03PythonStrings

February 1, 2022

1 Culture and Coding: Python - Unit 1 Strings

The following 'book' and 'slides' is actually a jupyter notebook, which allows for Markdown and Python side by side. You may be reading this in the exported PDF form.

The notebook file is available on the CS0: Culture and Coding Github.

1.1 Words

- Have you really thought about the words you use?
  - What conteext surrounds those words?
  - Romeo and Juliet
    - What meaning does that bring?
    - Does it have a connectation to you?
  - Famous Star Trek TNG episode:
    - And the walls fell

1.2 Game Time!

- Play Hang'um.
- Try to use words (or phrases) from python or related to programming
- If the word has a space, make sure to include that!
- Example:

  ```
  H e l l o   W o r l d
  - - - - - - - - - - -
  ```

1.2.1 Next Step

- Take the word you have
- Put numbers below the dashes
  - Start counting at 0

```
G  u  i  d  o     v  a  n     R  o  s  s  u  m
-  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15
```

This is a **string** * Strings are characters in specified order * Often called an "order sequence of characters" * We do it all the time (crosswords, hang'em, wordly, etc) * But we don't think of it that way

A string has * an ordered sequence of characters * functions associated with it * indexes for each location

Given the indices 0-15: **what can we tell about the string?** * The first character is G at location 0 * The number of characters is 16 * Since we start counting at 0, the length is always the last index +1 * Similiarly, the last character is at length-1 (location 15) * The middle of the string is length / 2 * In this case, it is better to round down (truncate) if the string is odd in length

Lets look at code!

> Shoutout:
> Guido van Rossum is the creator of the python programming language

```
[ ]: pythonCreator = "Guido van Rossum"
     lastOnly = "Van Rossum" # Dutch naming conventions according to Guido
     homepage = "https://gvanrossum.github.io/"

     print(pythonCreator)
     print(lastOnly)
     print(homepage)

     # let's concatinate the strings using +

     print(pythonCreator + "'s homepage is " + homepage)
```

```
Guido van Rossum
Van Rossum
https://gvanrossum.github.io/
Guido van Rossum's homepage is https://gvanrossum.github.io/
```

> **Practice TASK**
> Reading the slides as a notebook/interactively? You should change the variables above to be someone else's name, and their homepage. Hit the run arrow to see how it changes the output.

2 Functions for Strings

There are a number of functions for strings. Let's look at two for now.

- `str(val)`
  - Takes any number and converts it to be a string. Called 'casting'.
  - You have used this when adding a number to a string for printing.
- `len(str)`
  - Gives you the length of any sequence
  - A string is a sequence of characters, so gives you length of the string!

```
[ ]: age = 50
     burgler = "Bilbo Baggins"
     combo = burgler + "'s age is " + age # fix this code by putting str(age) in␣
      ↪place of age
```

```
print(combo)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_25436/3440036219.py in <module>
      1 age = 50
      2 burgler = "Bilbo Baggins"
----> 3 combo = burgler + "'s age is " + age # fix this code by putting str(age)
 ↪in place of age
      4 print(combo)

TypeError: can only concatenate str (not "int") to str
```

[ ]:
```
combo = burgler + "'s age is " + str(age) # fixed version
print(combo)
```

```
Bilbo Baggins's age is 50
```

Now let's use `len(string)`

[ ]:
```
print(len(burgler))
lengthOfCombo = len(combo)
print(lengthOfCombo)
```

```
13
25
```

2.1 Slicing Strings

Any sequence like a string, we can "slice" into smaller parts * Really - we create a smaller string, we never change the original! * The operator we use is square brackets [] after the variable name * Inside the brakets is a number, or two numbers split by the colon : * Let's look at some examples:

[ ]:
```
print(burgler) # set above to be Bilbo Baggins
print(burgler[0]) # just give me the first character
print(burgler[6:13])
```

```
Bilbo Baggins
B
Baggins
```

2.1.1 Baggins!

What does `burgler[6:13]` do?

- It says, start at the 6th index

B	i	l	b	o		B	a	g	g	i	n	s
0	1	2	3	4	5	6	7	8	9	10	11	12

- Then give me the **substring** (part of string) until the second number
  - but the second number is 13?
  - correct I said *until* which does not include the second number
  - formally stated:
    - inclusive of the first value
    - exclusive of the second value
- If a value is omitted at the end, it grabs until end of string
  - `burgler[6:]` is the same as `burgler[6:13]`
  - the former is most common
- If a value is omitted at the start, 0 is assumed

```
[ ]: print(burgler[6:])
     print(burgler[:5])
```

```
Baggins
Bilbo
```

Also, if we want to use len(string) combined with this, it works well!

```
[ ]: length = len(burgler)
     half = int(length / 2) ## why would I want an int here? Do indices have␣
      ↪decimals or whole numbers?
     print(burgler[half:])
     print(burgler) # just a reminder with all this, the original remained
```

```
Baggins
Bilbo Baggins
```

We will come back to slicing. Moving foward we also want to make strings *look good*.

2.2 String Format

- Having to worry about string contination and types is a pain
- Furthermore, what if I wanted something to be more human readable
- Example: 33.33333333% isn't as nice as 33.33%
- Returning to Bilbo who really just wants to be left alone

```
[ ]: combo = burgler + "'s age is " + str(age)
     print(combo)

     # a better way, string format

     combo2  = f"{burgler}'s age is {age}"
     print(combo2)
```

```
Bilbo Baggins's age is 50
Bilbo Baggins's age is 50
```

Notice the **f** before the quotation mark!

- There are multiple ways to do string format
- You can also say `"{}'s age is {}".format(burgler, age)`
- The "f-string" notation does the same, but
  - Variables are directly in the string
  - It tells the computer to insert value in variable location!
  - Don't forget the curly brackets!
- This is also common in other languages like javascript

2.2.1 Formatting Numbers

- Often going from numbers to string, we *want* to make them look better.
- There are a bunch of different options for formatting in python
- Do not memorize them all!
- Look them up, just a few known (**.[precision]f** is most common)

Type	Description	Example	Output
s	String (default presentation type - can be omitted)	name = 'Aiden' print(f'{name:s}')	Aiden
d	Decimal (integer values only)	number = 4print(f'{number:d}')	4
b	Binary (integer values only)	number = 4print(f'{number:b}')	100
x, X	Hexadecimal in lowercase (x) and uppercase (X) (integer values only)	number = 255print(f'{number:x}')	ff
f	Fixed-point notation (6 places of precision)	number = 4 print(f'{number:f}')	4.000000
.[precision]f	Fixed-point notation (programmer-defined precision)	number = 4print(f'{number:.2f}')	4.00
0[precision]d	Leading 0 notation	number = 4 print(f'{number:03d}')	004

[ ]:
```
repeating  = 1 / 3
print(repeating)

## now let's make it look better
print(f"{repeating:.2f}")

myPercent = f"{repeating*100:.0f}%"

print(myPercent)
```

```
0.3333333333333333
0.33
33%
```

Notice!
We can do mathmatical operations inside the string. `repeating*100`, made it 33.3333, then `:.0f` made it 33

2.2.2 A program (thinking deeper / advanced concept)

- Let's combine what we learned this week
- Background:
  - HTML color schemes use hexidecimal format, with two character for Red, Green, Blue (in that order)
    - Additional the color code is always two digits long, so 0 should be listed as 00
    - And it starts with a pound / hash sign (#), so a common code for white is **#FFFFFF**
  - However, many color picker program provide Red, Green, Blue as numbers 0-255
    - White would be 255,255,255
- We can use string format to easily convert between the two!

[ ]:
```python
def toHtmlColor(red, green, blue):
    return f"#{red:02X}{green:02X}{blue:02X}"

red = int(input("Give me a red (0-255"))
green = int(input("Give me a green (0-255)"))
blue = int(input("Give me a blue (0-255"))

print(f"HTML Color is: {toHtmlColor(red, green, blue)}") #yes, we can do this!
```

```
HTML Color is: #FF0A7F
```

2.3 Recap

- Focus on strings being a sequence of characters
- Focus on splicing strings
- Know that string format makes your life easier, but not required!