

# CIS 122

## Strings and Things

# Math Module Madness

- Python has a few built in mathy functions
  - int
  - abs
  - round
- But where's the heavy duty stuff?
  - log
  - sin
  - factorial

# Math Module Madness

- Python stores extra variables / functions in modules
  - math
  - random
  - time
- Need to **import** module before using it
  - `>>> import math`
  - `>>> math.sin(7)`
- Modules use **dot notation**
- Why not make everything available all the time?

# Math Module Madness

- So what's in the math module?
- Ask python for `help`
  - `>>> help(math)`
  - Make sure you import math first...
- For a briefer list, use `dir`
- IDLE makes things even easier
  - Tries to finish your word when you press `<TAB>`
  - What happens if you type "math." + `<TAB>` ?

# String Things

- We can perform mathematical operations with numbers
- What would we like to do with strings?

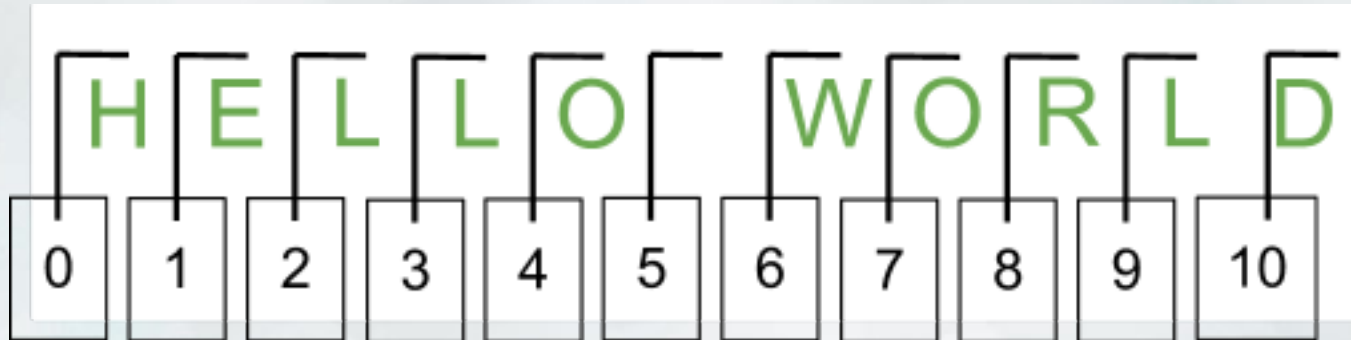
# String Things

- String Length
  - `>>> len("abc")`
  - Works on any object with a "length"
- String Comparison
  - `>>> "a" < "b"`
  - What are Python's rules for string ordering?
  - (The `ord` function offers some insight)
- Substrings
  - Need to know a little more about strings first...

# Anatomy of a String

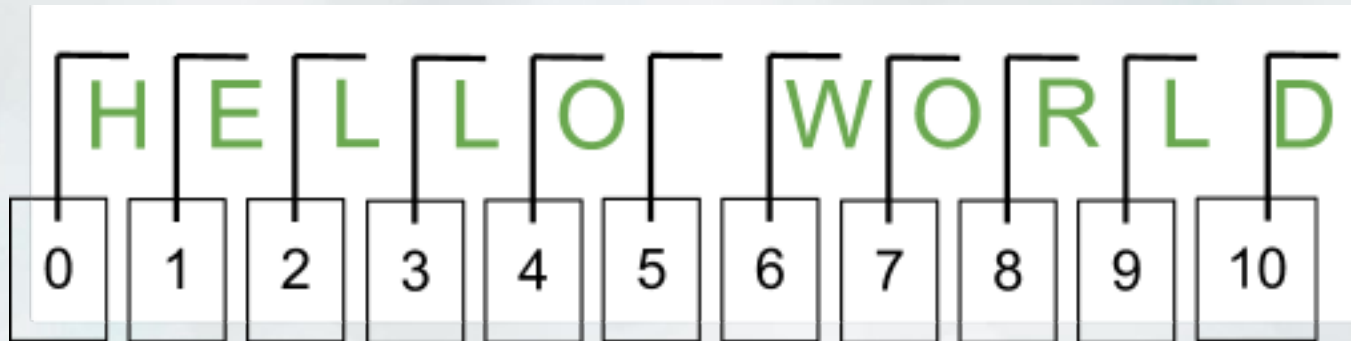
H E L L O W O R L D

# Anatomy of a String



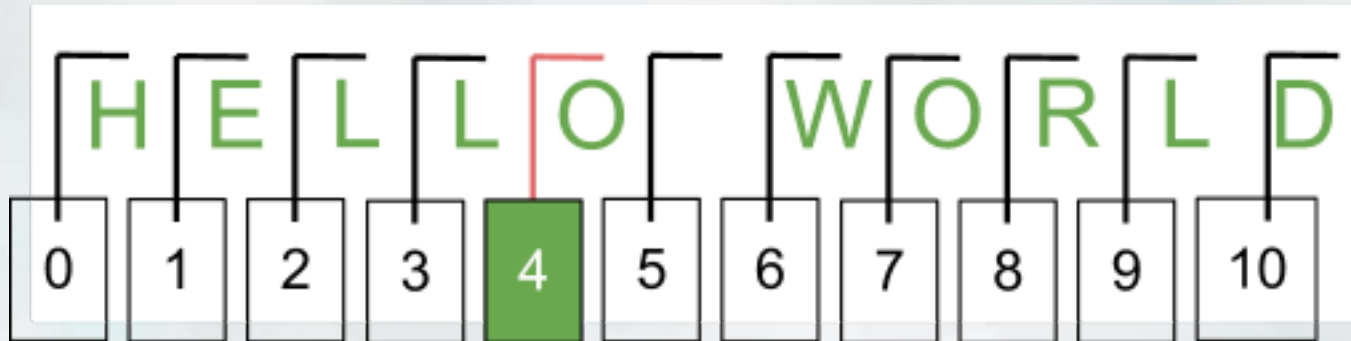


# Anatomy of a String



Index into strings using bracket notation

# Anatomy of a String

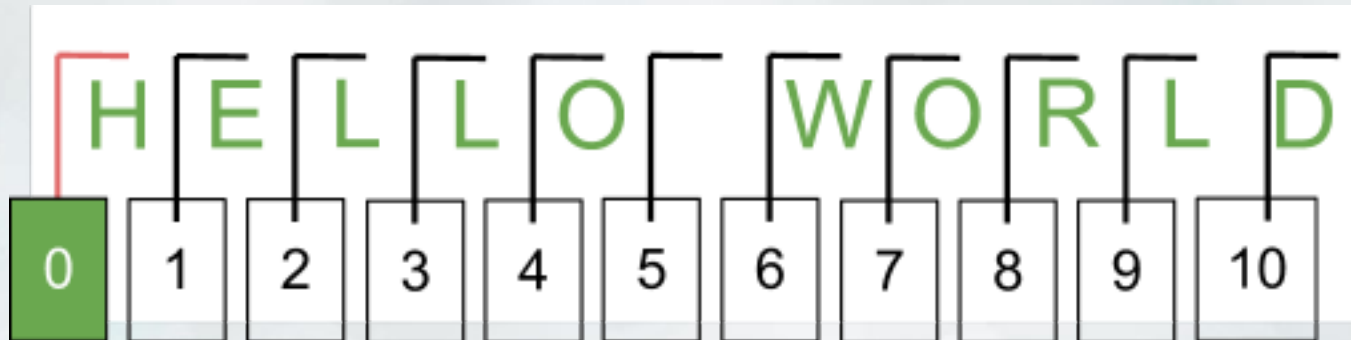


Index into strings using bracket notation

```
>>> "HELLO WORLD"[4]
```

```
'O'
```

# Anatomy of a String

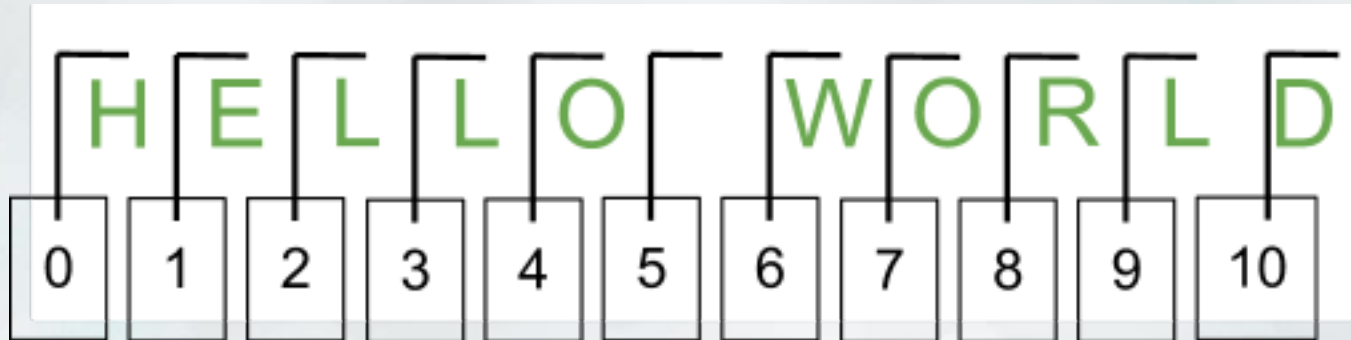


Index into strings using bracket notation

```
>>> "HELLO WORLD"[0]
```

```
'H'
```

# Anatomy of a String



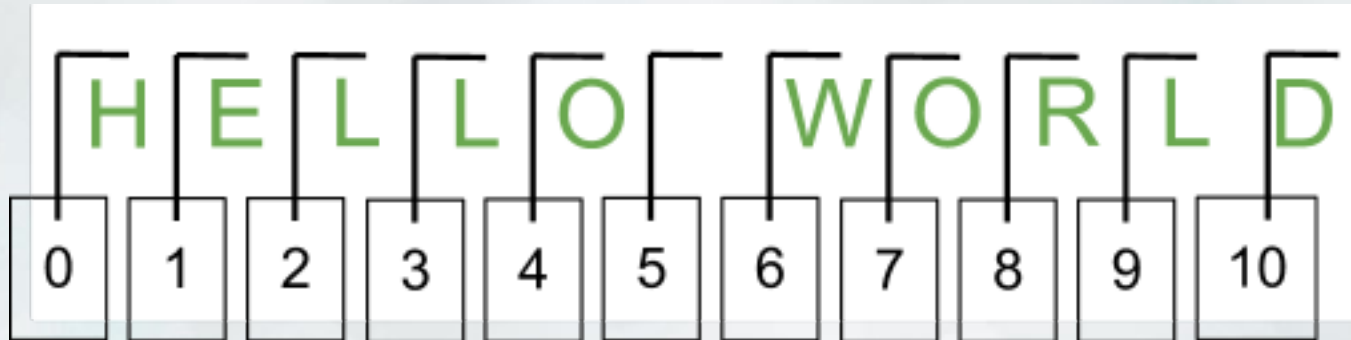
Index into strings using bracket notation

```
>>> "HELLO WORLD"[20]
```

```
???
```

What happens here?

# Anatomy of a String

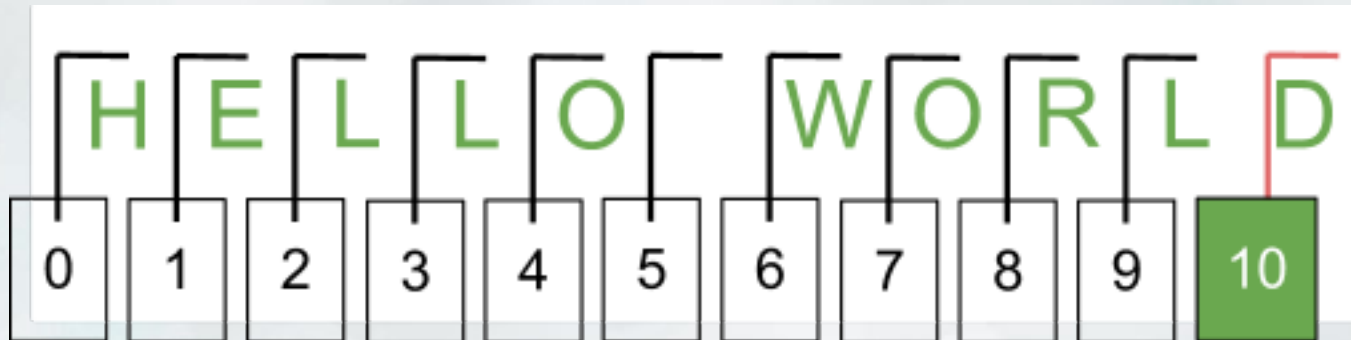


Index into strings using bracket notation

```
>>> "HELLO WORLD"[len("HELLO WORLD")]  
???
```

What about this?

# Anatomy of a String

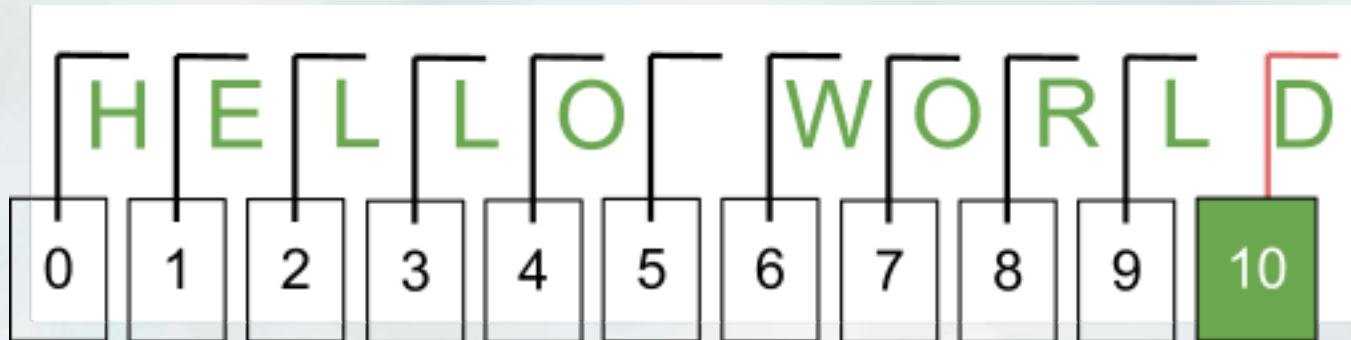


Index into strings using bracket notation

```
>>> "HELLO WORLD"[len("HELLO WORLD") - 1]  
'D'
```

The last character of a string is NOT the length of the string!

# Anatomy of a String



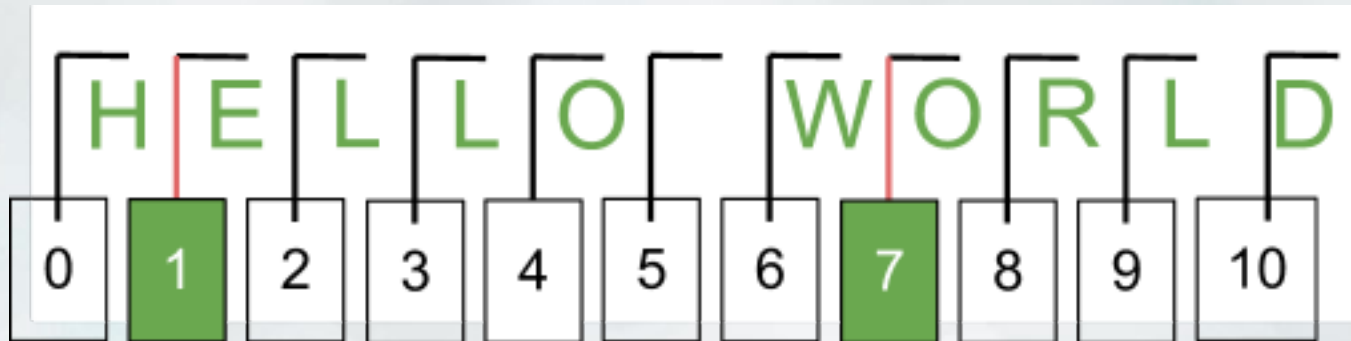
Index into strings using bracket notation

```
>>> "HELLO WORLD"[-1]
```

```
'D'
```

Here's a shortcut

# Anatomy of a String



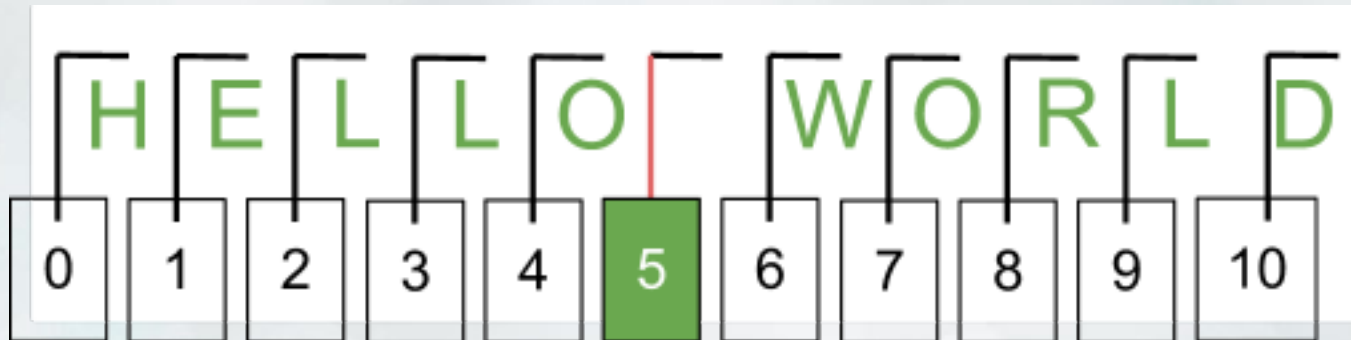
Get substrings using bracket notation

```
>>> "HELLO WORLD"[1:7]
```

```
'ELLO W'
```



# Anatomy of a String



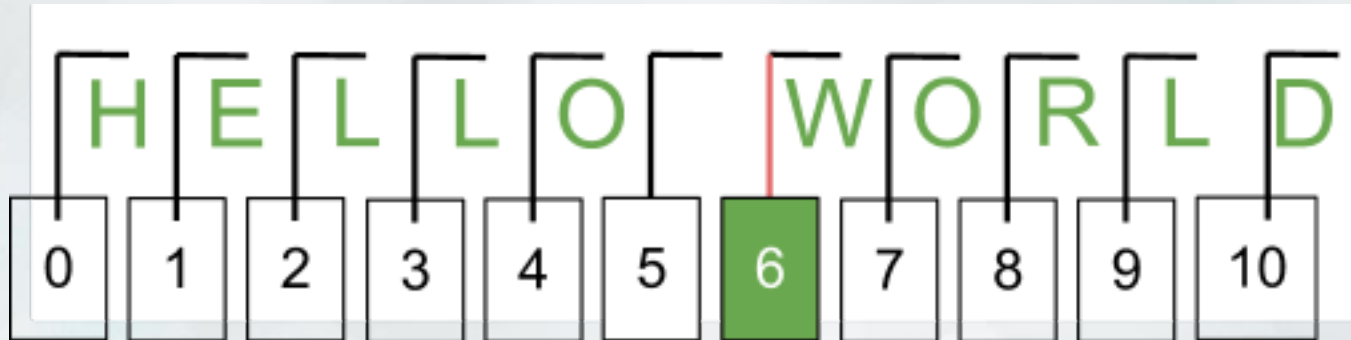
Get substrings using bracket notation

```
>>> "HELLO WORLD"[:5]
```

```
'HELLO'
```

If you leave off an index, Python goes to the beginning / end

# Anatomy of a String

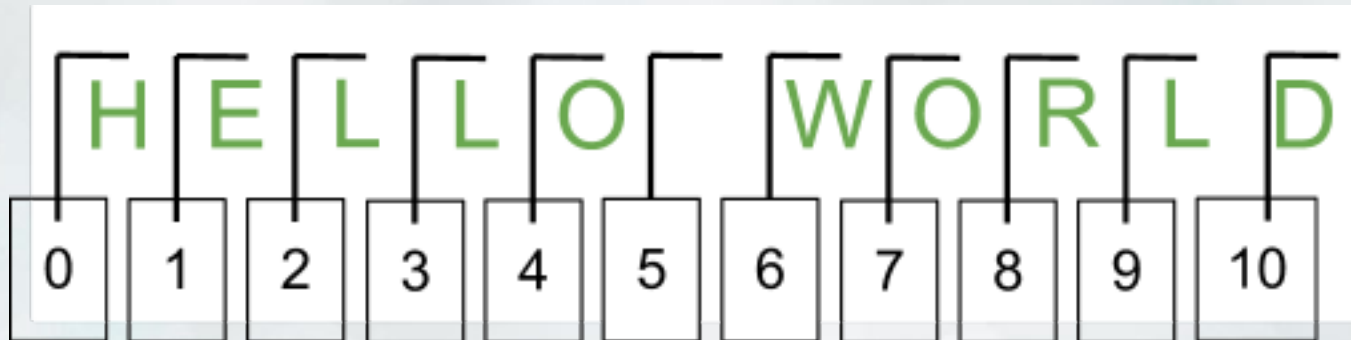


Get substrings using bracket notation

```
>>> "HELLO WORLD"[6:]  
'WORLD'
```

If you leave off an index, Python goes to the beginning / end

# Anatomy of a String

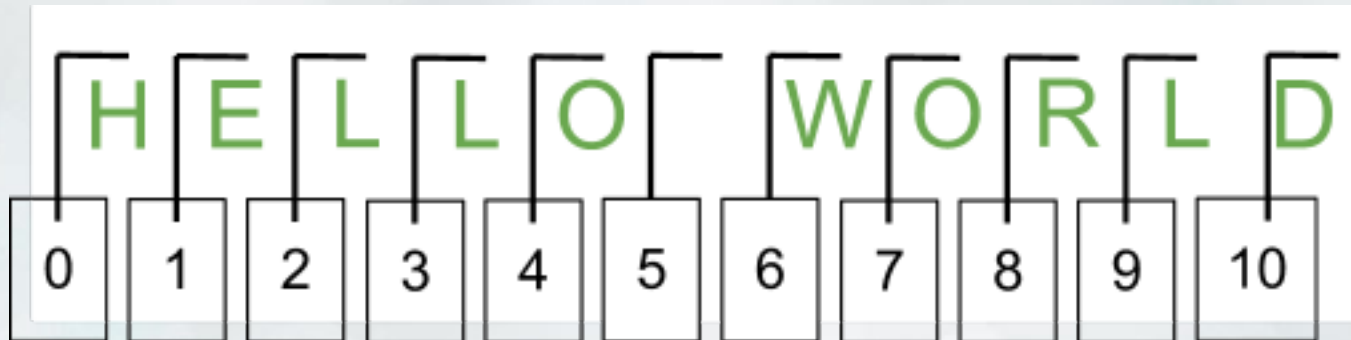


Get substrings using bracket notation

```
>>> "HELLO WORLD"[:]  
'HELLO WORLD'
```

If you leave off an index, Python goes to the beginning / end

# Anatomy of a String

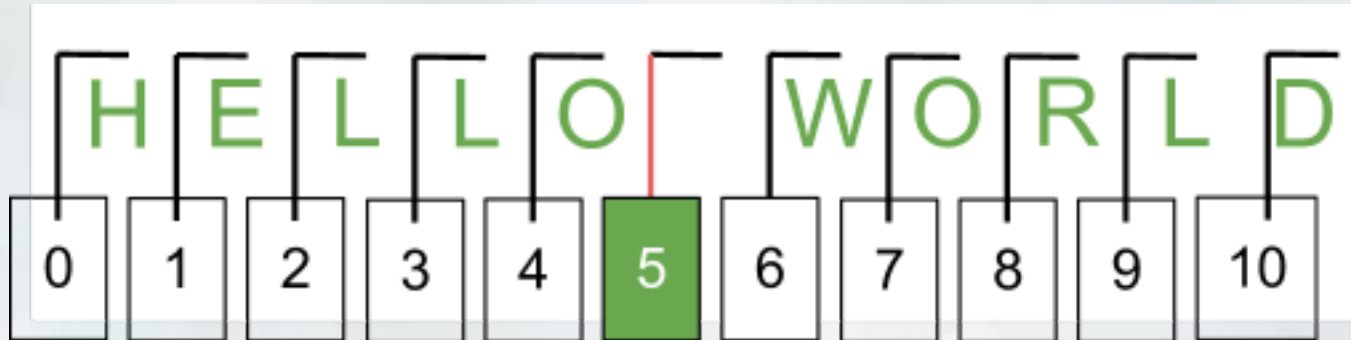


Get substrings using bracket notation

```
>>> "HELLO WORLD"[5:5]
```

```
???
```

# Anatomy of a String

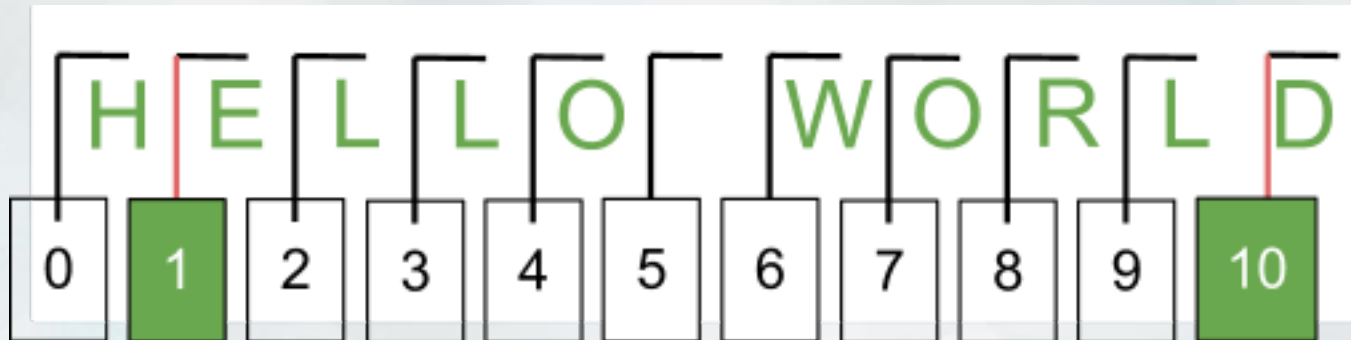


Get substrings using bracket notation

```
>>> "HELLO WORLD"[5:5]
```

```
"
```

# Anatomy of a String

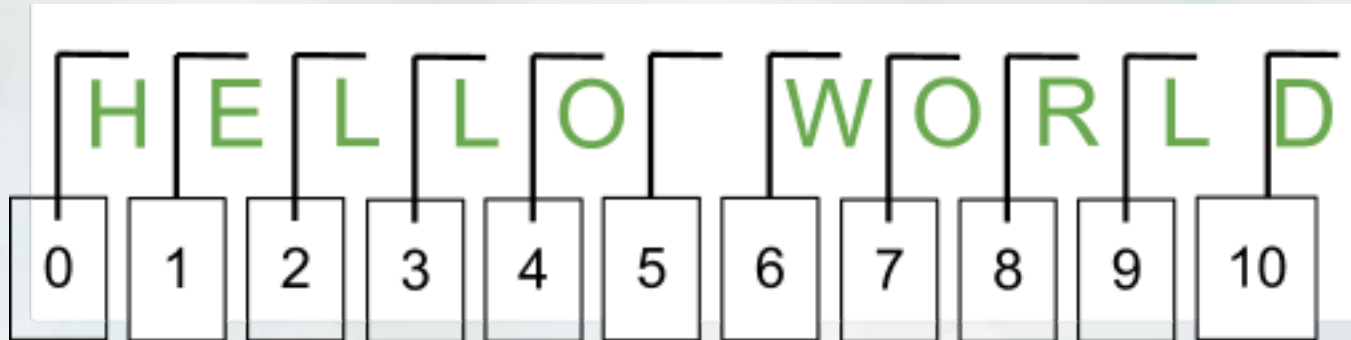


Get substrings using bracket notation

```
>>> "HELLO WORLD"[1:10:2]  
'EL OL'
```

You can even tell Python to skip characters

# Anatomy of a String

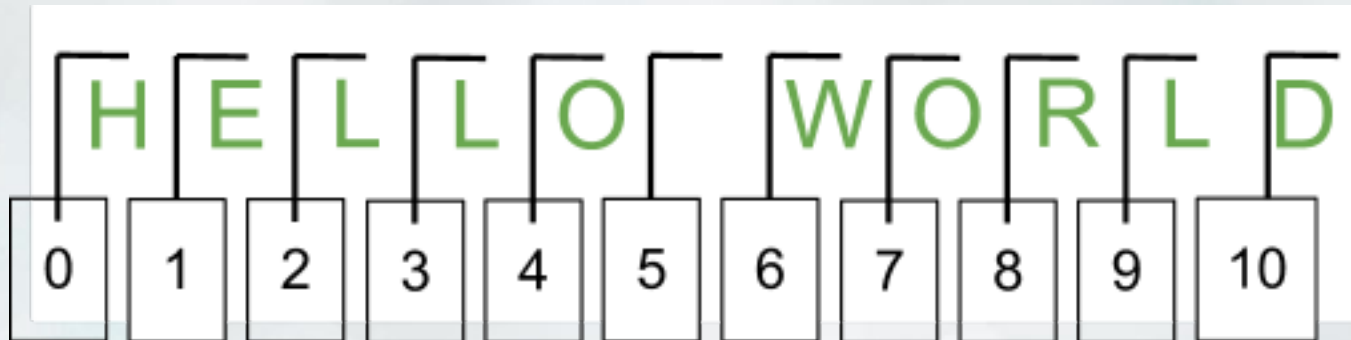


Get substrings using bracket notation

```
>>> "HELLO WORLD"[::5]  
'H D'
```

You can even tell Python to skip characters

# Anatomy of a String



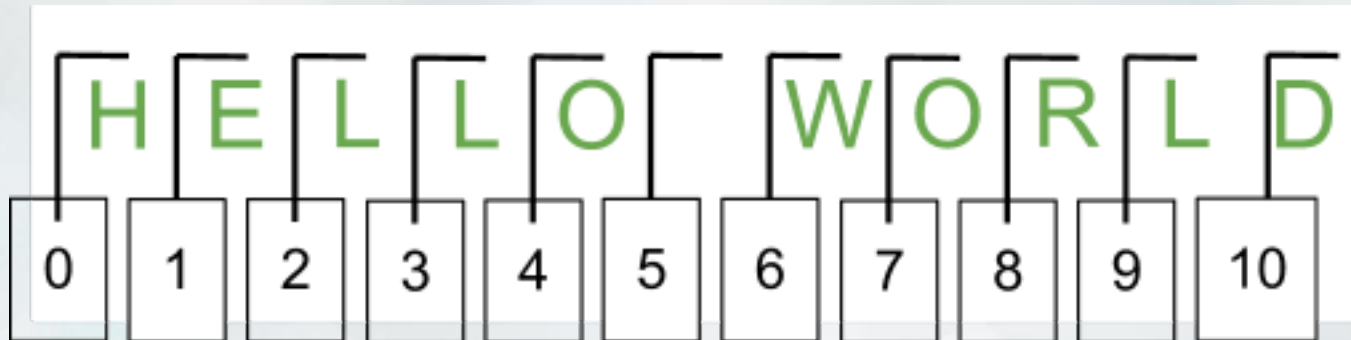
Get substrings using bracket notation

```
>>> "HELLO WORLD"[::-1]
```

```
???
```



# Anatomy of a String



Get substrings using bracket notation

```
>>> "HELLO WORLD"[::-1]  
'DLROW OLLEH'
```

# String Things

- String Indexing
  - `s[i]`
  - Return the character in string `s` at position `i`
  - Start counting from zero!
- You can index with negative numbers too
  - `s[-i]`
  - Return the `i`th character from the right
  - Start counting from one!
- Why isn't Python consistent?

# String Things

- String slicing
  - `s[i:j]`
  - Return a subset of characters in `s`
  - Starting at character `i`,
  - Up to (but not including) character `j`
- What happens if  $i > j$ ?
- If you leave off an index, defaults to beginning / end
  - `s[i : ]` - all characters from character `i` onward
  - `s[: i]` - all characters up to (but not including) character `i`

# String Things

- String slicing with skips
  - `s[i:j:k]`
  - Start at character `i`
  - Count up by `k`...
  - Stop before character `j`
- You can skip backwards too!
  - What are Python's rules?

# String Things

- Skipping backwards
  - `s[i:j:-k]`
  - Start at character `i`
  - Count down by `k`
  - Stop before character `j`
  
- What if `i < j`?