FULL-STACK DEVELOPMENT

SESSION 2

BY: MASOUD DORRIKHTEH

Agenda

- Review
- Notes and QnA
- Some Theory
- Coding!

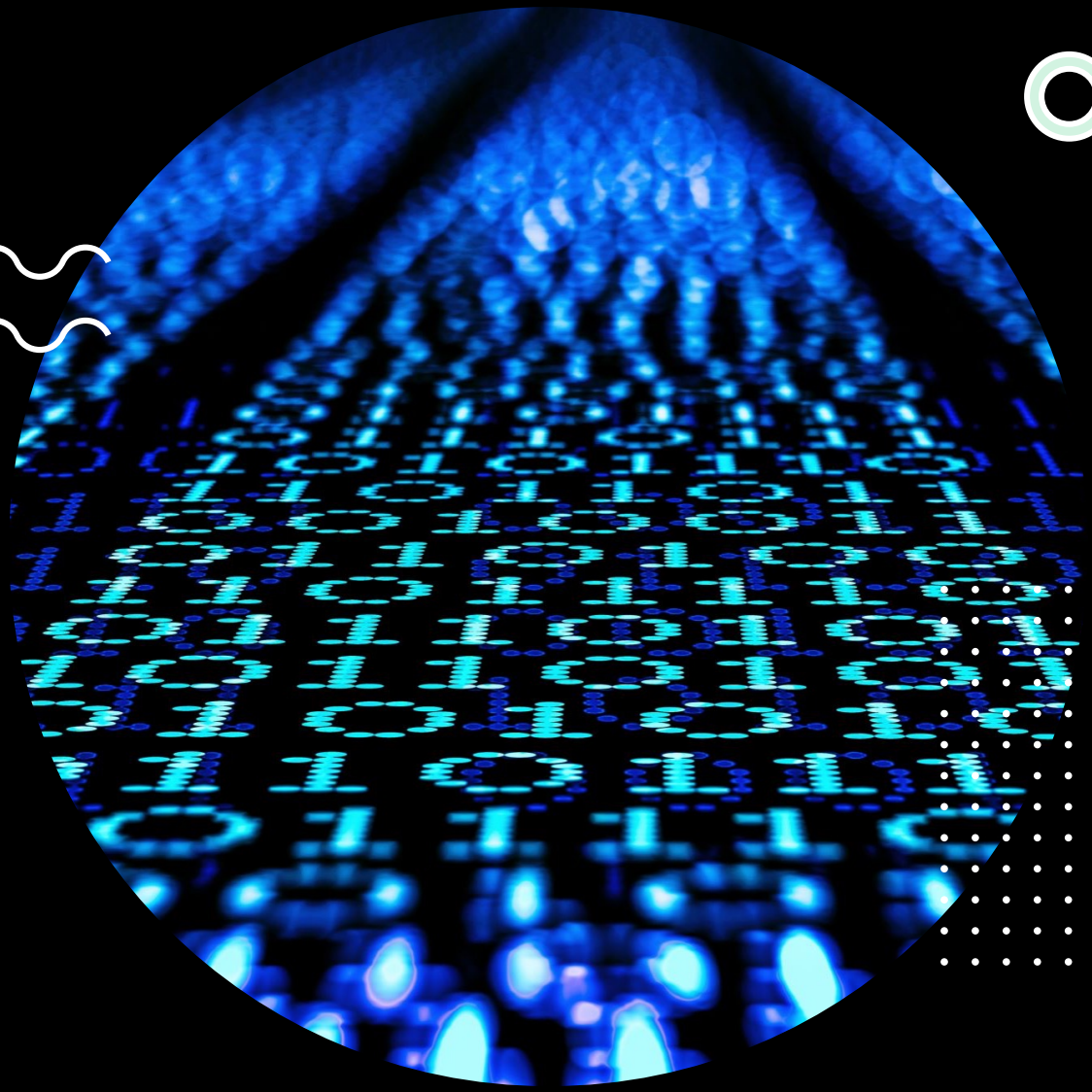
What did we cover

- Programming Basics
- Working with an IDE
- Identifiers
- Identifiers and Variables
- Statements
- Operators and Data Types (int, string, float)

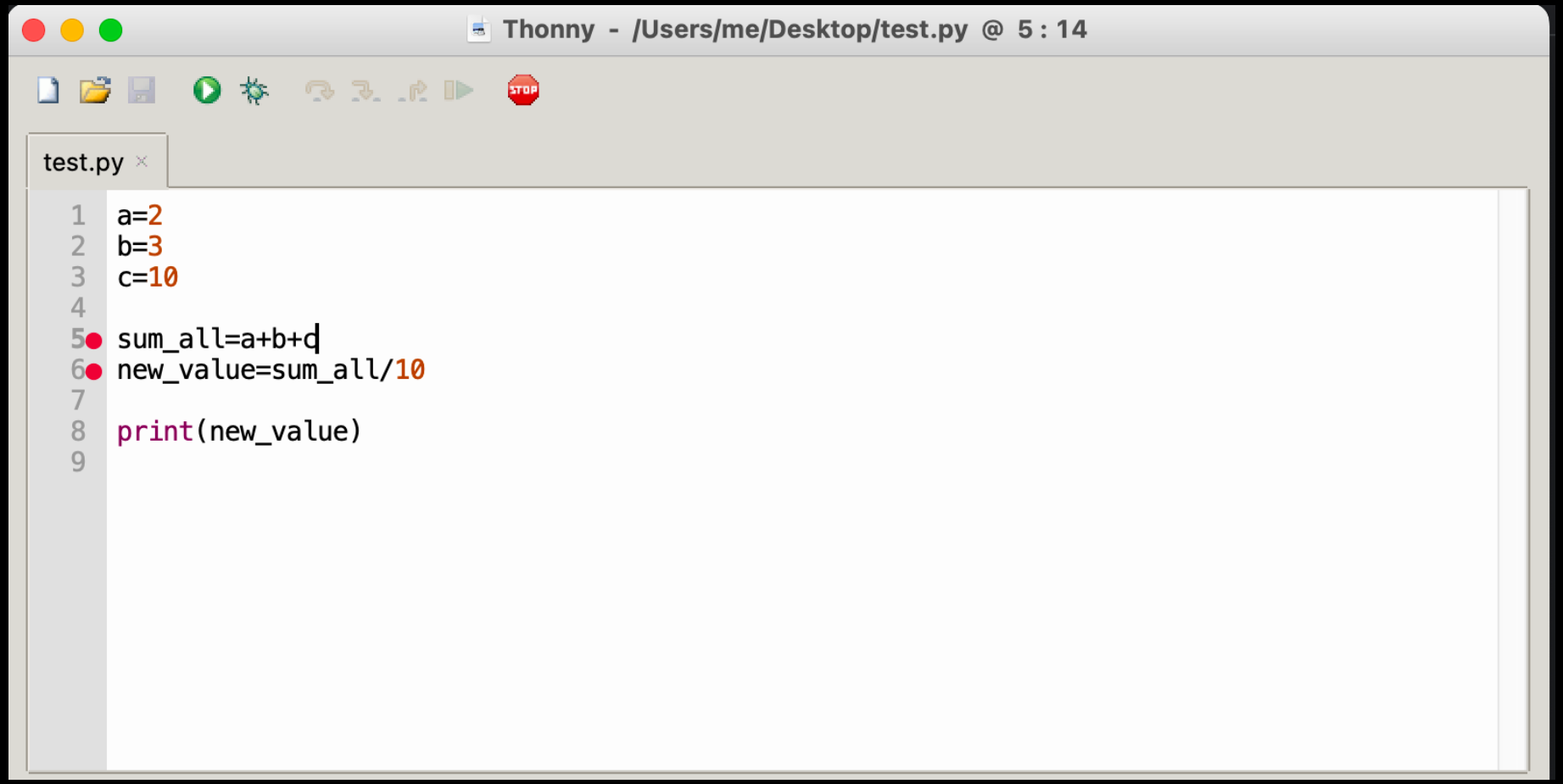
Some Notes…

- Your submissions are important
- You didn’t ask questions
  - $\frac{\frac{2}{3}}{4} \neq \frac{\frac{4}{2}}{3}$ --> (a/b)/c ≠ c/(a/b)
- Order matters
- Fail your code!

NOW, LETS CODE...



Debugging 1 - Break Points



Writing Comments

```
#This is a comment
print "Hello, World!"

• ---------

"""
Comment in more than just one line
"""
print "Hello, World!"
```

List, Set and Tuple

List	Set (No Duplicate)	Tuple (Cannot be changed)
Lists is Mutable	Set is Mutable	Tuple is Immutable
It is Ordered collection of items	It is Unordered collection of items	It is Ordered collection of items
Items in list can be replaced or changed	Items in set cannot be changed or replaced	Items in tuple cannot be changed or replaced

```
# Creating a List of numbers
List = [10, 20, 14]
print("\nList of numbers: ")
print(List)
```

```
dataScientist = set(['Python', 'R',
'SQL', 'Git', 'Tableau', 'SAS’])
dataEngineer = set(['Python', 'Java',
'Scala', 'Git', 'SQL', 'Hadoop'])
```

```
# Creating a Tuple with
# the use of list
list1 = (1, 2, 4, 5, 6)
```

*Use Google to learn about Python Dictionary.

Source: https://www.geeksforgeeks.org/difference-between-list-vs-set-vs-tuple-in-python/

Working with lists and array

- Accesing an item
  - Value_from_list = my_list[0]
  - Value_from_nested_list=my_nested_list[0:1]

- A=B Aliasing (Be careful, most of the time use A=B[:]
- Help(A)
- **Exercise: Look for the differences between extend and append**

PYTHON CODE SCOPE AND CODE BLOCK

LOOK AT THE BOARD!

FLOW CONTROL

Conditions

- If, elif, else

```
if num > 0:
        print("Positive number")
elif num == 0:
        print("Zero")
else:
        print("Negative number")
```

Nested If

- Implement the previous example without using “elif”.

Solution

```
num = float(input("Enter a number: "))
if num >= 0:
        if num == 0:
                print("Zero")
        else:
        print("Positive number")
else:
        print("Negative number")
```

For loop

1. For each

```
for value in my_list:
    sum = sum+value
```

2. For range

```
for i in range(5):
    print(value)
```

3. For loop with else

```
my_numbers = [0, 1, 2]
for i in my_numbers:
    print(i)
else:
    print(”No more items.")
```

While loop

```
i = 1
while i < 100:
        print (i**2)
        i += i**2
```

Class Exercise

- Write a program that asks for a student name and prints his/her grade..
  - Hint: use Python Dictionary

Functions

- A group of statements to perform a specific task

```
def greet(name):
        print("Hello, " + name + ". Good morning!")
```

```
def send_email(to,subject,content):
        #some code to send email
        return True

if (send_email("test@test.com","Hello Test", "this is a sample email")):
        print("email sent")
        else:
                print("email failed")
```

More on GitHub

- Push
- Pull Request
- Pull

Boolean Expressions

- == equals: 1 == 1 yields True
- ! = does not equal: 1 != 1 yields False
- > greater than: 3 > 2 yields True
- >= greater than or equal: 5 >= 5 yields True

Logical Operatord

- Or
- And
- not

Combination of Conditions

and

a	b	a and b
True	True	True
True	False	False
False	False	False
False	True	False

or

a	b	a or b
True	True	True
True	False	True
False	True	True
False	False	False

For more info: https://www.pythontutorial.net/python-basics/python-logical-operators/

Variable Scope

- Global variables can be accessed from (almost) anywhere.
  - Add ”global” before the variable name --> (global x = 2)
- Local variables can only be accessed within a function
- Some examples…

Object Oriented Programming (OOP)

- Code as if you are creating an object
  - Think of attributes
  - Think of actions
  - An object (instance) is an instantiation of a class
- Let’s build a car!

Class example

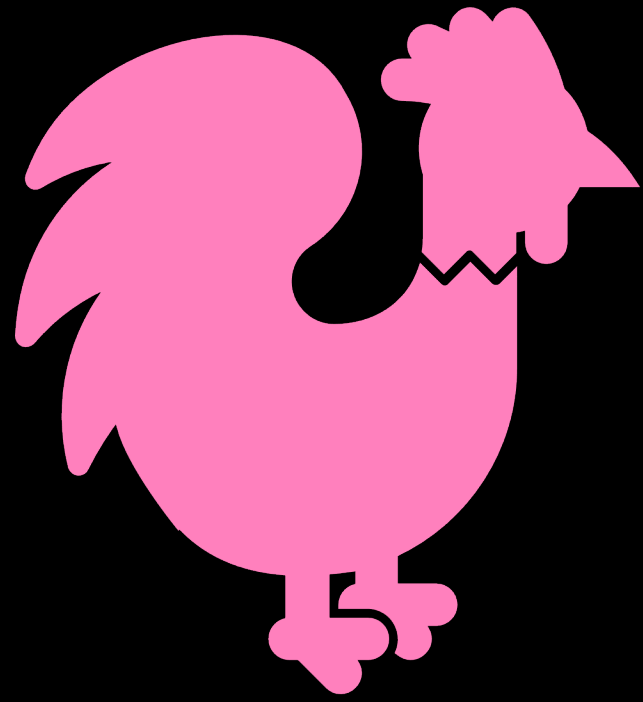
```
class Vehicle:

     # class attribute
     type = ”Sedan"

     # instance attribute
     def __init__(self, production_year, make):
          self.production_year = production_year
          self.make = make


# instantiate the Parrot class
model_3 = Vehicle("2020", "Tesla")
model_e = Vehicle("1990", "Mercedes-Benz")

# access the class attributes
print("Model 3 is a {}".format(model_3.__class__.type))

# caclculate how old is the car
```

CLASS – INHERITANCE

Inheritance Example

```
# parent class
class Vehicle:

      def __init__(self):
            print("Vehicle is Ready")

      def who_is_this(self):
            print("Vehicle")

      def engine_start(self):
            print("starting engine")

      # child class

class Truck(Vehicle):

      def __init__(self):
            # call super() function
            super().__init__()
            print("Truck is Ready")

      def who_is_this(self):
            print("Truck")

      def offload(self):
            print("Truck is offloading")


big_truck = Truck()
big_truck.who_is_this()
big_truck.engine_start()
big_truck.offload()
```

C L A S S – E N C A P S U L A T I O N

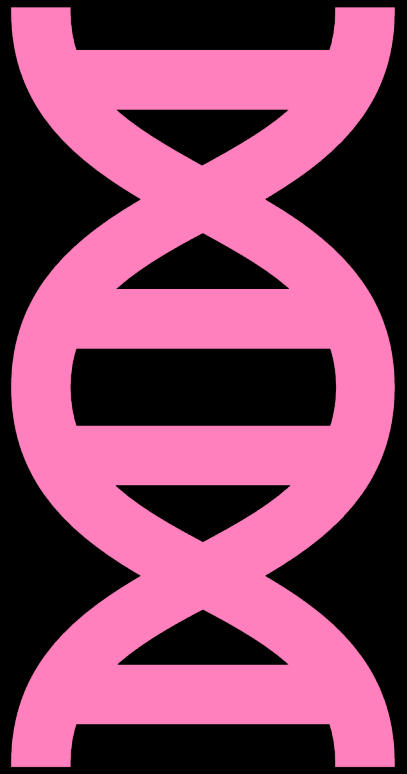


Encapsulation Example

```
class Computer:

    def __init__(self):
        self.__maxprice = 900

    def sell(self):
        print("Selling Price: {}".format(self.__maxprice))

    def setMaxPrice(self, price):
        self.__maxprice = price

c = Computer()
c.sell()

# change the price
c.__maxprice = 1000
c.sell()

# using setter function
c.setMaxPrice(1000)
c.sell()
```

Source: https://www.programiz.com/python-programming/object-oriented-programming

CLASS - POLYMORPHISM



Polymorphism Example

```
class Parrot:

    def fly(self):
    print("Parrot can fly")
    def swim(self):
    print("Parrot can't swim")

class Penguin:

    def fly(self):
    print("Penguin can't fly")
    def swim(self):
    print("Penguin can swim")

# common interface
def flying_test(bird):
    bird.fly()

#instantiate objects
blu = Parrot()
peggy = Penguin()

# passing the object
flying_test(blu)
flying_test(peggy)
```

Source: https://www.programiz.com/python-programming/object-oriented-programming

Class Exercise 2

- Ask for student name and grade
- Save them
- Let the user to search for students' grade
- Hint: Ask the user if s/he wants to add or search
- Bounce: Allow the user to delete or update the data