinning disk' if you want to find more. So when the computer starts it will load some information from the disk into memory and start the operating system, which is just a program, not very different than the one you wrote for the touch typing, it takes input, does something with it and then has some output.

## [DAY-0] Files

When you store something to disk you usually store it as a file, files can be all kinds, only text, or images, or just raw binary data used for different purposes. Text files are literally that, file that contains text, later you will learn that there is no such thing as text to the computer, all is just bits of information, but we (the humans) decided it is quite useless to look at some numbers of information, so our programs display information in different way, for example the file can contain the number 97 in its raw form 01100001, and if seen from a text editor, it will show the letter 'a', but if seen from image editor it might use it to show some blue-ish color, colors are stored in 4 numbers: transparency(called alpha), red, green and blue, but different formats use different way to store the colors, so it will depend on which format does the program think it is going to display.

See it is all in the eye of the beholder, the file still contains only 97 (01100001), but each program will decide what to do with it.

There are also Directories(also called Folders) (which in many file systems are also files.. but we will get to that later), that can contain files or other directories. Example structure looks like:

```
/
├─ home/
│   └─ jack/
│       └─ type.py
│       └─ example.txt
│       └─ image.jpg
```

The word File is used because people that made those things up were thinking of filing cabinet, with lots of folders that contains pieces of paper.

```
/
/
/ _____
|_____
/_____/|||
|".__."|||
|_____|/|
|| ".__."|| /
||_____|/
|_____|/ Felix Lee
```

See how those pictures have sometimes names attached to them. Those are their creators, some pictures are from the 80s, so it is amazing we can still find out who made them. You should always attribute the work of an artist.

So the name 'File' and 'Folder' is used. If people were thinking of libraries I guess we would've been using Book for a file and Shelf for Folder? Anyway, my point is: all things have names, some of the names are strange and old, so don't worry if they make no sense.

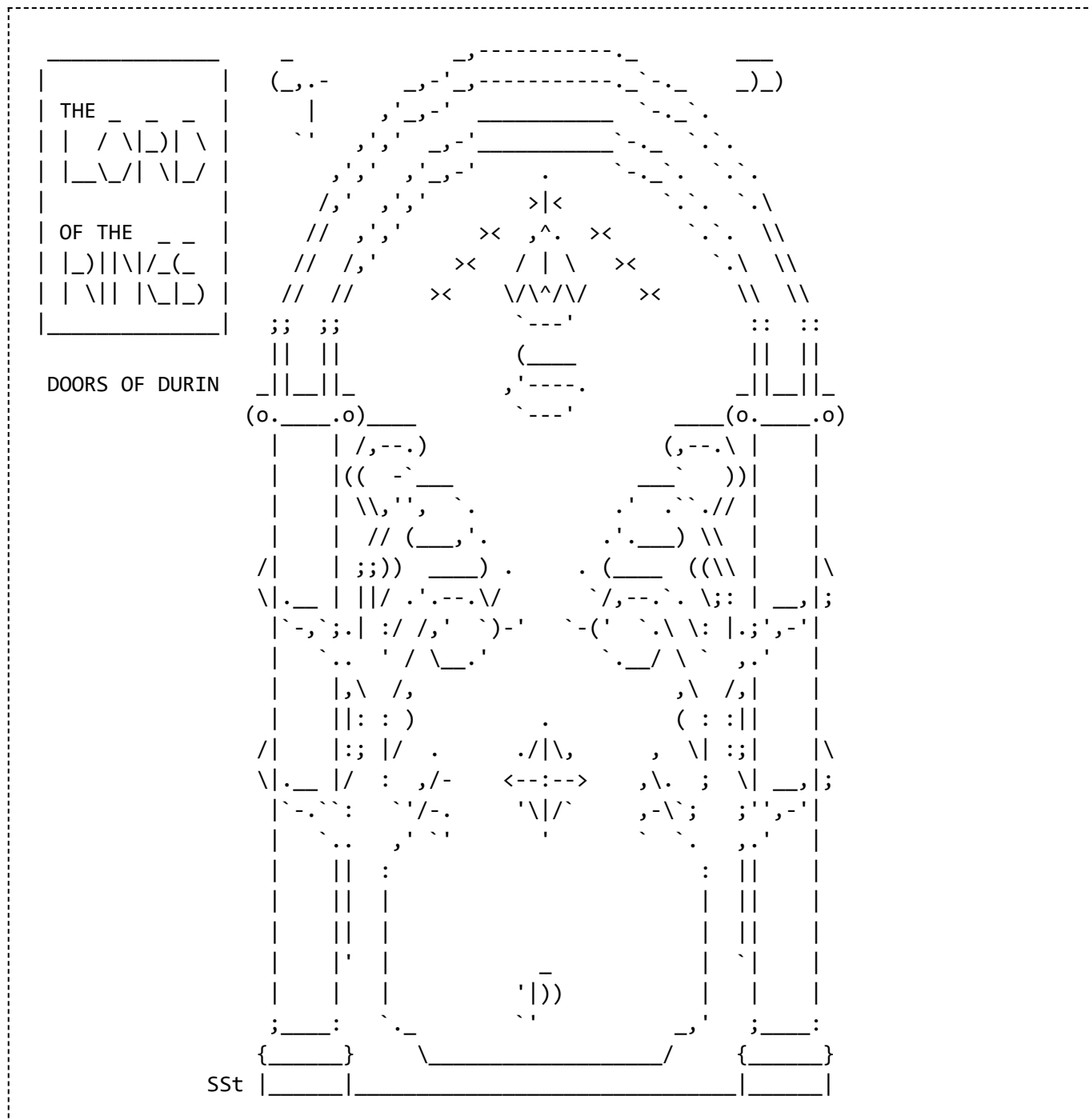
## [DAY-0] ASCII

Computers do not understand text, so when you see text it is usually some number that is displayed as a character, for example 65 is A, 66 is B, and so on. About 60 years ago some people agreed on a common way to map number to character, called THE ASCII STANDARD, super fancy name, for such a simple thing.

32	SPC	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	\$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_		

Try to decode: 110 105 99 101 32 119 111 114 107 33

## [DAY-0] Magic



Have you seen The Lord of the Rings? Do you remember the Door of Durin? Where Gandalf couldn't remember how to enter and the fellowship almost got eaten?

The experience through this book will feel like that. Sometimes you will be stuck, and I mean really really stuck, and at some point, as if the cosmos itself colludes, something will click, and you will level up. Sadly I can not know when this will happen to **you**, as every person is different.

## [DAY-0] How to Google

Anything you don't know how to do, you should use google to search for answers. However it is not easy to know what is good link to read and what is not, try to read from verified sources, like wikipedia, python.org, stackoverflow, mozilla and university websites, ask your parent to help.

There is a lot of scams on the internet, you will open websites that claim you have viruses on your computer and you need to pay 20\$ to clean it, or that you need to install this amazing program that will speed up your



internet. Sometimes on very trusted websites you will see ads that are pure scams, for example ads that looks like Download buttons to get the program you wanted, but it will just download a virus.

Its getting increasingly difficult to distinguish between good and bad information as well, some websites are just poorly written articles, so they can make people clicking on them in order to get money from ads. With time, and with the help of your parent you will learn how to recognize the good articles.

The rules about how to use the internet are:

- The internet is sus!
- Never put your real name or phone number or address anywhere
- Never download anything unless your parent approves
- Never run anything you downloaded (sometimes downloads just start automatically)
- Don't trust messages like 'click here to get free iPhone' or 'clean your computer for free'
- Always ask your parent for help if you are not sure.

There is noting free on the internet. Even this book, even though I am writing it for free, and you can get it for free, but I am hosting it on GitHub (and I don't pay for that), and GitHub, and therefore Microsoft, knows you are reading it, and later they will use that information to show you better ads, or maybe they will use it for some other purpose, like to train a machine to create artificial text, but one thing you must remember is that there is nothing free on the internet.

## [DAY-1] Touch Typing

Touch typing is just typing without looking, if you are super slow while typing you will get frustrated too soon.

Use <https://www.keybr.com/> or ask someone to recommend you something. After using the touch typing app for few days, write your own.

## [DAY-2] Install python

In general the first thing you should do when you don't know how to do something is to google it.

So google 'how to install python', you will see lots of results, some of them will tell you how to install the old version of python, and some will be for the too old or too new version of windows. This makes things a bit more difficult.

Try to download from <https://www.python.org/downloads/> and do it yourself, ask your parent for help if you are stuck.

## [DAY-2] Make a useful program

After you have installed python3 run IDLE (ask your parent for help) and go to File -> New File and type:

```
import random
letters = 'fjffjffjffjffghghghgaa;a;a;abcdefghijklmnopqrstuvwxy'
difficulty = 2

while True:
    q = ''
    for i in range(difficulty):
```

```
q = q + random.choice(letters)

a = input(q + ': ')
if a == q:
    difficulty += 1
else:
    if difficulty > 2:
        difficulty -= 1
```

Then hit F5 (this will ask you to save the file as well) to run the program. If it is too easy, try adjusting the difficulty or the letters.

The program will ask you to type some letters, after you are done, hit enter, and it will ask you for more or less letters. It will keep doing that forever. It kind of looks like this:

```
aj: aj
akg: akg
jjgh: jjgh
```

If you type it correctly, it will ask for a bigger sequence of characters, and if you make mistake it will ask for smaller.

Don't rush it.

Place your fingers properly on the keyboard, open <https://images.google.com> and search for 'touch typing' you will see many images of how to place your fingers.

## [DAY-3] Touch Typing using your program

Open IDLE again, go to File -> Open File, and find where you saved your program, then double click to open it. After opened hit F5 and it will start again.

Spend the rest of the time for the day touch typing. Don't rush it. Place your fingers properly on the keyboard and slowly type the letters.

## [DAY-4] HTML

We will start with HTML.

Making a page can be a bit overwhelming, so just start small. for example:

Open notepad and make a file on your desktop named "first.html", then write in it:

```
<html>
  <body>
    <h1>welcome!</h1>
    <p>this is my page</p>
  </body>
</html>
```

Open Windows Explorer and find the file on your desktop and double click on it.

Congrats! You have made your first web page!

Now try this:

```
<html>
  <body>
    <marquee>
      <h1>check this out!</h1>
    </marquee>
    <mark>Why would somebody use this?</mark>
  </body>
</html>
```

The rest of the day you will just try some other examples:

List:

```
<html>
  <body>
    <ul>
      <li>first <b>item</b></li>
      <li>second <i>item</i></li>
      <li>third</li>
    </ul>
  </body>
</html>
```

Horizontal Line:

```
<html>
  <body>
    first
    line<br>
    second line<br>
    <hr>
    third line<br>
  </body>
</html>
```

'br' means line break, when the browser sees it know it has to show whatever comes next on another line, 'hr' means horizontal rule, just like when you take your ruler and draw a line from left to right in the text book. You see, even though you write things in separate lines (like the word first and line are clearly on separate lines), the browser will not know what to do until you tell it to 'br'. Maybe in school you had a project you have to use PowerPoint to make a presentation? With the special words ul, li, br, hr, h2 and b you can make your own presentation in HTML!

## [DAY-5] HTML

The app you are using right now to see this website is called a web browser, I don't know if kids still call it like that, but adults do.

First the browser app downloads the web page, which is just a normal text file (it is what you see when you click on View Source), then it has to process it, in the same way you read pigpen code or ascii code, you look symbol by symbol and try to make sense out of it by making it into letters you understand.

It looks for word between '<' and then '>', when it sees '<' it knows what ever is between '<' and '>' is important word. Then it has kind of a table so it know what to do with different words, when it sees 'b' it knows that whatever is inside is gonna get **bold**. There are many words like that you should try for yourself 'b' 'i' 'u' 'h1', and many more.

For example, try this on your page: `<b>this is bold</b>`

## [DAY-6] Touch Typing

Spend half the time using your program and half the time on keybr.com.

# Chapter 1 - Week 1

---

```
day0: make new touch typing program
day1: touch typing on keybr.com
day2: touch typing using your program
day3: HTML
day4: HTML
day5: touch typing
day6: HTML
```

## [DAY-0] New Touch Typing Program

Make new file on your desktop, touchtyping.html and type in it:

```
<center>
  <pre style="font-size: 30px; letter-spacing: 4px;" id="question"></pre>
</center>

<script>
var makeNewQuestion = function() {
  var letters = 'fjffjffjffjffghghghgaa;a;a;abcdefghijklmnopqrstuvwxy'
  var difficulty = 10
  var q = []
  for (var i = 0; i < difficulty; i++) {
    q.push(letters.charAt(Math.floor(Math.random() * letters.length)));
  }
  return q
}

var currentQuestion = makeNewQuestion()
var questionElement = document.getElementById("question")
```

```

questionElement.innerHTML = currentQuestion.join("")

var position = 0
document.body.addEventListener('keydown', (e) => {
  if (e.key == currentQuestion[position]) {
    position++
    if (position == currentQuestion.length) {
      currentQuestion = makeNewQuestion()
      position = 0
    } else {
      currentQuestion[position-1] = '<b>' + currentQuestion[position-1] + '</b>'
    }
    questionElement.innerHTML = currentQuestion.join("")
  }
})
</script>

```

Don't worry about what it all means. Just type it character by character. Now double click on the file and try out your program, just start typing as the web page is open, and characters will become bold as you type them.

## [DAY-1] Touch Typing

Today just spend the day chilling on [keybr.com](https://keybr.com)

## [DAY-2] Touch Typing using your program

Open [touchtyping.html](#) and spend the day using your awesome program. Once you open it, right click and then click on [View Source](#) to remember that you actually wrote this! Good job!

## [DAY-3] HTML

From now on the HTML examples will be more code and less text, so you just have to go through them with your parent.

```

<html>
  <body>
    <h2>welcome!</h2>
    <p>
      this is my <b>first web page</b> it has also <i>weird twist</i>!
    </p>
    <hr>
    <p>
      and a line
    </p>
  </body>
</html>

```

Try to touch type as you are writing this, no need to hurry.

```

<html>
<body>
  <h2>welcome!</h2>
  <p>
    this is my <b>first web page</b>!
  </p>
  and it has a bug!<br>
  <button onclick="document.body.appendChild(this.cloneNode(true))">🐛</button>
</body>
</html>

```

This is a funny one, you might not know how to write 🐛, but just google 'bug emoji' and copy it from there. After you open the web page, click on the bug to see what happens.

You see we add this `onclick=...` which is code that is going to run every time you click on the bug, and what this code does it copies the button and adds it to the page, including the `onclick=...` stuff, so the other button you can press and it will copy itself again!

Try to add more buttons by yourself with other emojis, 🐛, 🐞, 🐙 or 🐜 for example.

## [DAY-4] HTML

Making a presentation with HTML

```

<html>
  <body style="font-size: 24px;">
    <h1>Presentation For School</h1>
    <small>by: John, from class 4</small>
    <br>
    <br>
    <br>
    <br>
    <br>
    <br>
    <hr>

    <h1>First Slide</h1>
    <ul>
      <li>Something</li>
      <li>very <b>important</b></li>
      <li>and very <i>strange</i></li>
    </ul>
    <br>
    <br>
    <br>
    <hr>
    <h1>Second Slide</h1>
    <ul>
      <li>Something Else</li>
      <li>very <u>underline</u></li>
      <li>and very <del>weird</del></li>
    </ul>

```

```
<br>
<br>
<br>
<hr>
<h1>Third Slide</h1>
<ul>
  <li>... surely we can come up with a line</li>
  <li>... and another line</li>
</ul>
<br>
<br>
<br>
<hr>
</body>
</html>
```

## [DAY-5] Touch Typing


Use your program to touch type or go to [keybr.com](https://keybr.com).

Don't rush.

## [DAY-6] HTML

This day is completely free style, try to put all kinds of HTML words inside each other, for example:

```
<html>
  <body>
    <marquee>
      <ul>
        <li>
          <h1>
            <i>
              hello
            </i>
          </h1>
        </li>
      </ul>
    </marquee>
  </body>
</html>
```

Try to add the self cloning  button to that page.

## Chapter 2 - Week 2

```
day0: HTML tables
day1: HTML tables and lists
day2: HTML multiplication table
```

day3: HTML multiplication table  
day4: touch typing  
day5: HTML Links  
day6: HTML fun

## [DAY-0] Tables

Tables are amazing, in fact, tables are one of the things that makes the modern world work. In your school for example, there is a table with the name of each student and their score. Or in your TV there is a table with each program number and the program there (e.g. 24 is 24 Kitchen). On your iPhone there are many programs that use many tables internally to make decisions. There is of course the ASCII table, which can tell you how to get from a character to its raw representation and the other way around. The periodic table is also a famous example. The table on the bus stop tells you which bus will come at what time.

We use tables everywhere.

Here are few examples:

power	pokemons
electricity	Pikachu
fire	Charmander
grass	Bulbasaur
poison	Bulbasaur

Or something you have probably seen, a multiplication table:

a	x	b	=
5	*	5	25
5	*	6	30
5	*	7	37

Make new file (or open the some old html page) and type this:

```
<html>
  <body>
    <table border="10">
      <tr><th>Name</th><th>Year</th></tr>

      <tr>
        <td>
          Donald Duck
        </td>
        <td>
          1937
        </td>
      </tr>
```



```

<tr>
  <td>
    Daisy Duck
  </td>
  <td>
    1940
  </td>
</tr>
<tr>
  <td>
    Scrooge McDuck
  </td>
  <td>
    1947
  </td>
</tr>
<tr>
  <td>
    Ludwig Von Drake
  </td>
  <td>
    1961
  </td>
</tr>
</table>
</body>
</html>

```

Only 3 new tags we have to learn **table**, **tr**, **td**, **th** are the main things we will work with, **tr** means table row and **td** is table data cell, or a column, **th** is table header. All the html key words you have learned so far are called **tags** or **elements**.

Make `<table border="100">` and see what happens.

## [DAY-1] More tables

```

<html>
  <body>
    <table border="10">
      <tr><th>Name</th><th>Year</th></tr>

      <tr>
        <td>
          <marquee>
            hello world
          </marquee>
        </td>
        <td>
          <marquee>
            hello universe
          </marquee>
        </td>
      </tr>
    </table>
  </body>
</html>

```

```

    </tr>
    <tr>
      <td>
        <b>this does not move</b>
      </td>
      <td>
        <marquee>
          <i>this is italic</i>
        </marquee>
      </td>
    </tr>
  </table>
</body>
</html>

```

Now make list in a table

```

<table border=100>
  <tr>
    <td>
      <ul>
        <li>a</li>
        <li>b</li>
        <li>c</li>
      </ul>
    </td>
    <td>
      <ol>
        <li>a</li>
        <li>b</li>
        <li>c</li>
      </ol>
    </td>
  </tr>
</table>

```

Super small difference between `ul` and `ol`, one means 'unordered list' and the other one 'ordered list', so if you use `ol` every `li` will have its own number

## [DAY-2] Multiplication table

Start writing the whole multiplication table (will take more than 1 day)

## [DAY-3] Multiplication table

Finish writing the whole multiplication table.

## [DAY-4] Touch Typing

Spend the day touch typing.

## [DAY-5] Links

The most powerful feature of HTML is to be able to link to another place in the internet, try this:

```
<html>
  <body>
    hi, <a href="https://wikipedia.com">click here</a> to go to wikipedia
  </body>
</html>
```

The [a](#) tag with its [href](#) attribute is one of the most important things of the modern internet, it means I can have a web page, and I can link to someone else's web page, and they can link to someone else and so on..

Whatever is in the [href](#) attribute this is where the browser will go after you click on whatever is inside the [<a>](#) [</a>](#), in our case [click here](#). So after you click on [click here](#) your browser will go to wikipedia.com, and from there when you click somewhere it will go to wherever that [a's href](#) points to.

## [DAY-6] HTML fun

Rainbow:

```
<html>
  <body>
    <h1>🌈</h1>
    <h1 style="color: red;">red</h1>
    <h1 style="color: orange">orange</h1>
    <h1 style="color: yellow">yellow</h1>
    <h1 style="color: green">green</h1>
    <h1 style="color: blue">blue</h1>
    <h1 style="color: indigo">indigo</h1>
    <h1 style="color: violet">violet</h1>
  </body>
</html>
```

You don't always need [html](#) and [body](#), the most browsers will show a page anyway, but it is good practice to do it proper and use [html](#) and [body](#) tags. Anyway, the next example is not "up to code".

It will make a sheep that follows your mouse, it's pretty cool.

```
<div id='sheep' style='position: absolute'>🐑</div>
<script>
  var sheep = document.getElementById('sheep')
  document.body.onmousemove = (e) => {
    sheep.style.left = (e.clientX - 5 + 'px');
    sheep.style.top = (e.clientY - 5 + 'px');
  }
</script>
```

And a ghost button! it will remove itself when you press it.

```
<button onclick="this.parentNode.removeChild(this)">👹</button>
```

The way it works is `onclick=...` will call the code when you click on the button, in this case it says, "whoever my parent is, tell it to remove myself from its children list".

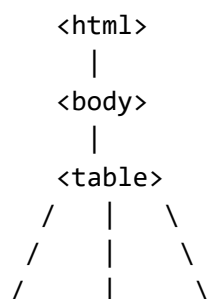
You see, HTML is like a tree, let me show you.

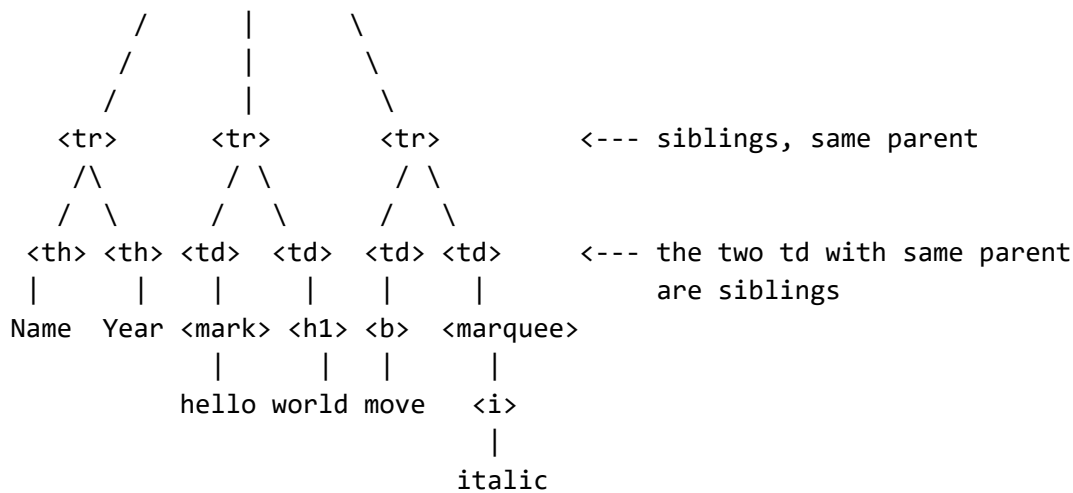
lets say our page is:

```
<html>
  <body>
    <table border="10">
      <tr><th>Name</th><th>Year</th></tr>

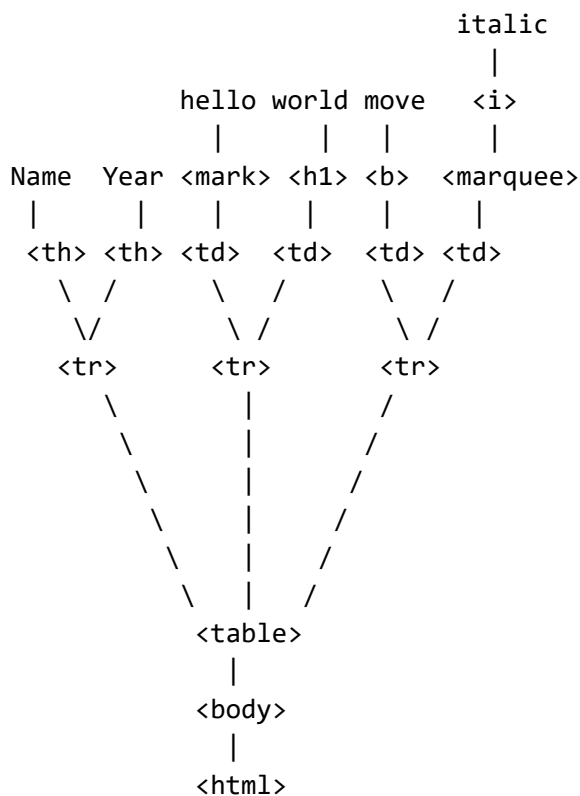
      <tr>
        <td>
          <mark>
            hello
          </mark>
        </td>
        <td>
          <h1>
            world
          </h1>
        </td>
      </tr>
      <tr>
        <td>
          <b>move</b>
        </td>
        <td>
          <marquee>
            <i>italic</i>
          </marquee>
        </td>
      </tr>
    </table>
  </body>
</html>
```

It actually is a tree, every tag has children, and siblings





Well trees in the world are actually the other way around haha. Leave it to programmers to not know how to make a tree :)



But you can see how all things that have the same parent are related, like brothers and sisters.

This is very important, we will spend the next week thinking about it as well.

## Chapter 3 - Week 3

day0: View Source

day1: Inspect

day2: Images  
day3: Licenses  
day4: Touch Typing  
day5: HTML  
day6: HTML fun

## [DAY-0] View Source

Browsers on laptops or pcs will allow you to rightclick on any page and then click on [View page source](#), this will show you the HTML of the page your are looking at.

Open <https://archive.org> then right click on the pager and then click on [View page source](#).

We will spend the rest of the day looking at some page sources

- <https://archive.org>
- <https://www.spacejam.com>
- <https://www.asciart.eu/>
- <https://www.wikipedia.org>
- <https://www.gutenberg.org/>
- <https://digitalcomicmuseum.com/>
- <https://www.terrypratchettbooks.com/>
- <https://www.goodreads.com/>

## [DAY-1] Inspect

Make new html file and write the following example:

```
Right click on this page, click Inspect and then go to Console.  
<script>  
    console.log('🐱🐱🐱🐱 You can see that in the console. 🐱🐱🐱🐱')  
    console.log('🐱🐱🐱🐱 type 12312*18237978123 in the console and see what  
happens. 🐱🐱🐱🐱')  
</script>
```

Open the file and then click F12, or right click on the page and click Inspect. You will see the logs.

This will open the developer tools of your browser, you will be able to see the console, where you can run small programs or see the errors and the logs of the current page. Try writing [12312\\*18237978123](#) in the console and see the result.

In the inspector you can also modify the HTML that you see. This wont modify the actual page on the server you are downloading from, but you can try things to see how they would look. There are actually many scammers that use this method to lie to people as if they are giving them money, or they say 'I will teach you how to add 10000 robux to your account if you pay me 5\$' and they just show you how to inspect and edit the html you see. This does not actually give you robux of course, it just modifies your local version of the html to show different value and after you reload the page you will see the old value.

Lets try it. Make a new file with this content:

```
<html>
  <body>
    Your Robux Balance is: <b>0$</b>
  </body>
</html>
```

Open the file and then right click on `0$` and click inspect. Then you will see `<b>0$</b>` if you double click on it, you will be able to change it, and you will see the new version. Now reload the page, and you will see the old value is back.

## [DAY-2] HTML Images

To show an image you need the `img` tag, and give it `src` attribute with the address of the image, for example if I want to display `https://picsum.photos/id/237/200/300` I have to do type it like this:

```
<html>
  <body>
    
    
  </body>
</html>
```

Lets put the images in a table:

```
<html>
  <body>
    <table>
      <tr>
        <td>
          
        </td>
        <td>
          
        </td>
      </tr>
    </table>
  </body>
</html>
```

Now of course we can have `img` in `a`, try this:

```
<html>
  <body>
    <table>
      <tr>
        <td>
          <a href="https://wikipedia.com">
            
          </a>
        </td>
      </tr>
    </table>
  </body>
</html>
```

```

        </a>
      </td>
    <td>
      <a href="https://gutenberg.org">
        
      </a>
    </td>
  </tr>
</table>
</body>
</html>

```

Now you can click on the dog or the nose and it will lead you to the pages you link to.

Is it possible to have `img` and text in `a`? Well I am glad you asked!

```

<html>
  <body>
    <table>
      <tr>
        <td>
          <a href="https://wikipedia.com">
            
            <br>
            This dog leads to wikipedia
          </a>
        </td>
        <td>
          <a href="https://gutenberg.org">
            
            <br>
            This cat (I think) leads to guttenberg.org
          </a>
        </td>
      </tr>
    </table>
  </body>
</html>

```

I even have `<br>` in the `a`, now you can click either on the image or on the text.

There is one more very important attribute of the `img` tag, and this is the `alt` attribute, it is used by people who are blind or visually impaired to know what kind of picture is on the page. In our example we can put `alt="puppy"` on the puppy picture.

```

<p>
  Hello, and welcome to my page.
  I hope you will like this image.
</p>


```



If a blind person visits this page, they use a screen reader that reads most of the text on the page, and if we have `alt` attribute on images it will say **Image** ... and whatever the value of `alt` is, in our case "puppy". So the reader will say:

Hello, and welcome to my page. I hope you will like this image.

Image puppy.

If you have an image that has no information, like it is just there to make the site pretty, use `alt=""` then the screen reader will skip it.

## [DAY-3] Licenses

If you take a picture of something you are the owner of that picture, and you can put it on your website and say you have the rights of that picture. It is up to you to decide if people should link to it by deciding what under license you will publish the image. There are many licenses you can choose from, there are some that say 'this picture is free for anyone to do whatever they want' or 'you can republish the picture but you cant make money selling it' or 'you can publish it but not print it in books' etc etc.

Complicated stuff this licensing, but the thing you have to remember, it is up to the creator to decide under what license their work can be distributed.

Some licenses you can check out with your parents:

- Creative Commons Licenses CC-BY <https://creativecommons.org/licenses/>
- GPL <https://www.gnu.org/licenses/gpl-3.0.en.html>
- MIT License <https://opensource.org/licenses/MIT>
- Apache License <https://www.apache.org/licenses/LICENSE-2.0>
- Public Domain [https://en.wikipedia.org/wiki/Public-domain-equivalent\\_license](https://en.wikipedia.org/wiki/Public-domain-equivalent_license)
- Copyright <https://en.wikipedia.org/wiki/Copyright>

When you want to use someone else's work and it is not clear what license it uses, it is best to ask them. At least that is what I do. People are usually nice and they give me permission to use their work.

It is somewhat controversial what is the right way forward, which license to use and what is the best for you and for the world. Ask to your parent about what happens when you take a picture of a painting of a picture of a video and then you edit the video to include the picture you took in it..

Check out <https://tldrlegal.com/> for super short description of licenses

## [DAY-4] Touch Typing

relax and spend the day touch typing

## [DAY-5] HTML Title

```
<html>
  <head>
    <title>THIS IS SPARTA</title>
```

```
</head>
<body>
  <marquee>
    <ul>
      <li>
        <a href="https://wikipedia.com">click me</a>
      </li>
      <li>
        <mark>hello world</mark>
      </li>
    </ul>
  </marquee>
</body>
</html>
```

Usually bodies have a `<head>` hehe.

In html in the `head` tag you can put things that will help the browser. For example, what is the `title` of this page? or what language is this page? Who is the author of this page? etc..

You can also put there some style, changing the body's background color, or the color of the text in the `h1` or `p`:

```
<html>
  <head>
    <style>
      body {
        background: blue;
      }
      p {
        color: pink;
      }
      h1 {
        color: cyan;
      }
    </style>
    <title>some title</title>
  </head>
  <body>
    <center>
      <p>
        Hello Universe
      </p>
      <h1>hello world</h1>
    </center>
  </body>
</html>
```

The *language* we use inside the `style` tag is called `CSS`, and it is quite simple (on its surface). We wont get deep into it for a while, but if you are interested check out at <https://developer.mozilla.org/en-US/docs/Web/CSS>

You see how in certain tags we can use different language, not html, like in `<style>` we use CSS, in `<script>` we use JavaScript, we are going to do more work with it soon, but you can check out at <https://developer.mozilla.org/en-US/docs/Web/JavaScript>

## [DAY-6] HTML fun

### Many buttons

```
<button onclick="makeManyButtons()">🐱</button>
<script>
  function makeManyButtons() {
    // try to make million buttons!

    for(var i = 0; i < 100; i++) {
      var button = document.createElement('button')
      button.innerText = '🐱'
      document.body.appendChild(button)
    }
  }
</script>
```

### Replicator

```
<button onclick="replicate(10, '🐱')">🐱</button>
<script>
  function replicate(n, buttonText) {
    // try to make million buttons!

    for(var i = 0; i < n; i++) {
      var button = document.createElement('button')
      button.onclick = function () {
        replicate(20, '🐱')
      };
      button.innerText = buttonText
      document.body.appendChild(button)
    }
  }
</script>
```

### Canvas

```
<html>
<body style="padding: 0; margin: 0">
<canvas id="squares"></canvas>
<script>
var canvas=document.getElementById('squares');
var ctx=canvas.getContext("2d");
canvas.width=window.innerWidth;
canvas.height=window.innerHeight;
```

```
function draw() {
  ctx.fillStyle= '#' + Math.floor(Math.random()*16777215).toString(16);
  ctx.globalAlpha=0.5;

  var size=Math.floor((Math.random()) * 60) + 1;
  var x = Math.floor(Math.random()*canvas.width)
  var y = Math.floor(Math.random()*canvas.height)

  ctx.fillRect(x, y, size, size);
};

setInterval(draw,10);
// why dont you try to make them rectangles instead of squares?
// or maybe even circles? google for 'canvas fillRect' and 'canvas arc'
</script>
</body>
</html>
```

The last one is quite crazy! But I am sure you will like the result when done.

## Chapter 3 - Week 3

```
day0: Basics of Basics
day1: Dungeon Game
day2: Dungeon Game
day3: Favorite Food
day4: Dungeon Game
day5: Dungeon Game
day6: Dungeon Game
```

### [DAY-0] Basics of Basics

Make a new file form IDLE, and write in it:

```
print("Hi!")
```

Hit F5 and enjoy!

Your first(or second) python program.. Quite useless, but nevertheless. A program is a program!

`print()` is called a function, functions are kind of mini useful programs, this one will print whatever you tell it. In this case `print("Hi!")` will output `Hi!`. Pretty amazing, I hope one day to explain to you what goes into showing a character on your screen. Remind me to show you Ben Eater's 6502 videos when you grow up.

"Hi!" is a string, strings are just series of characters, you can make them in python with `"something"` or `'something'`, either single quotes or double quotes.

```
while True:
    print("Hi!")
```

`while True` that means forever. so you will see:

```
Hi!
Hi!
Hi!
Hi!
Hi!
Hi!
Hi!
Hi!
...
```

Hi forever.

```
while 1 == 1:
    print("Hi!")
```

`1 == 1` is also True, so this is the same as `while True`

So `while <condition>` will keep running the code **inside** it, while the condition holds true, which in the case of `1 == 1` will always be the case.

```
while True:
    zzz = input("what is your name: ")
    print(zzz)
```

`input` asks you to type something, and it returns whatever you typed.

`zzz = input(...)` takes whatever input returns, and puts it into memory that we can use later. `zzz` is just a name I picked so I can refer to this value later in the program, in this case on the next line when I do `print(zzz)`

`print(name)` prints whatever is in the memory pointed by `zzz`.

`zzz` is called a **variable**, you can choose the names of your variables, and `zzz` is surely a poor name, because if I use it 100 lines later in the code, I will forget what kind of information it stores, so usually we give names of the variables that make sense, for example:

```
while True:
    name = input("what is your name: ")
    print(name)
```

A group of statements with common parent, is called **code block**, in python we use spaces to say what belongs where, so the closest **while, for, if, elif, else** going up is the parent.

```
while True:
    name = input("what is your name: ")
    print(name)
```

Will throw an error: **IndentationError: unexpected indent** because it makes no sense, there is no valid parent to **print**, This is quite unique in python, most other languages make code blocks with **{}**, but python makes it prettier and easier to read by using indentation (the spaces/tabs). **print(name)** has 4 spaces, while **name = input..** has 2 spaces, so python expect everything in the **while:** to have 2 spaces unless a new block starts.

```
while True:
    name = input("what is your name: ")
    if name == "pikachu":
        print(name)
```

This works fine, **print(name)** now has valid parent that also has a parent.

```
while True:
    | \
    | \
    | \
    | \
    if:  name = input
    |
    print(name)
```

**while True** has two children, **name = input ..** and **if name == ..** so they must have same indentation, then **if ..:** also expects a code block, so the children of **if name == ..** need to have one more indentation to whatever indentation the **if** had.

Anyway..

Don't worry too much about it, just don't panic when you see **IndentationError**, and I can promise you, you will see a lot of those. Look at where you got the spaces wrong, and if the block has correct parent.

You have seen by now in JavaScript we use **{ }** to group statements into blocks.

```
if (name == "pikachu") {
    console.log("pikaaaachuuuuu")
    console.log("evolutiooonn!")
}
```

Now lets break out of the loop!

```
while True:
    name = input("what is your name: ")
    if name == "pikachu":
        break
```

`if name == "pikachu":` will run the code inside `if` if whatever is in the `name` variable is equal to the string "pikachu". In this case its only 1 instruction inside it, the `break` instruction.

`break` breaks out of the closest `while` loop, so basically this program will ask what is your name until you type pikachu. It is a bit boring because we never actually print what you typed.

```
while True:
    name = input("what is your name: ")
    print(name)
    if name == "pikachu":
        break

print("DONE")
```

There we go, now it will ask you to type a name, it will print whatever name you typed, and then it will check if the name is equal to "pikachu" it will break the while loop and stop asking. We can actually write this program in a different way. After you break out of the loop, it just continues to execute the program below, in this case will print DONE.

```
name = ''
while name != "pikachu":
    name = input("what is your name: ")
    print(name)

print("DONE")
```

MAGIC!

we start by making name equal to empty string, a string with no characters, and then we check is the statement `name != "pikachu"` True? If this is True we will execute the code **inside** the while loop, after the last instruction in the loop, the program jumps back to `while name != "pikachu"` and checks again is name still not equal to "pikachu"? if the statement is False it will not run the code **inside** but will continue, in the above example it will print DONE, because this is the first thing **after** the while loop.

Lets make a more complicated one, this one will ask you what is your name, and if you answer pikachu will print pikaaaaaachuuuuu and stop, any other answer it will print "Hello, " + answer, so if you type "Jane" it will show "Hello, Jane" and it will ask again.

```
while True:
    name = input("what is your name: ")
    if name == "pikachu":
        print("pikaaaaaachuuuuu")
```

```
break
else:
    print("Hello, " + name)
```

In python you can add two strings, if you type "charmander" for name, `x = "Hello, " + name` will make x to be equal to "Hello, charmander". You can't do `"hello" - name`, but you can do `"hello" * 5` and get "hellohellohellohellohello".

`while, if, else, break` are keywords, kind of like `<html>` and `<body>`, `<h1>` etc, those are coming from python itself. There are not many python keywords, for reference, this is the **complete** list:

```
False
True
None

and      -> name == "pikachu" and age == "33"
or       -> name == "pikachu" or name == "charmander"
not      -> not name == "pikachu" is the same as name != "pikachu"

for      -> used if you know how many times you want to do something
while    -> do something until condition is True
break    -> break out of the for or while loop
continue -> go to the start of the while/for loop and continue from there

if       -> if something is true
elif     -> else if something else is true
else     -> else do this

in       -> checks if something is in something else, e.g. "pika" in name
def      -> make a function

as
assert
async
await
class
del
except
finally
from
global
import
is
lambda
nonlocal
pass
raise
return
try
with
yield
```



THATS THE WHOLE LANGUAGE, there are no more keywords.

## [DAY-1] Dungeon Game

```
print("Welcome to the forest!")
print("This is a world of magic and wonders. You are on a cross road, you can go
north east south or west.")
print("South leads to the swamps, where the aligators live")
print("West leads to the mountains, where the yeti lives")
print("North leads to the jungle, where the tigres live")
print("East leads to the desert, where the meerkats live")
print("")

what = input("what would you do next: ")

if what == "east":
    print("Welcome to the desert")
    print("It is very hot, and you need remember to put sun screen.")
    print("Oh, you remember to put your hat as well.")
    print("In the distance you see a meerkat approaching.")
    print("What would you do? Run or Fight the meerkat?")
    print("")

    what = input("what would you do next: ")
    if what == "run":
        print("You start running, and the meerkat is super fast, it catches you in no
time.")
    elif what == "fight":
        print("You try to fight it, but turns out it was friendly and wants to become
your friend.")
        print("What would you do?")
        print("")
        what = input("Do you want to be its friend: ")

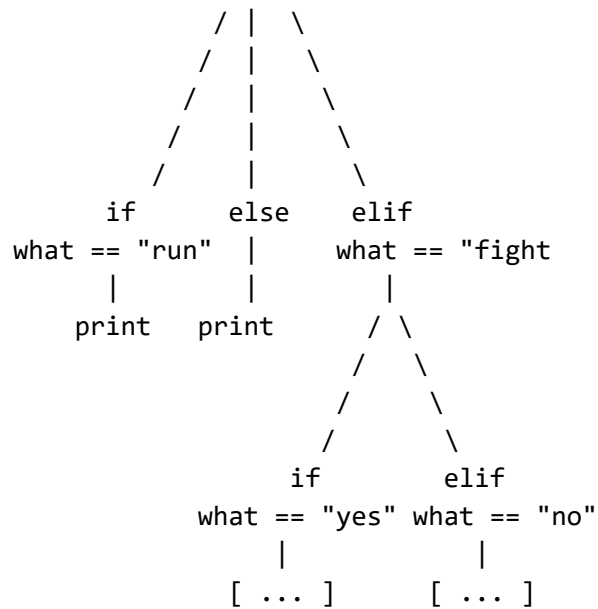
        if what == "yes":
            print("Great! Now the meerkat wants to introduce you to his family.")
        elif what == "no":
            print("The meerkat starts crying!")

    else:
        print("I dont understand: " + what)
```

That is a lot of typing.

In python there are few very important things. First even though it doesn't look like it, it is actually a tree, like HTML tree. The parent of `print("The meerkat starts crying!")` is `elif what == "no":`

```
      |
      |
if what == "east"
      |
```



Lets discuss another example:

```

a = "he"
b = "ll"
c = "o"
if a == "he":
    if b == "ll":
        if c == "o":
            print("hello")

```

Can you tell who is the parent of who?

BTW, you can also use **and**, **or** and **not** when you want to see if something is **True**

```

a = "he"
b = "ll"
c = "o"
if a == "he" and b == "ll" and c == "o":
    print("hello")

```

## [DAY-2] Dungeon Game

Our game is quite limited, and a quick step to improve it is to make you ask where you want to go until you pick one of the options.

```

def ask(possible_answers):
    answer = ''
    print("---") # print empty line
    while True:
        answer = input("> What would you do next: ")
        if answer not in possible_answers:

```

```

        print("> try again, it must be one of:", possible_answers)
    else:
        return answer

print("Welcome to the forest!")
#...

what = ask(["east","west","north","south"])

if what == "east":
    print("Welcome to the desert")
    #...

    what = ask(["run","fight"])
    if what == "run":
        print("You start running, and the meerkat is super fast, it catches you in no time.")
    elif what == "fight":
        print("You try to fight it, but turns out it was friendly and wants to become your friend.")
        # ...
        what = ask(["yes","no"])
        if what == "yes":
            print("Great! Now the meerkat wants to introduce you to his family.")
        elif what == "no":
            print("The meerkat starts crying!")
    elif what == "north":
        print("Welcome to the north pole, it is super cold here.")

```

`ask` is a function, just like `print` or `input`, it will keep asking you `> What would you do next:` until the answer you type is in the `possible_answers` list, and if it is it will return it to wherever it is called from. So when we have `zzz = ask(["yes","no"])` the value of `zzz` will be whatever is returned from `ask`. Same as `name = input("what is your name")` will put in `name` whatever is returned from `input` which is whatever you typed on your keyboard.

To make a function in python you need to use the `def` keyword.

```

def sum(a,b):
    return a + b

r = sum(1737,1231231)
print(r)

```

Whatever is between `def` and `(` is the name of the function, in the above example its `sum`. Between `( )` you put in the name of the variables you expect to use when someone calls your function. I want to sum two numbers, I dont know what the numbers are, so I just make two variables `a`, `b` and expect whoever calls my function to give me the numbers, like `r = sum(1737,1231231)`

## [DAY-2] Dungeon Game

FIGHT TO THE DEATH!

```
import random
import time

def fight(playerHP, enemyHP, enemy_name):
    # fight to the death!

    while playerHP >= 0 and enemyHP >= 0:
        punch = random.randint(0, 20)
        if random.choice(["player", "enemy"]) == "player":
            playerHP = playerHP - punch
            print("<" + enemy_name + "> hits you for " + str(punch) + ", " + str(playerHP) + " left")
        else:
            enemyHP = enemyHP - punch
            print("you hit <" + enemy_name + "> for " + str(punch) + ", " + str(enemyHP) + " left")

        time.sleep(1)

    return playerHP

fight(100, 50, "meerkat")
```

`import` imports a module.

`random` and `time` those are the `modules` you import, modules are just a group of functions that you can use, for example `time.sleep(1)` makes python sleep for 1 second, it calls the function `sleep()` in the `time` module. `random.choice()` you can give a list of things to choose from, and it will randomly select one of the list. `random.randint(0,20)` means give me a random number between 0 and 20.

`str` is needed to convert integer to string, because 1 is not the same as "1", "1" is actually the ASCII value of the character 1, so somehow we have to convert the raw number 1 to ASCII code 61, and `str()` does that.

lets use it now in our dungeon game

```
import random
import time

def ask(possible_answers):
    ...

def fight(playerHP, enemyHP, enemy_name):
    ...

...

```

```

health = 100

what = ask(["east","west","north","south"])
if what == "east":
    ...
    what = ask(["run","fight"])
    if what == "run":
        ...
    elif what == "fight":
        meerkatHP = 50
        health = fight(health, meerkatHP, "meerkat")
        if health <= 0:
            print("GAME OVER! THE MEERKAT BEAT YOU")
        else:
            print("You won against the meerkat, you have " + str(health) + " HP left))

```

## [DAY-3] Favorite Food

Today you have to make only two programs:

```

bad = ["broccoli","chocolate","pineapple"]

while True:
    food = input("what is your favorite food: ")
    if food in bad:
        print("ewwww I hate " + food)
    else:
        print("yumm, I love " + food)

```

And the second one:

```

bad = ["broccoli","chocolate","pineapple"]
good = ["pizza","popcorn"]
while True:
    food = input("what is your favorite food: ")
    if food in bad:
        print("ewwww I hate " + food)
    elif food in good:
        print("yumm, I love " + food)
    else:
        print("I have never tried " + food)

```

Great job!

Spend the rest of the day touch typing.

## [DAY-4] For

`for` does something number of times, if I want to print the numbers from 0 to 99, I could do:

```
for i range(100):  
    print(i)
```

or

```
for i in range(10, 20):  
    print(i)
```

will print the numbers from 10 to 19

```
colors = ["red","green","blue"]  
for color in colors:  
    print(color)
```

will print each of the elements in the list. If you want to print each character of a string in a similar way you can do:

```
name = "jack"  
for c in name:  
    print(c)
```

prints:

```
j  
a  
c  
k
```

## [DAY-5] Love Tester

```
def love_test(a,b):  
    sum = 0  
    for c in a+b:  
        print(c + ': ' + str(ord(c)))  
        sum += ord(c)  
  
    print('sum is:' + str(sum))  
    return sum % 100
```

```
while True:  
    nameA = input("first name: ")  
    nameB = input("second name: ")
```

```
print("love test: " + str(love_test(nameA,nameB)))
```

`ord` takes the ascii code of a character.

`%` is the remainder, so `109 % 100` is 9, basically what is left after you can cleanly divide two numbers, `812 % 100` is 12, you can fit 100 in 819 exactly 8 times, and then there is 12 left, this 12 is the remainder.

So in this program we take two names into the variables `nameA` and `nameB` and then we give them to the `love_test` function, which sums the ascii code of `nameA+nameB`, and then returns the remainder of 100.

```
first name: jack
second name: jane
j: 106
a: 97
c: 99
k: 107
j: 106
a: 97
n: 110
e: 101
sum is:823
love test: 23
```

BTW, now you know how those "professional" love testers are made, so if you use <https://www.lovetester.nl/> or something similar.. don't read too much into the result. You can easily tweak it to do whatever you want.

## [DAY-6] Touch Typing

Whoaa it has been a difficult week.

Relax with some touch typing.