Pi515

GIVING YOUTH THE {CODE} TO SUCCEED

JOHN DEERE

str = “Strings”

index, slice, methods

Learning Objectives: Strings

Initialize string variables

Access values from string variables

Manipulate string values with immutability

Initializing string variables

Double quotes	`doubleQuote = “string”`
Single quotes	`singleQuote = ‘string’`

Single quote strings are standard

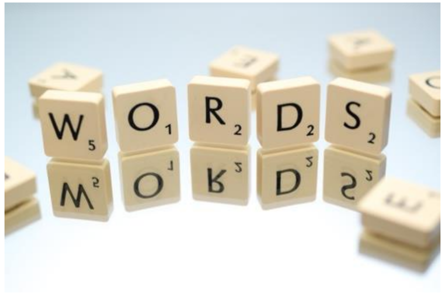
Exercise 1

Initialize 2 string variables:

One for your first name	firstName = ‘Ryan’
One for your last name	lastName = ‘McDaniel’

Sentences and Strings

What is a sentence made of?



What are words made of?

Sentences and Strings

● Composed of a set of symbols	alphabet, punctuation
● Also allows empty space	‘This is a sentence!’ or “enter” key - ‘\n’
● Order matters	‘ate’ vs. ‘tea’

Terminology

String

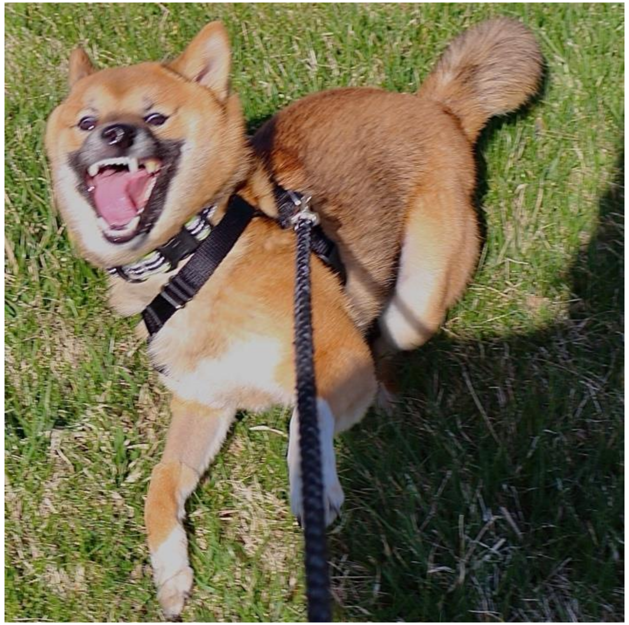
An ordered series of characters or symbols

Strings are the basis for computer science challenges like natural language processing (NLP) and computational genetics

ex. Siri, ChatGPT, DigitizeMyDocs (Ryan's old team)

Whiteboard Exercise

Whiteboard Exercise



Terminology

Index

A number (key), which provides access to a value within a string (lock).

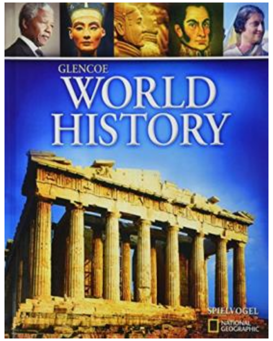
Indices (plural)

In computing, indices usually start at 0. **‘0th index’**

Real Life Indices

Addresses[**123 Street St**] =

Library[**123-4-56-789012-3**] =



Phones[**515-123-4567**] =

Individual String Facts

Have personal stats (like sportball!)

```
len('MyString')
```

And fun facts about themselves

```
's' in 'string'
```

GO SPORTS!
I'M SUPER WORKED-UP BECAUSE MY TEAM DID STUFF!
MOVE THE THING TO THE OTHER THING!

Individual String Facts

The len() function tells us the number of characters in a string

```
len(‘MyString’)
=> 8
```

The in keyword tells us if one string can fit inside the other

```
‘s’ in ‘string’
=> True

‘z’ in ‘string’
=> False
```

Terminology

Keyword

Words that are reserved solely for the purpose of special programming syntax within a certain language. Keywords cannot be used as identifiers like variable names.

ex. in = 'not allowed'

secretString Exercise

N O I C E

job

What if I only want a certain part of a string?

Terminology

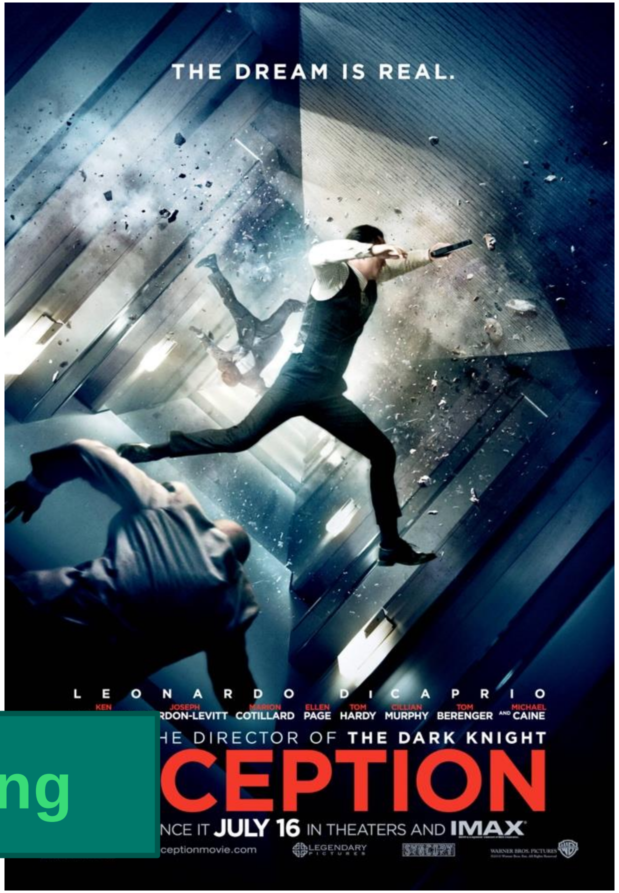
Substring

A string that exists within another string
ex. ‘and’ within ‘Sand’

Terminology

Substring

A string that exists within another string
ex. ‘and’ within ‘Sand’



String

Exercise 3

Use index accessors to print out the substring “Some” from the larger string “Something”

What if there’s an easier way?

Introducing Slices

Remember math class? Domains? Ranges?

x is defined on domain [0,3]…

Python kind of has its own version of this called slices.

string[start : stop]

Exercise 3b

Use slices to print out the substring “Some” from the larger string “Something”

NOICE

job

Translating Slice Syntax

“Everything from 0 to 4 except 4”

```
‘Something’[0:4]
‘Some’
```

Slices Save Time

Imagine writing this except with a string from 0 to 100

```
a = ‘Something’[0] # ‘S’
b = ‘Something’[1] # ‘o’
c = ‘Something’[2] # ‘m’
d = ‘Something’[3] # ‘e’
```

More Slice Stuff

What happens if we leave out parts of the domain?

```
'Something'[:4]
'Something'[0:]
```

What happens if we use negative numbers?

```
'Something'[0:-1]
'Something'[-1:-4]
```

Negative Indices Explained

Negative indices are really just regular indices with an implied length expression

```
ex = “NoIce”
ex[0:-1]
ex[0:len(ex) – 1]
ex[0:5 – 1]
ex[0:4]
```

Negative Indices Explained

Negative indices are really just regular indices with an implied expression with the string length

```
ex = “NoIce”
ex[-3:]
ex[len(ex) – 3:]
ex[5 - 3:]
ex[2:]
```

Programming Terminology

Expression

Code that can go on the right side of the assignment operator (‘=‘)

Sometimes you can *inline* an *expression* into other syntax, which can remove the need for an extra variable

```
example[0:len(example) – 1]
```

N O I C E
job

String Methods

String Methods

All strings have functions called *methods* that can be accessed by using the *dot operator* ('.'). These *methods* allow us to handle strings in complex ways.

Terminology

Method

Methods are functions that exist as properties on certain variables, data, or objects in a programming language.

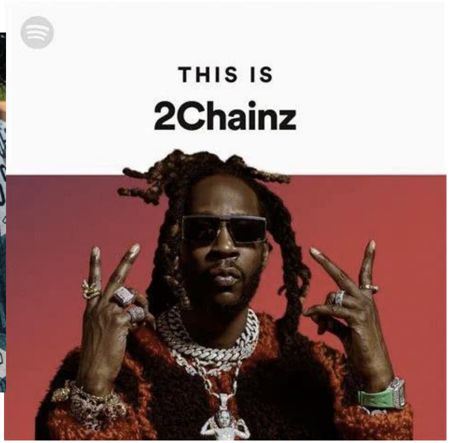
String.method(parameters)

Exercise 5

Capitalize your **firstName**	firstName.capitalize()
Make your **firstName** all uppercase	firstName.upper()
Make your **firstName** all lowercase	firstName.lower()
‘No**Ice**’ to ‘**Noice**’ in one line	‘NoIce’.lower().capitalize()

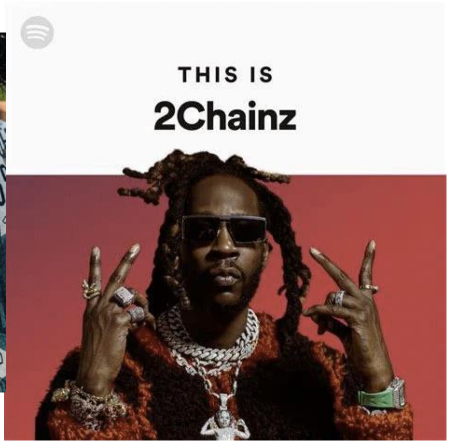
N O I C E
job

Terminology



Key Python term – can you guess?

Terminology

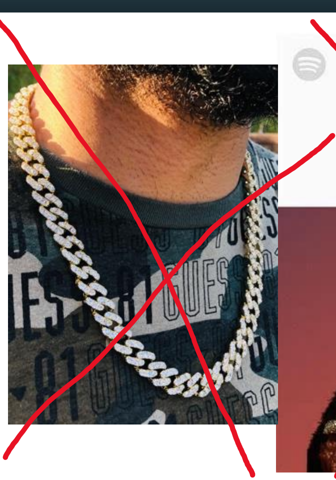


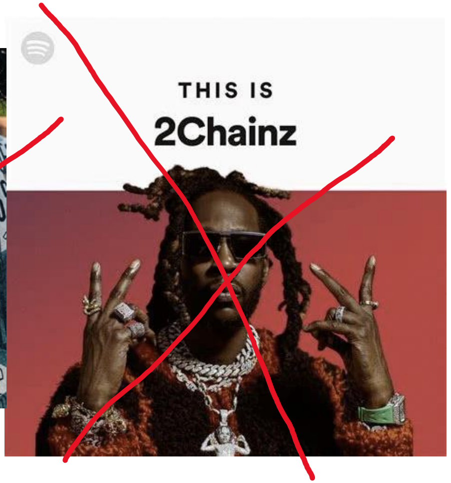
Key Python term – can you guess?

Chaining

Terminology

Chaining





Sequential access of properties or methods originating from the same variable, data, or object. This programming pattern can be more readable sometimes.

```
myVariable
  .filter(unwanted)
  .sum()
```

```
'NoIce'
  .lower()
  .capitalize()
```

Terminology

Immutable

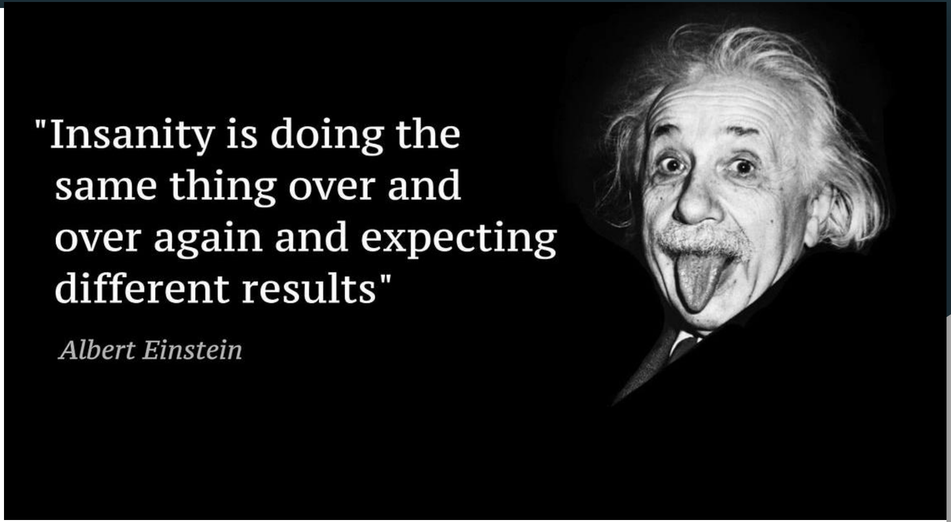
Once a string is initialized or returned from a method, the string is considered constant and *immutable*. Directly changing strings is forbidden in Python.

```
noIce = ‘NoIce’
noIce[2] = ‘i’    # …???
```

```
‘NoIce’                    # memory – ‘NoIce’
  .lower()                 # ‘noice’
  .capitalize()            # ‘Noice’
```

Terminology

Idempotent

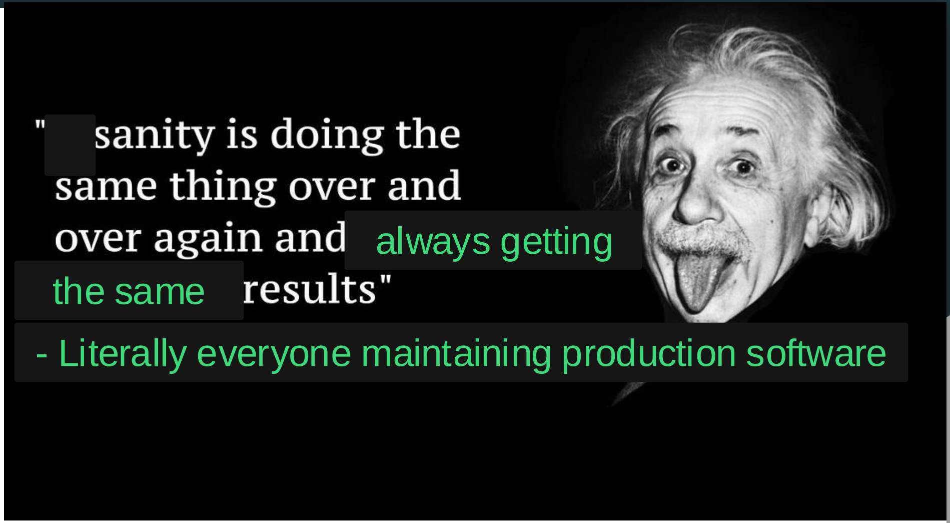

"Insanity is doing the same thing over and over again and expecting different results"
Albert Einstein

no matter how many times you execute something, you always get the same result for the same input

This concept will become **very important** in mathematics, computer science, and especially in **software engineering (i.e. API development).**

Terminology

Idempotent


"[redacted]sanity is doing the same thing over and over again and always getting the same results"
- Literally everyone maintaining production software

no matter how many times you execute something, you always get the same result for the same input

This concept will become **very important** in mathematics, computer science, and especially in **software engineering (i.e. API development).**

Terminology

Idempotent

Like math functions, calling string methods on the same exact strings and arguments **always** results in the same outputs.
(a.k.a. reliable, predictable code)

2 + 2 = 4
add(2, 2)

```
'NoIce'
  .lower()
  .capitalize() # ALWAYS returns 'Noice' given 'NoIce'
```

Split

string.split(delimiter)

method breaks apart the string anywhere that the *delimiter* is present and returns the resulting substrings in a list

‘Make like a banana and split!’.split(‘banana’)

=> [‘Make like a ’, ‘ and split!’]

Terminology

Delimiter

A character, symbol, or *substring* that indicates the beginning or end of data units within a larger sequence.

Important to understand for data manipulation with formats like *.csv* files.

Join

delimiter.join(subStrings)

takes a list of substrings and glues them together using a *delimiter*

‘banana’.join([‘Make like a ’, ‘ and split!’])

=> ‘Make like a banana and split!’

Replace

```
string.replace(search, replacement)
```

replaces all instances of <search> with <replacement> in the original string

```
'Make like a banana and split!'
            .replace('banana', 'pear')
=> 'Make like a pear and split!'
```

Formatting

There is a special character sequence that provides a pattern for formatting:

```
{}          ‘Today is {}’.format(day)

{id}        ‘Today is {month} {day}’.format(
                      month = ‘October’
                      day = ’27’
            )
```

Easier Formatting

print()
can directly include parameters to format

str()
converts numbers into strings

```
print('Today is', month, day)
=> 'Today is October 31'

day = 27
print('Today is', month, str(day))
=> 'Today is October 31'
```

Learning Objectives: Strings

Initialize string variables

Access values from string variables

Manipulate string values with immutability

Terminology Summary

Keyword – character sequence used for Python. No variables may be named ‘in’.

Substring – a string inside a string

Index – number (key) that accesses a value from a string (lock)

Expression – any code that be on the right side of an = sign

Method – function that exists off a string, accessed by the dot **.** operator (str.split)

Terminology Summary

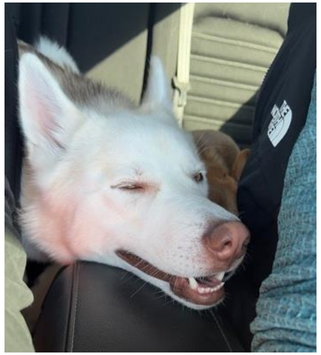
Chaining – calling string methods after another (str.lower().capitalize())

Immutable – can’t directly change strings, can only make new strings from strings

Idempotent – same input, **ALWAYS** same output. Reliable code that d

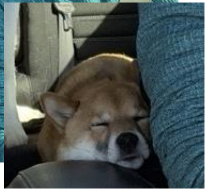
String – immutable character sequences with reliable, idempotent methods

Delimiter – substring that signals the beginning or end of other data inside a string



NOICE

job



TO THE
BARK SIDE

Exercise

There’s a mystery string of your name and then a cool phrase **‘N O I C E’.**

mysteryString = **firstName** + **lastName** + **‘N O I C E’**

1) Verify that the cool phrase actually exists in **mysteryString**
2) Get the cool phrase out as substring
3) Get the cool phrase out in other ways

Exercises

Find 2 ways to replace all the spaces ‘ ‘ with commas ‘,’ in **“N O I C E”**

Turn **‘N O I C E’** into **‘noice’**

Challenge:
How many ways can you concatenate your **firstName** and **lastName?**

Exercise

Capitalize your **firstName**

Make your **firstName** all uppercase

Make your **firstName** all lowercase

‘NoIce’ to **‘Noice’**

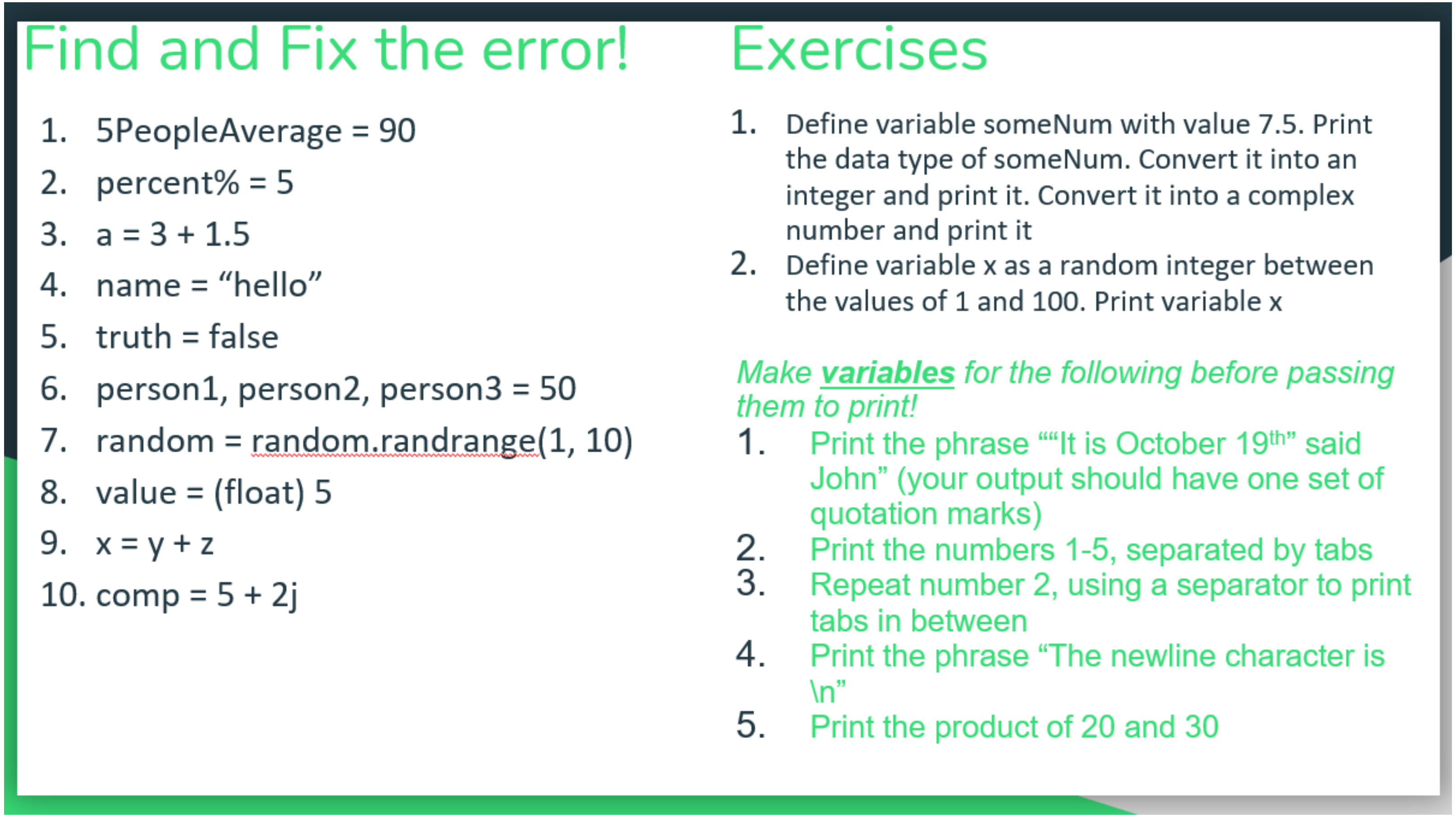
Exercises – Review

Define variable x as the following string: “Hello World”

a. Print the length of the string
b. Get the characters from index 2 to index 4
c. In the same line, convert the string to upper case and replace the character “l” with the character “s”

Define variable name with your name. Define variable age with your age. Define variable txt with the string “My name is {} and I am {} years old. Use the format() method to place the correct variables into the placeholders

Variables Review



Find and Fix the error!

1. 5PeopleAverage = 90
2. percent% = 5
3. a = 3 + 1.5
4. name = “hello”
5. truth = false
6. person1, person2, person3 = 50
7. random = random.randrange(1, 10)
8. value = (float) 5
9. x = y + z
10. comp = 5 + 2j

Exercises

1. Define variable someNum with value 7.5. Print the data type of someNum. Convert it into an integer and print it. Convert it into a complex number and print it
2. Define variable x as a random integer between the values of 1 and 100. Print variable x

*Make **variables** for the following before passing them to print!*

1. Print the phrase ““It is October 19th” said John” (your output should have one set of quotation marks)
2. Print the numbers 1-5, separated by tabs
3. Repeat number 2, using a separator to print tabs in between
4. Print the phrase “The newline character is \n”
5. Print the product of 20 and 30