CSCI-UA-0002

Intro to Computer Programming (No Prior Experience)

Module 7: Strings, Sequences, Slicing

Professor Emily Zhao

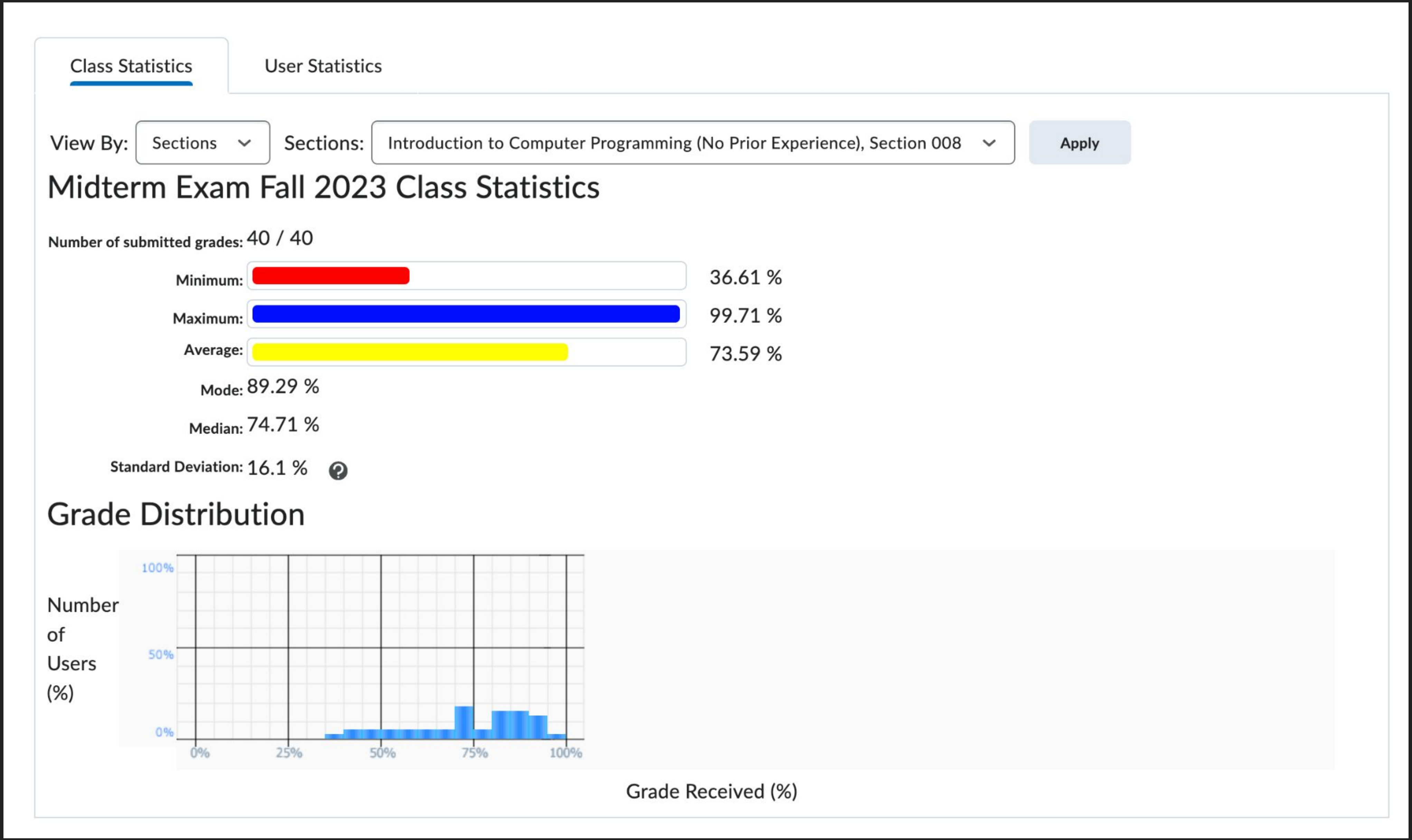
Section 008
T/R 12:30-1:45PM

Section 012
T/R 4:55-6:10PM

Agenda

- Discuss Midterm
- Midterm Reflection Extra Credit
- Go over Ed + Quiz Questions
- Practice Problems

Midterm



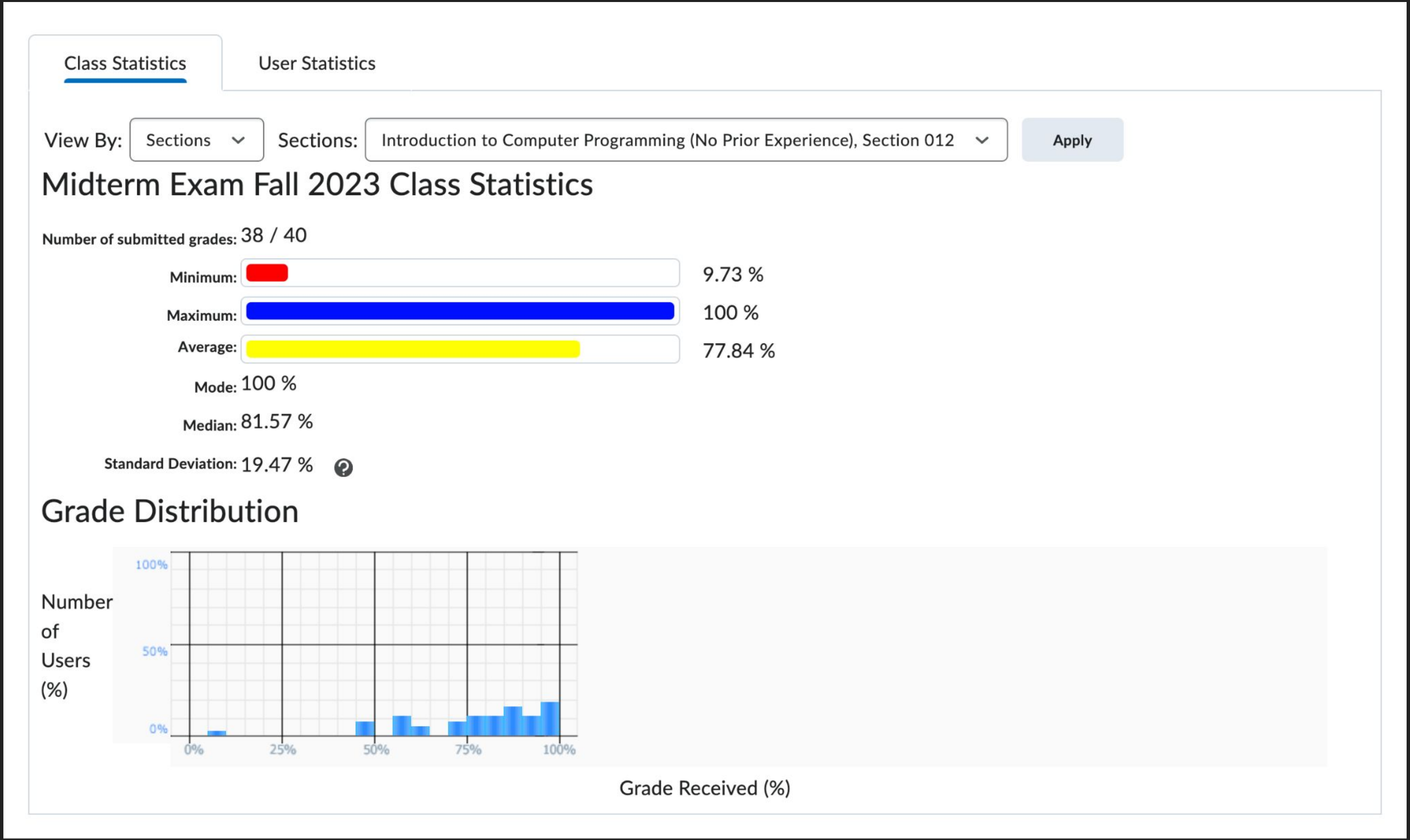
Class Statistics | User Statistics

View By: Sections | Sections: Introduction to Computer Programming (No Prior Experience), Section 008 | Apply

Midterm Exam Fall 2023 Class Statistics

Number of submitted grades:	40 / 40	
Minimum:	🟥	36.61 %
Maximum:	🟦	99.71 %
Average:	🟨	73.59 %
Mode:	89.29 %	
Median:	74.71 %	
Standard Deviation:	16.1 %	

Grade Distribution



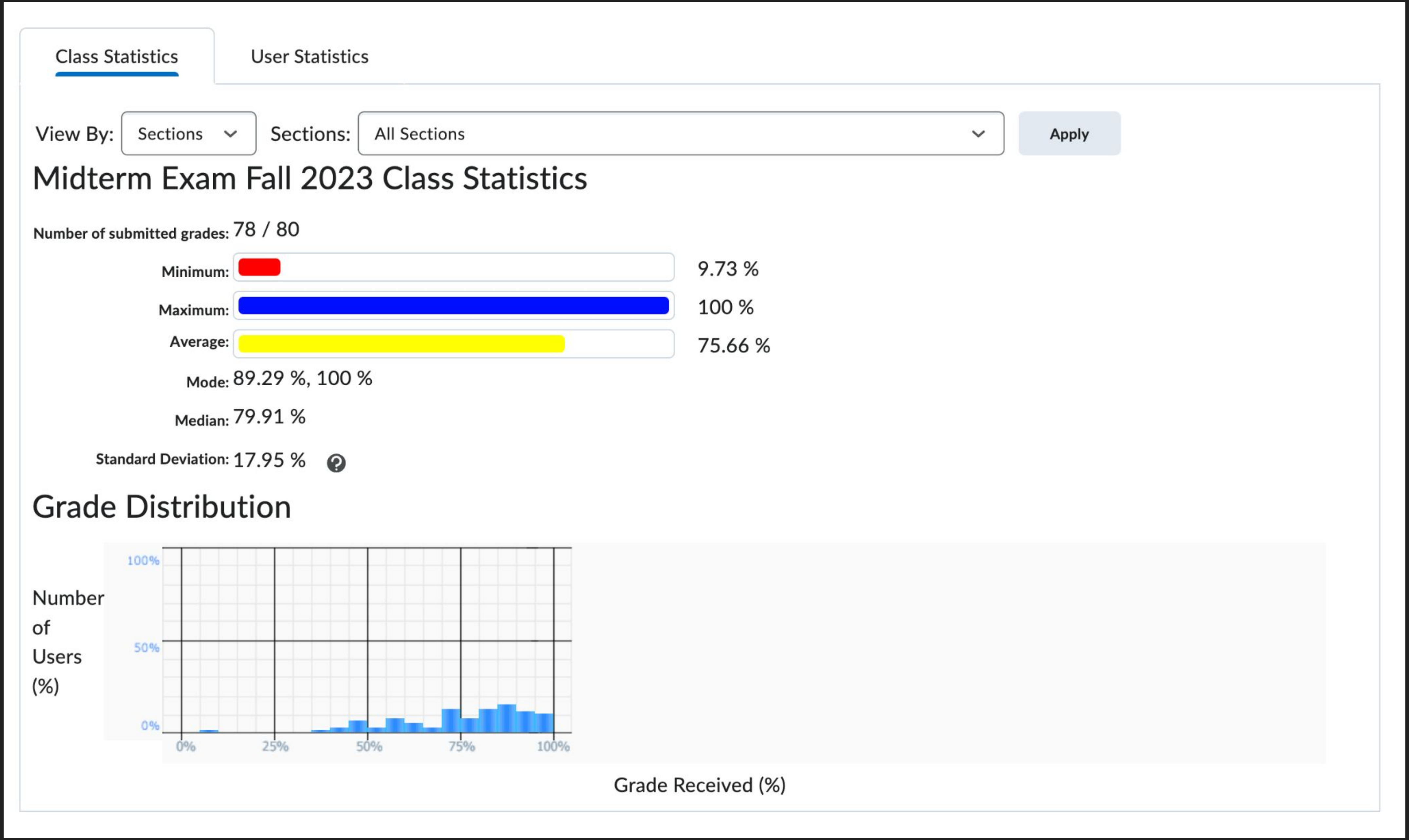
Class Statistics | User Statistics

View By: Sections Sections: Introduction to Computer Programming (No Prior Experience), Section 012 Apply

Midterm Exam Fall 2023 Class Statistics

Number of submitted grades:	38 / 40
Minimum:	9.73 %
Maximum:	100 %
Average:	77.84 %
Mode:	100 %
Median:	81.57 %
Standard Deviation:	19.47 %

Grade Distribution



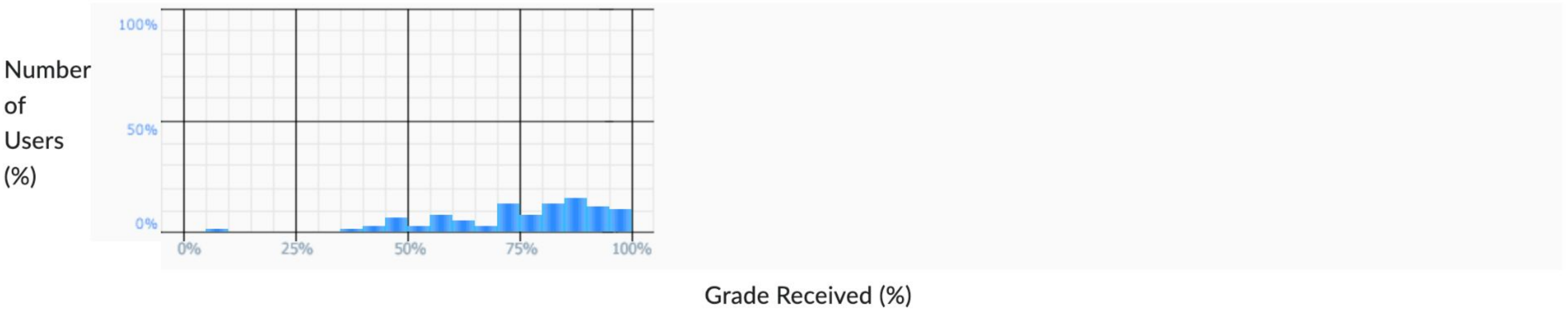
View By: Sections

Sections: All Sections

Midterm Exam Fall 2023 Class Statistics

Number of submitted grades:	78 / 80	
Minimum:	🟥	9.73 %
Maximum:	🟦	100 %
Average:	🟨	75.66 %
Mode:	89.29 %, 100 %	
Median:	79.91 %	
Standard Deviation:	17.95 %	

Grade Distribution



Midterm Curve

- To make sure that section 008 got an average score of 75%, I curved the exam by 1 point for everyone.
- That means the exam is now out of 27 points instead of 28.
- The new averages are now…

Section 008:	76%
Section 012:	80%
Both:	78%

Midterm Reflection (Extra Credit)

- Where did you struggle on the midterm?
  - What kind of questions did you miss?
  - What topics are still confusing to you?
- What do you need to do in order to prepare for the final?
- What can I do to help?
- What general confusions remain for you?
- How has your opinion/feeling about programming changed or remained the same since the beginning of class?
- Is there anything else you'd like to share with me about your experience in the class so far or anything that you think is important for me to know about you?

Book an office hour with me!

I won’t be in person this coming Thursday (11/1), but I have made 10-minute appointment slots on Zoom during class time for you to go over your midterm with me so you can write your reflection.

Module 07

Your Questions

→ What is negative string indexing? Why would I use it?
Can you use it in conjunction with positive string indexing?

→ How do you iterate over a string using a while loop?

→ Can you go over string slicing?

→ What is the difference between `isdigit()` and `isnumeric()`?

→ How does `str.find()` work?

What is a string again?

What is a string again?

— A data type
— Can be described as a “sequence of characters”
— Characters are arranged in a certain order

String indexing

```
word = "Python"
word[4] → o
Word[-2] → o
```

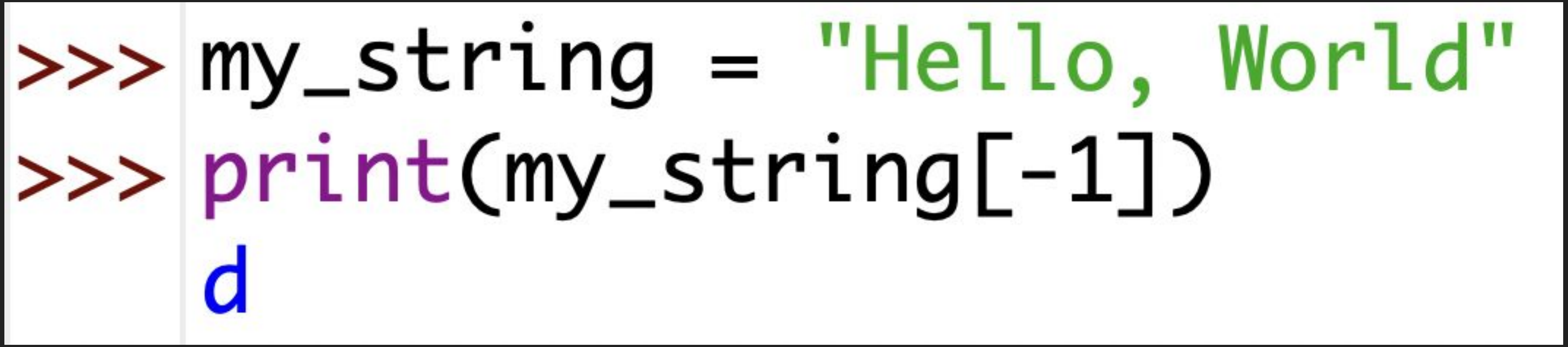
Forward direction indexing

0	1	2	3	4	5
P	y	t	h	o	n
-6	-5	-4	-3	-2	-1

Backward direction indexing

Backward String Indexing

- Avoids length calculation
- Easier to access characters at the end of a string
  - -1: last character
  - -2: second-to-last character

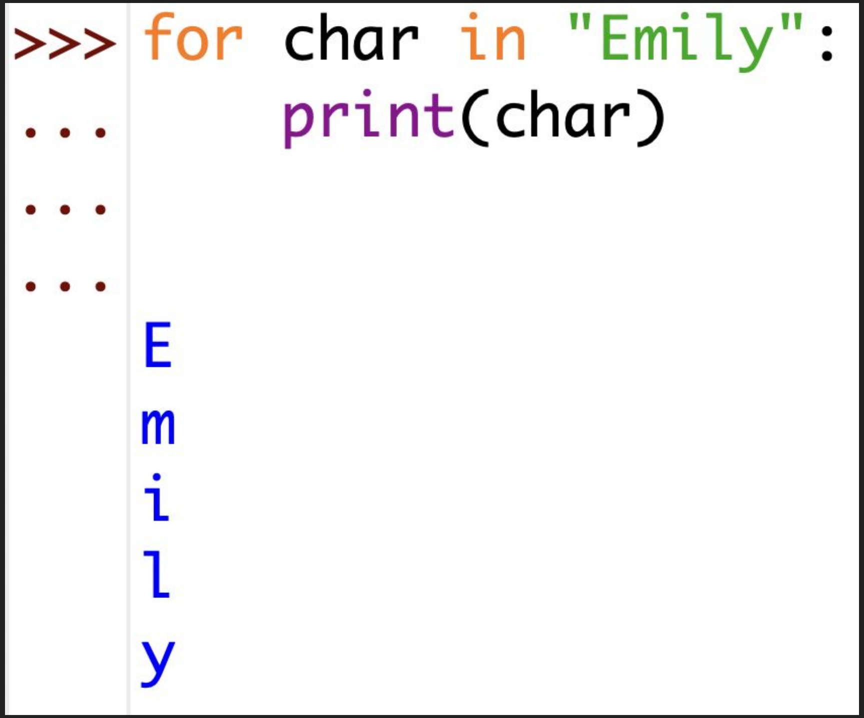


```
>>> my_string = "Hello, World"
>>> print(my_string[-1])
d
```

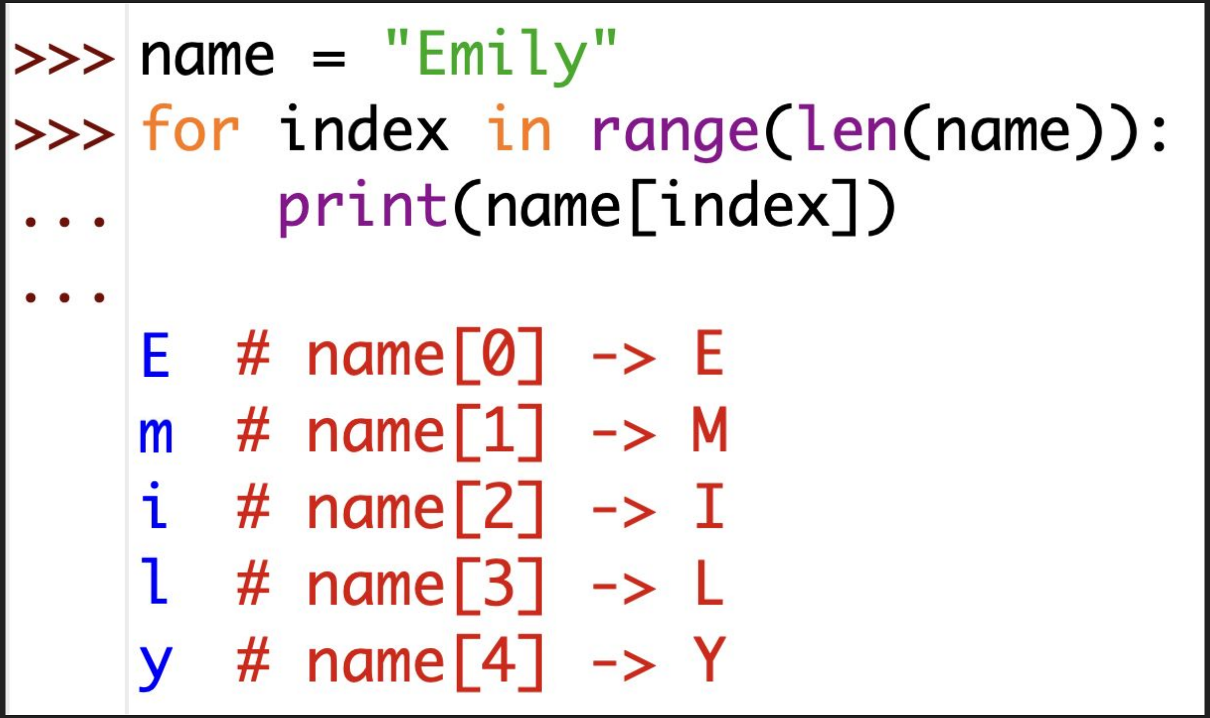
What are the three ways that I can iterate over a string?

```
name = "Emily"
>> E
>> M
>> I
>> L
>> Y
```

For Loops



```
>>> for char in "Emily":
...     print(char)
...
...
E
m
i
l
y
```



```
>>> name = "Emily"
>>> for index in range(len(name)):
...     print(name[index])
...
E  # name[0] -> E
m  # name[1] -> M
i  # name[2] -> I
l  # name[3] -> L
y  # name[4] -> Y
```

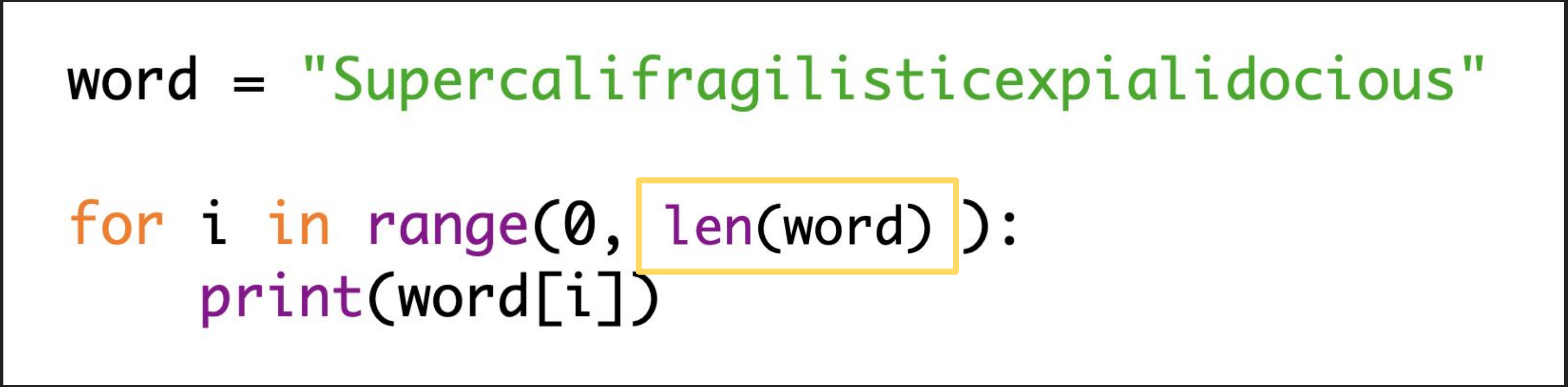
Control variable is each character

Control variable is the index

While Loop

```python
# When looking through a string with
# a While loop, you must use indexing

name = "Emily"
index = 0

# len(name) = 5

# name[0] -> E
# name[1] -> M
# name[2] -> I
# name[3] -> L
# name[4] -> Y

while index < len(name):
    print(name[index])
    index += 1
```

Using the length of the string to create a range:



```python
word = "Supercalifragilisticexpialidocious"

for i in range(0, len(word) ):
    print(word[i])
```

Programming Challenge

Write a function that counts the #'s of vowels in a string (A,E,I,O,U)

```
# name: countVowels
# input: a string
# processing: counts the number of vowels in a word
# output: the number of vowels
```

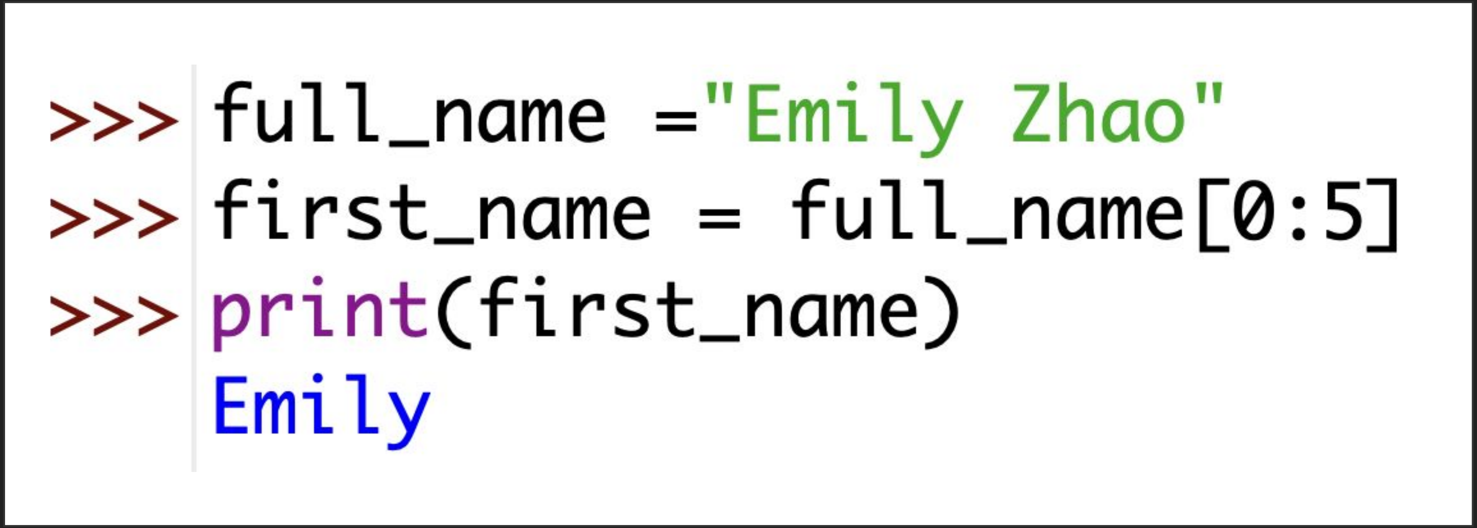
```
vowels = countVowels("Emily")
print(vowels) # 2
```

```python
def countVowels(word):
    vowelCount = 0
    for c in word.lower():
        if c == "a" or c == "e" or c == "i" or c == "o" or c == "u":
            vowelCount += 1
    return vowelCount
```

String slicing

Slicing a String

- Sometimes you may find it necessary to extract a portion of a string from another string.
- You can use “slicing” notation in Python to extract a span of characters from a string into a new string. We call this new String a "substring". For example:



```
>>> full_name ="Emily Zhao"
>>> first_name = full_name[0:5]
>>> print(first_name)
    Emily
```

Slicing a String

```
substring = bigstring[start:end:step]
```

- You must supply at least a start or an ending index value.
- Substrings contain all characters starting at the start value specified and continue up to (but do not include) the ending value.
- Omitting a starting or ending index value will cause Python to assume you want to start at the beginning of the string (if you omit a start value) or you want to continue slicing to the end of the string (if you omit the end value)
- **This should look a lot like the range function!**

String Slicing Notation

What will the following code print?

```
word = "Superman sings in the shower."
```

```
print (word[7])
print (word[0:8])
print (word[9:14])
print (word[:5])
print (word[9:])
print (word[-7:])
print (word[0:len(word):3])
print (word[30])
```

```
> n
> Superman
> sings
> Super
> sings in the shower.
> shower.
> Seasgit or
> IndexError: string index out of
range
```

Positive + Negative Indexing in Conjunction:

```python
string = "Hello, World!"

# Using both positive and negative indices for slicing
substring = string[7:-1]

print(substring)
```

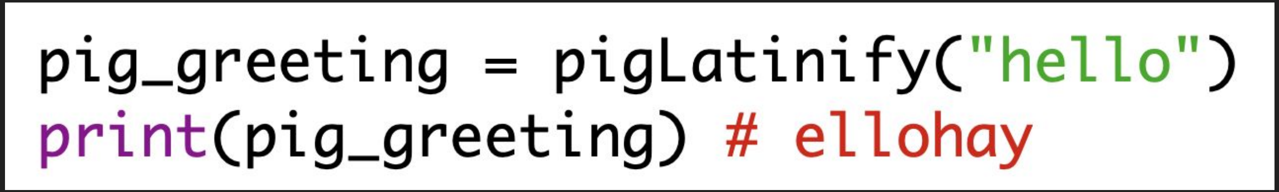
```
# > World
```

Programming Challenge: Pig Latin Translator

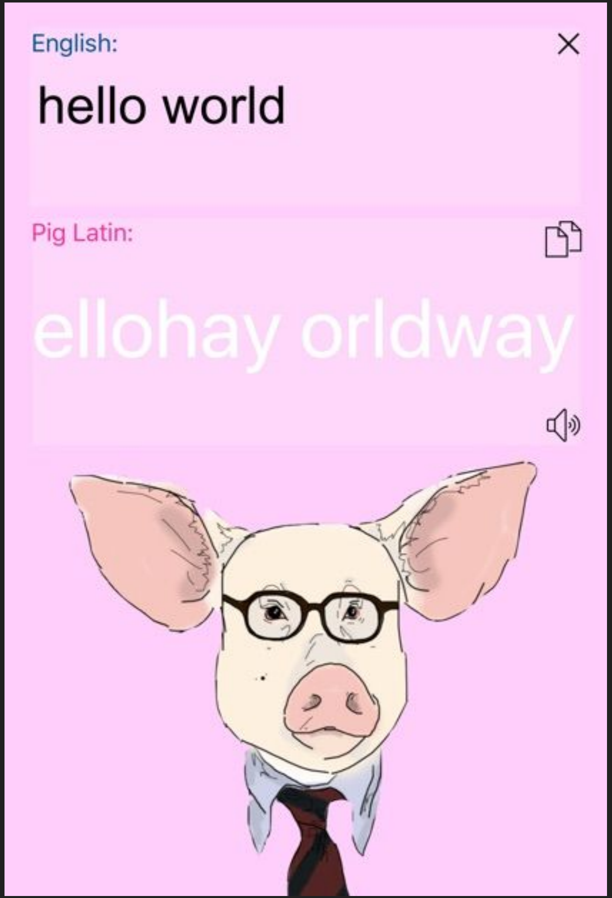
- Write a function that takes a word as an input and returns the pig latin version of that word.

A Pig Latin word can be generated using the following rules:

- Remove the first letter of the word
- Place the first letter of the word at the end of the word
- Add the string ”ay” to the end of the word



```
pig_greeting = pigLatinify("hello")
print(pig_greeting) # ellohay
```



English:

hello world

Pig Latin:

ellohay orldway

```python
def pigLatinify(word):
    # construct new word through concatenation and slicing
    pig_word = word[1:] + word[0] + "ay"
    return pig_word
```

Strings cannot be changed once they are created.

→ True or false?

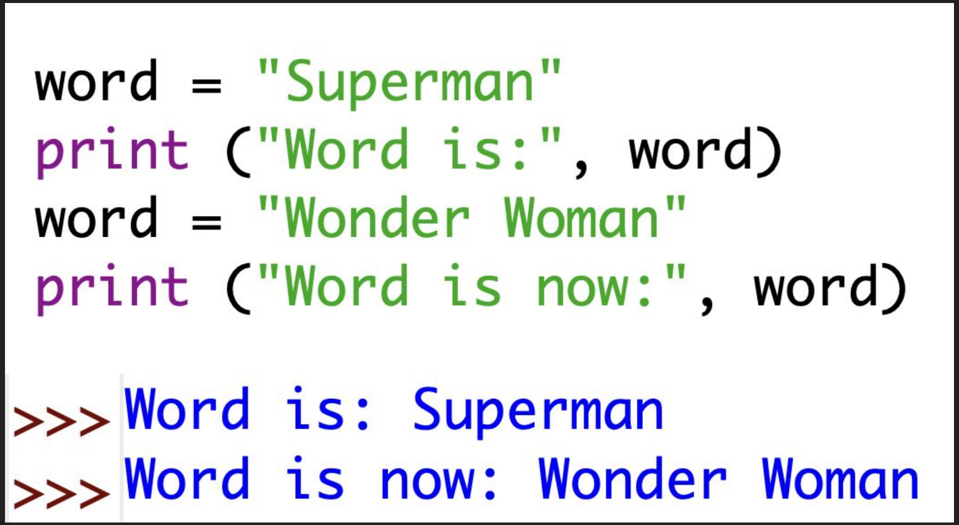
Strings cannot be changed once they are created.

→ True!

Strings are “Immutable”

- Strings are an **immutable** data type. This means that they cannot be changed once they are created.

- This may seem counterintuitive, since we have been doing the following since the beginning of the semester:



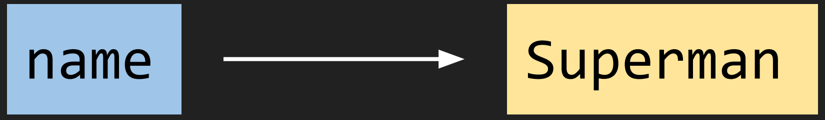
```
word = "Superman"
print ("Word is:", word)
word = "Wonder Woman"
print ("Word is now:", word)
```

```
>>> Word is: Superman
>>> Word is now: Wonder Woman
```

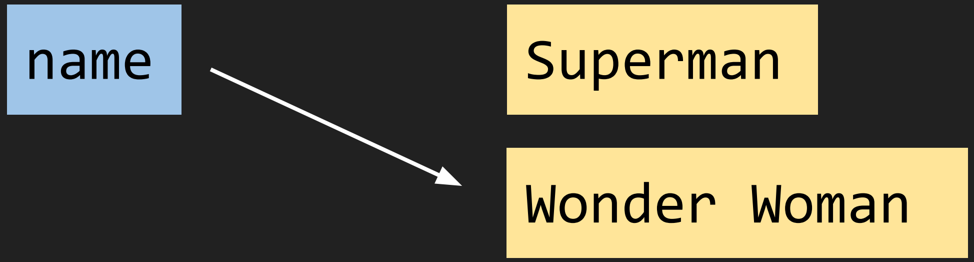
Strings are “Immutable”

What actually happens “under the hood” is that Python creates a separate string in your computer’s memory and “points” to that string instead of the original one.

```
name = 'Superman'
```

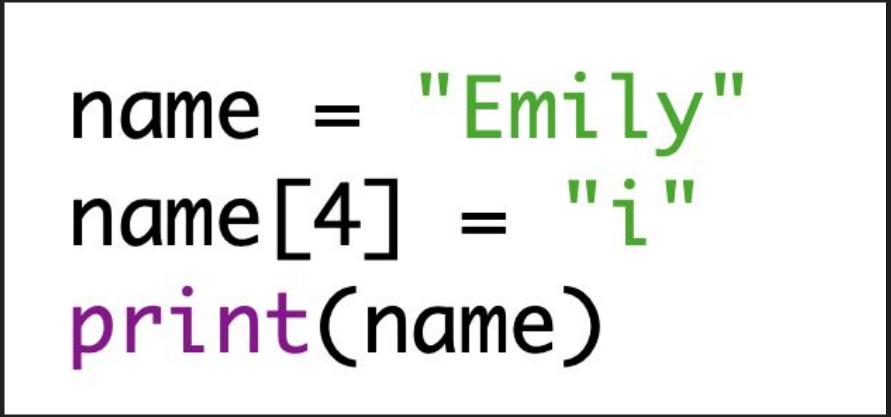


```
name = 'Wonder Woman'
```

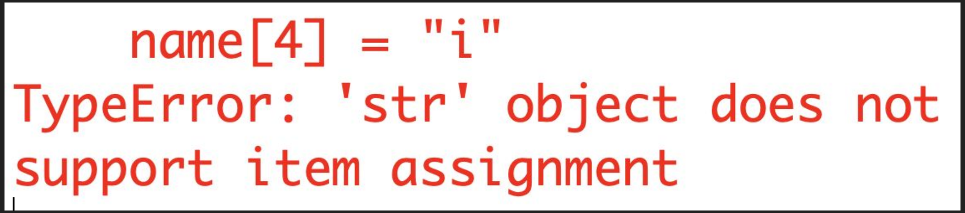


Strings are “Immutable”

– This means that you cannot change the individual characters within a string using index notation. You will raise an exception if you attempt to do so.



```
name = "Emily"
name[4] = "i"
print(name)
```



```
    name[4] = "i"
TypeError: 'str' object does not
support item assignment
```

How do you change a string then?

Programming Challenge

Write a function that replaces all vowels in a String with the underscore character (_)

`hello` → `h_ll_`

Will this work?

```
word = "hello"

for i in range(len(word)):
    # if the letter is a vowel, reassign it to _
    if word[i] in ["a", "e", "i", "o", "u"]:
        word[i] = "_"

print(word)
```

How do you change a string then? → Gotta make a new one!

```python
word = "hello"
new_word = ""

for i in range(len(word)):
    # if the letter is a vowel, add _ to new word
    if word[i] in ["a", "e", "i", "o", "u"]:
        new_word += "_"
    # if the letter is a consonant, just add the letter
    else:
        new_word += word[i]

print(new_word)
```

Testing Strings with in and not in

– The "in" operator is a Boolean operator that you can use to test to see if a substring exists inside of another string. Example:

```
word = "Grace Lily John Chris Tom"

if "Chris" in word:
    print ("Found him!”)
else:
    print ("Can't find Chris")

# > Found him!
```

– When you construct an expression with the “in” operator the result will evaluate to a Boolean

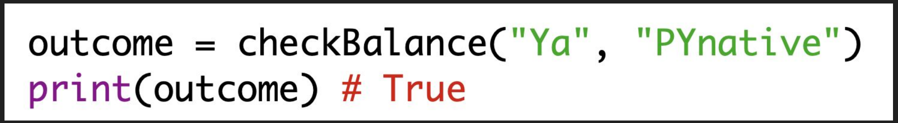
Programming Challenge: Balance Test

— Write a function to check if two strings are “balanced.”

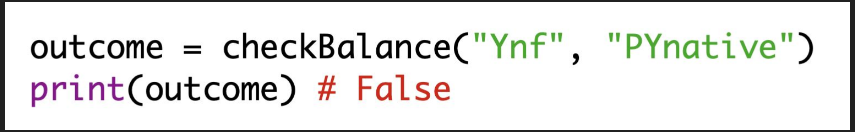
— For example, strings s1 and s2 are balanced if all the characters in the s1 are present in s2. The character’s position doesn’t matter.

```
Case 1:
s1 = "Ya"
s2 = "PYnative"
> True
```

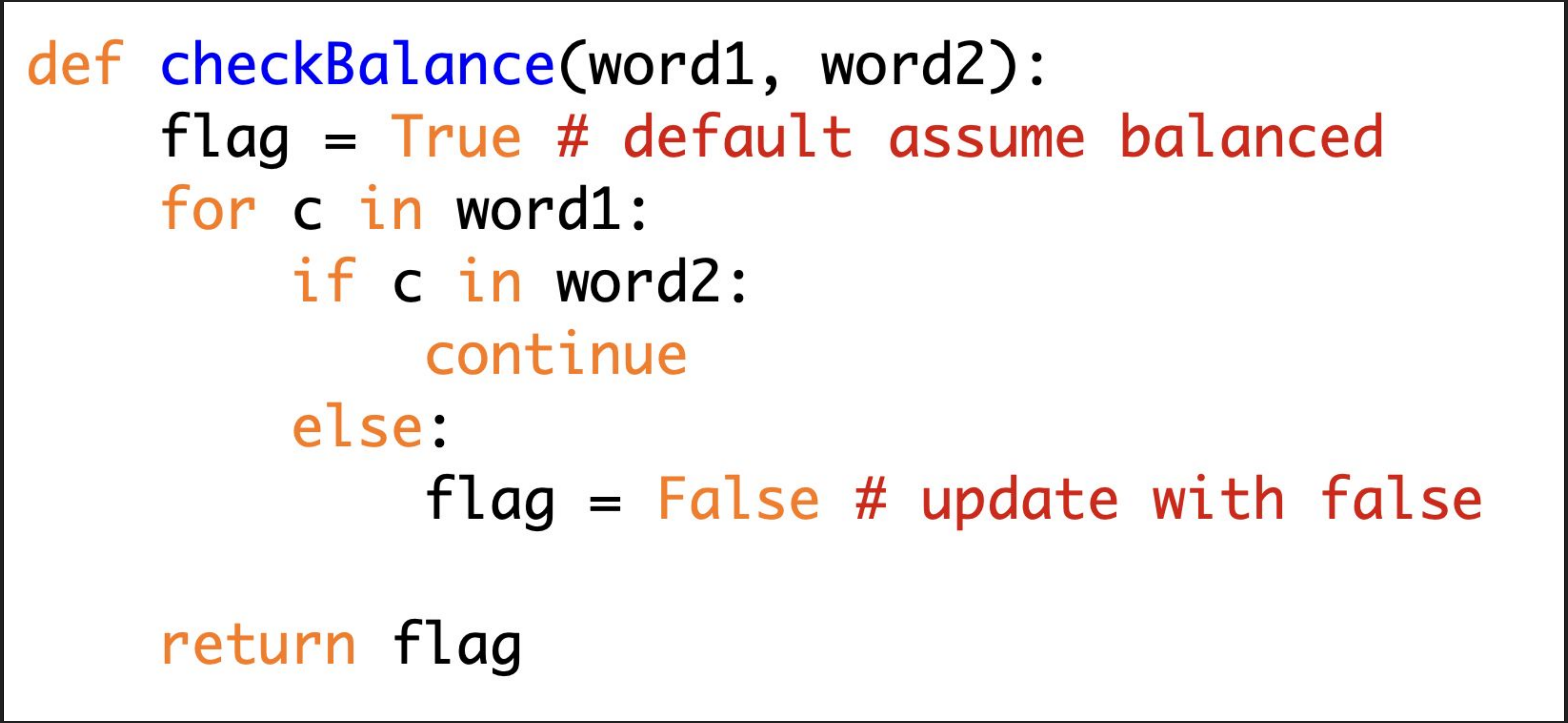
```
Case2:
s1 = "Ynf"
s2 = "PYnative"
> False
```



```python
outcome = checkBalance("Ya", "PYnative")
print(outcome) # True
```



```python
outcome = checkBalance("Ynf", "PYnative")
print(outcome) # False
```



```python
def checkBalance(word1, word2):
    flag = True # default assume balanced
    for c in word1:
        if c in word2:
            continue
        else:
            flag = False # update with false

    return flag
```

Programming Challenge: Palindrome Tester

– Write a function that returns whether a word is a palindrome (a word that reads the same backwards and forwards) or not.

```
result = isPalindrome("racecar")
print(result)
# > True
```

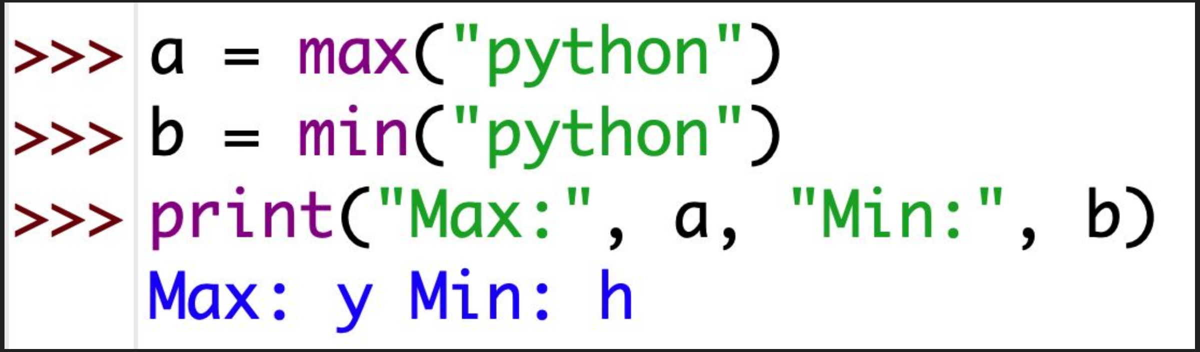
```python
def isPalindrome(word):
    word = word.lower() # make it case-insensitive
    backwards_word = ""

    # create backwards word
    for i in range(len(word)-1, -1, -1):
        backwards_word += word[i]

    if word == backwards_word:
        return True
    else:
        return False
```

```python
def isPalindrome(word):
    word = word.lower() # make it case-insensitive
    if word == word[::-1]:
        return True
    else:
        return False
```

String functions

Getting the largest and smallest character in a string

You can use two built in Python functions to obtain the maximum and minimum characters in a string (based on their ASCII codes)



```
>>> a = max("python")
>>> b = min("python")
>>> print("Max:", a, "Min:", b)
Max: y Min: h
```

String methods

```
stringvariable.method(arguments)
```

String methods

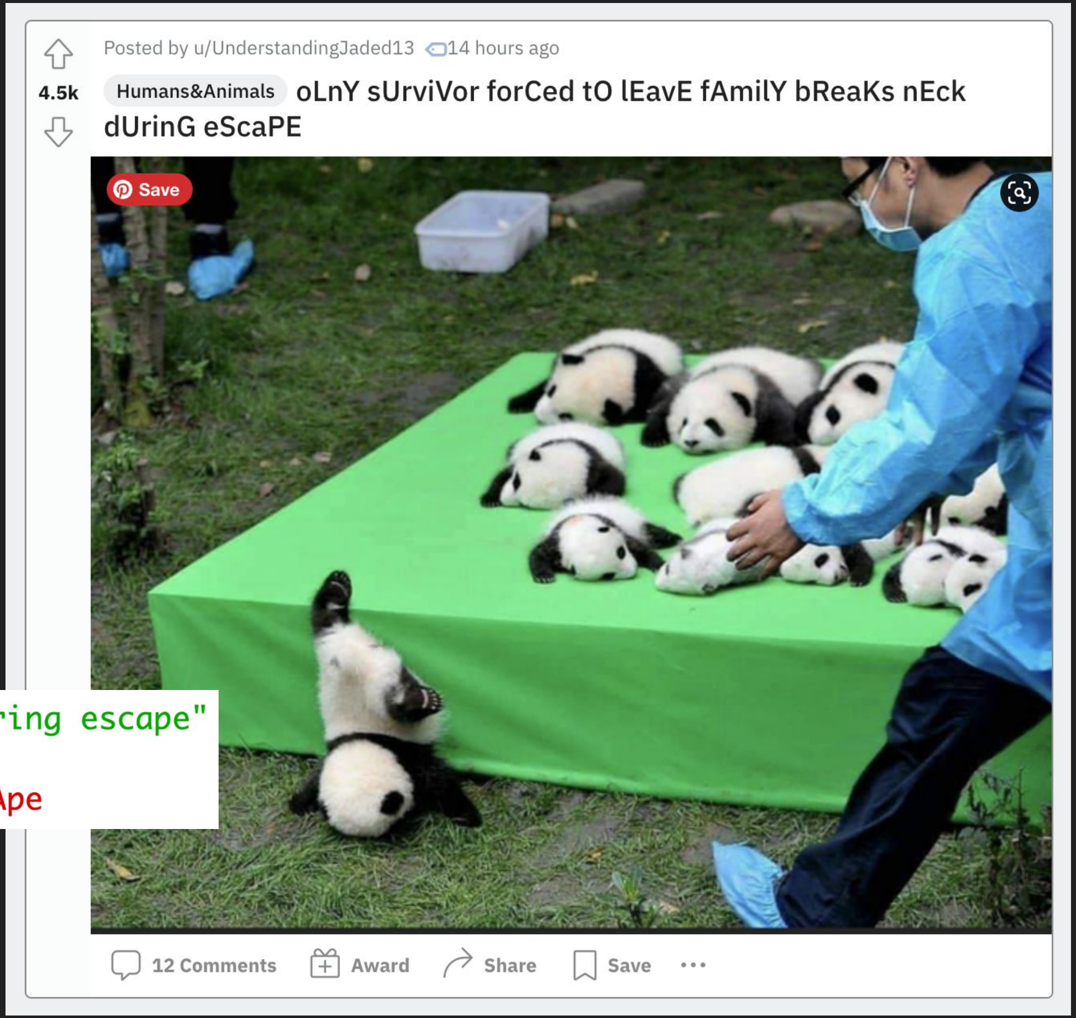
`.isalnum()`	True if all characters are alphanumeric
`.isalpha()`	True if all characters are alphabetic
`.islower()`	True is all alpha characters are lower
`.isspace()`	True if all characters are “whitespace”
`.isupper()`	True if all alpha characters are upper
`.isdigit()`	True if all characters in a string are digits from 0 to 9
`.isnumeric()`	True if all the characters in a string are numeric, not just digits (0-9), for example "½" or "١٢٣"

String modifications

`.lower()`	Returns a lowercase version of the string
`.upper()`	Returns an uppercase version of the string
`.rstrip()`	Removes whitespace at end of string
`.lstrip()`	Removes leading whitespace characters
`.capitalize()`	Returns a copy of the string with the first character capitalized
`.title()`	Returns a copy of the string with the first character of each word capitalized
`.swapcase()`	Returns a copy of the string where case is swapped among all alpha characters

Programming Challenge: Headline Generator

- There's a popular subreddit that likes to write the titles to their posts like such:
- Write a function that takes a string and returns a title that randomly alternates the casing of the text.


Posted by u/UnderstandingJaded13 14 hours ago

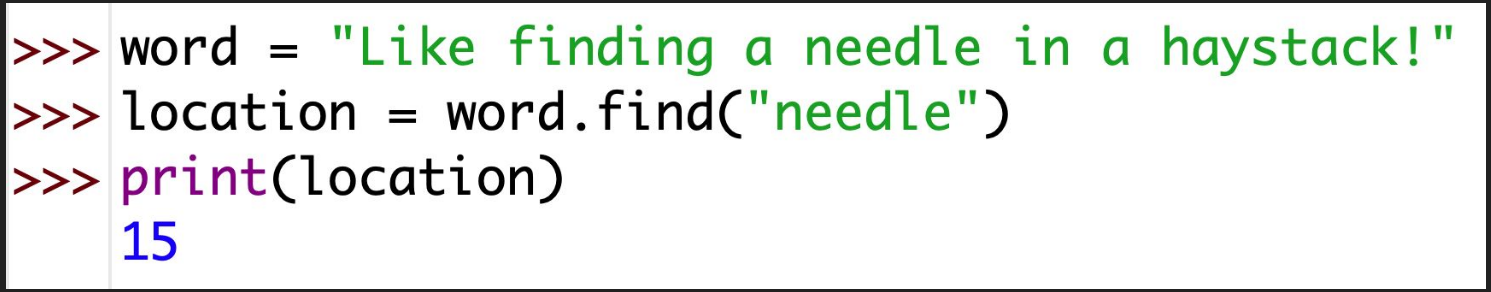
4.5k Humans&Animals oLnY sUrviVor forCed tO lEavE fAmilY bReaKs nEck dUrinG eScaPE

```
title = "only survivor forced to leave family breaks neck during escape"
print(headlineGen(title))
#oNLy SurViVor ForcEd tO LeavE FaMilY bReaKs NeCk DuriNG EscApe
```

```python
import random
title = "only survivor forced to leave family breaks neck during escape"

def headlineGen(title):
    new_title = ""

    for i in range(len(title)):
        num = random.randint(0,1)
        if num == 0:
            new_title += title[i].lower()
        else:
            new_title += title[i].upper()

    return new_title

print(headlineGen(title))
```

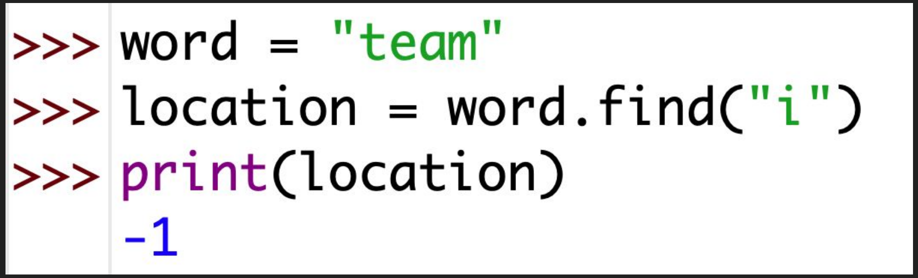
Finding substrings

You can find whether a string exists inside another string by using the `find()` method. Example:



```
>>> word = "Like finding a needle in a haystack!"
>>> location = word.find("needle")
>>> print(location)
    15
```

The `find()` method will return the index of the first occurrence of a substring within a string. If the `find()` method cannot find the desired substring it will return -1:

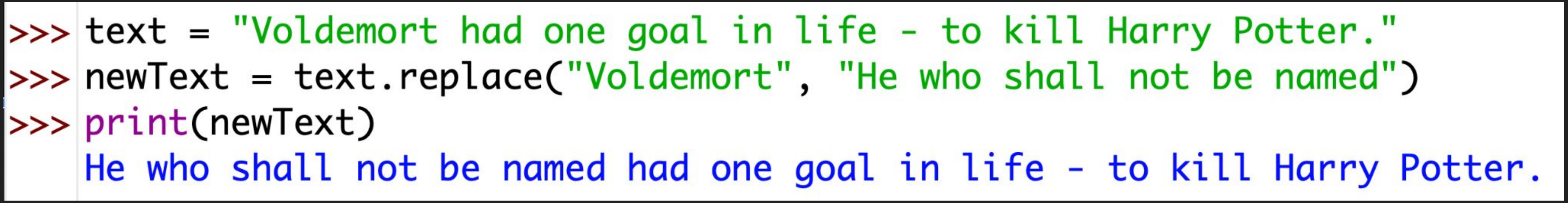


```
>>> word = "team"
>>> location = word.find("i")
>>> print(location)
    -1
```

Searching + Replacing

- Programs often need to perform search and replace functions on data, much like the “find and replace” functionality that exists in your word processor.

- You can have Python replace all occurrences of a substring by using the replace() method.



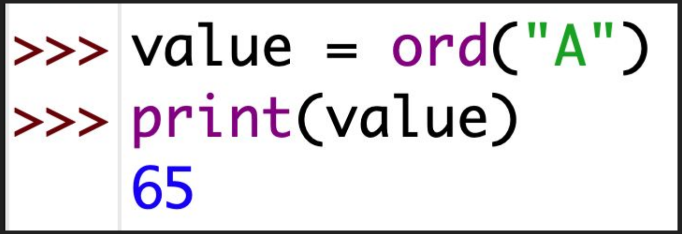
```
>>> text = "Voldemort had one goal in life - to kill Harry Potter."
>>> newText = text.replace("Voldemort", "He who shall not be named")
>>> print(newText)
    He who shall not be named had one goal in life - to kill Harry Potter.
```

Getting the ASCII value of a character

- Remember that Python (and all programming languages) use the standard ASCII encoding system to organize individual characters

- You can use the `ord()` function to look up the ASCII value of a character by doing the following:

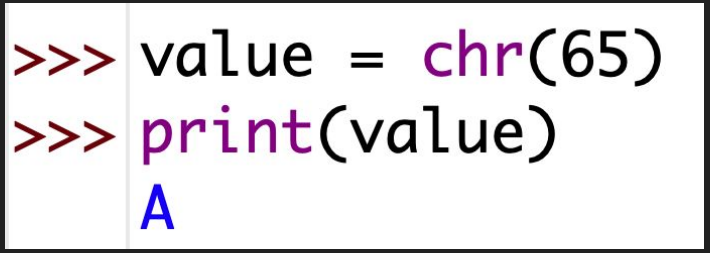
- The `ord()` function accepts one argument – a single character- and returns an integer that represents the ASCII value of that character



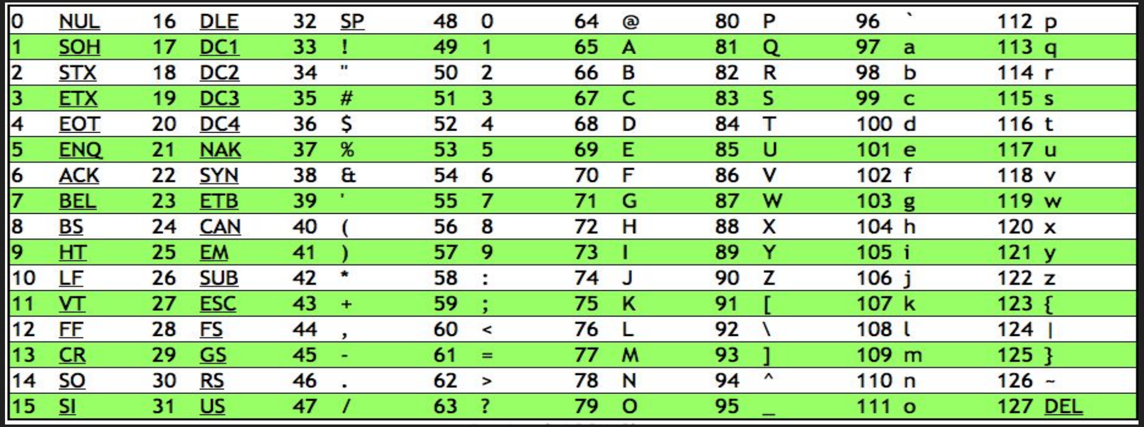
```
>>> value = ord("A")
>>> print(value)
    65
```

Getting the ASCII value of a character

You can also reverse the process and turn an integer into its equivalent letter value using the chr() function



```
>>> value = chr(65)
>>> print(value)
    A
```



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

Programming Challenge:

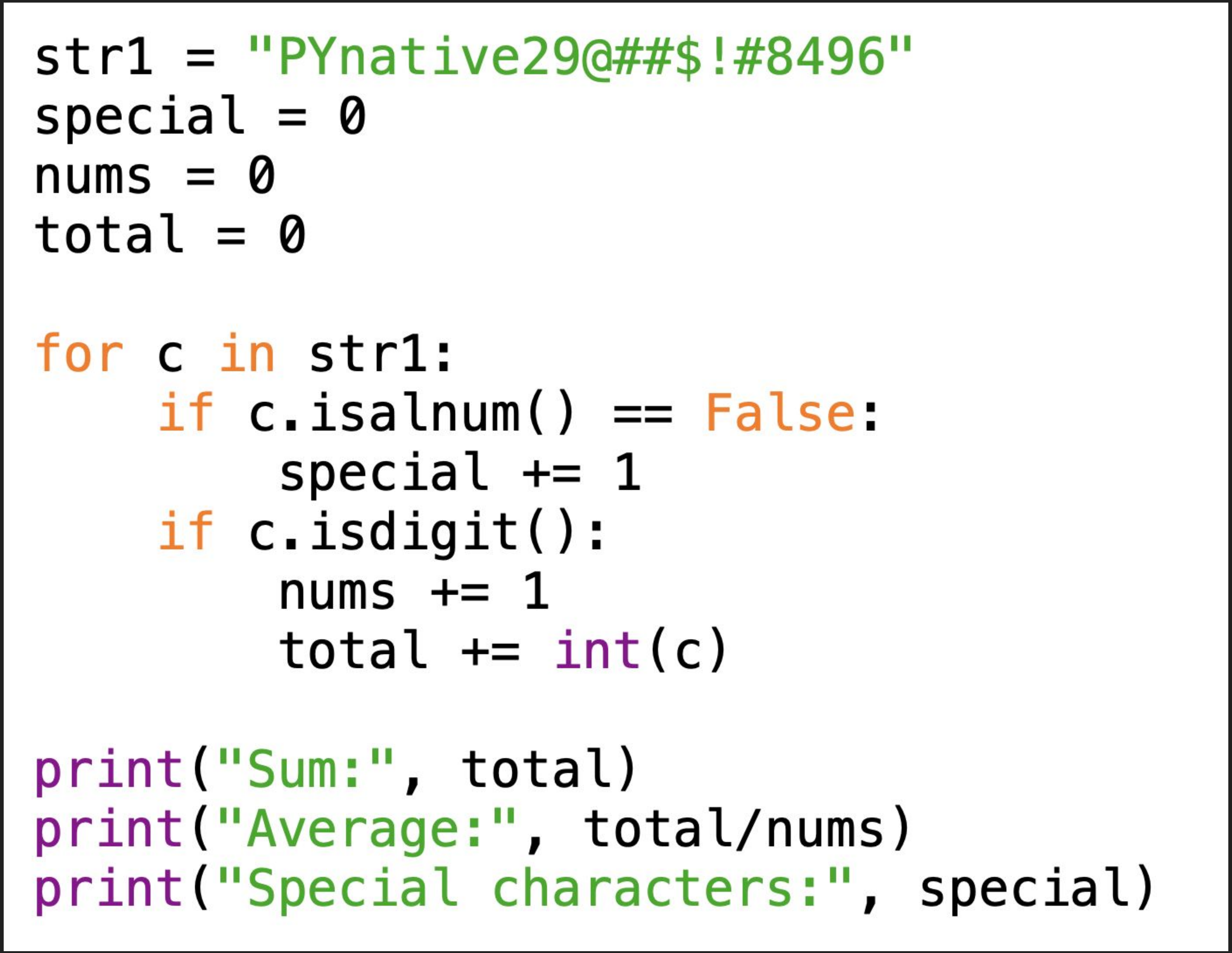
Calculate the sum and average of the digits present in a string. Also return how many special characters there are.

```
str1 = "PYnative29@##$!#8496"
```

```
> Sum: 38
```

```
> Average: 6.333333333333333
```

```
> Special character count: 6
```



```python
str1 = "PYnative29@##$!#8496"
special = 0
nums = 0
total = 0

for c in str1:
    if c.isalnum() == False:
        special += 1
    if c.isdigit():
        nums += 1
        total += int(c)

print("Sum:", total)
print("Average:", total/nums)
print("Special characters:", special)
```

Homework

- Self-Paced Learning Module #8 (due next Tues)
- Assignment #7 (due next Thurs)
- Midterm Reflection (due next Thurs)