Principles of Software Programming

Lecture 6: More on data structures and classes

WIRTSCHAFTS UNIVERSITÄT WIEN VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS

Anton Yeshchenko

SS 2018

Some slides and/or ideas were borrowed from:
MIT Introduction to Computer Science and Programming in Python
and Svitlana Vakulenko WS 2017 lecture slides

APRIL 2018

EFMD EQUIS ACCREDITED AACSB ACCREDITED ASSOCIATION OF MBAS ACCREDITED

Recap of the Lecture 5

- **OOP**!

Recap of the Lecture 5

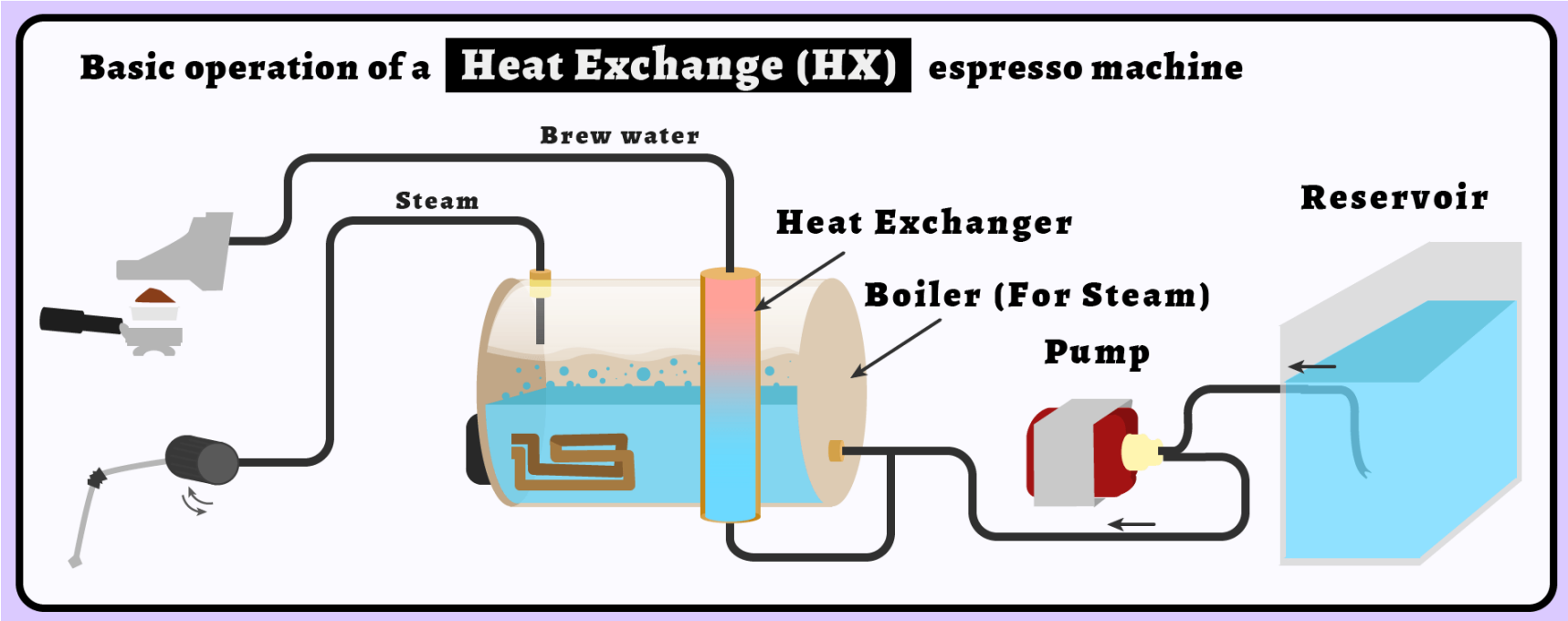
- OOP!

1. Encapsulation

A language mechanism for restricting direct access to some of the object's components

- private variables
- private methods of a class

How espresso machine works!



HTTPS://KNOWYOURGRINDER.COM/WP-CONTENT/UPLOADS/2016/01/CUPANDBREW-HOW_IT_WORKS-HEAT_EXCHANGER_ESPRESSO_MACHINE.PNG

How espresso machine need to work!

- **MAKE_COFFEE()**

HTTP://MEDIA.CULTUREMAP.COM/CROP/B6/EA/633X475/ESPRESSO-MACHINE_152205.JPG

Recap of the Lecture 5

- OOP!

1. Encapsulation
2. Abstraction

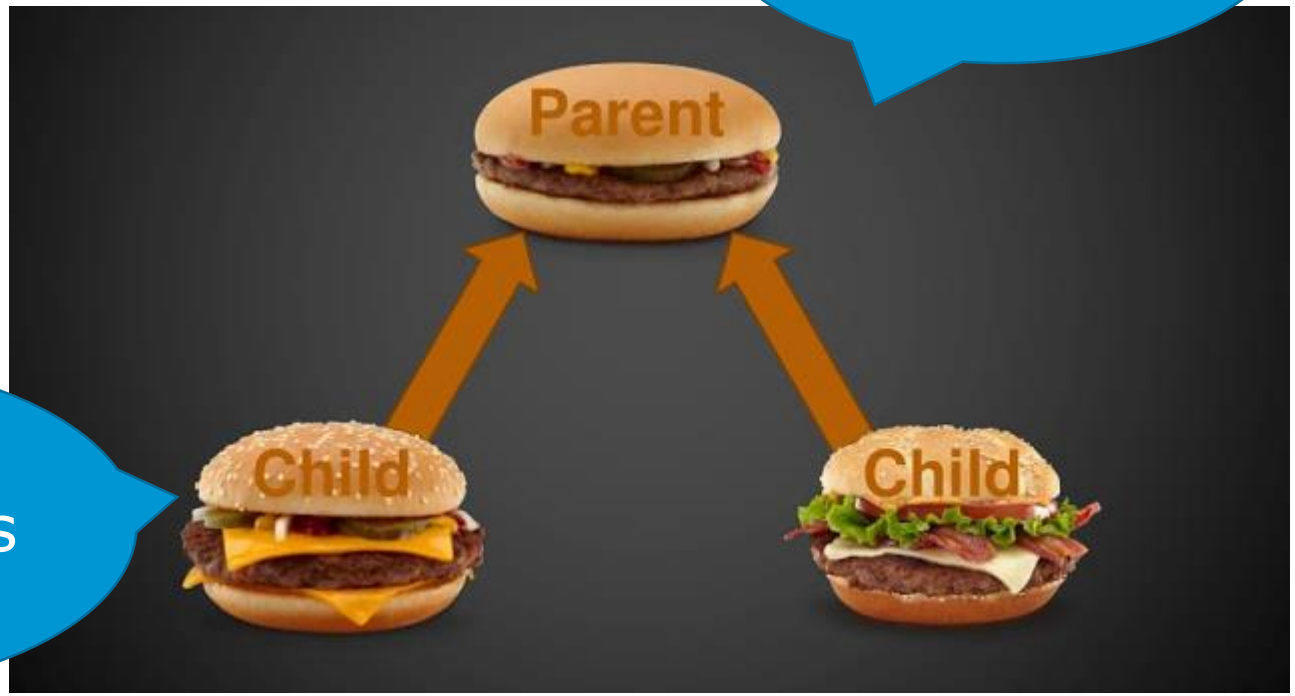
Its main goal is to handle complexity by hiding unnecessary details from the user.

Recap of the Lecture 5

- OOP!

1. Encapsulation
2. Abstraction
3. Inheritance

Super class,
Base class,
Parent class



Subclass,
Derived class
Child class

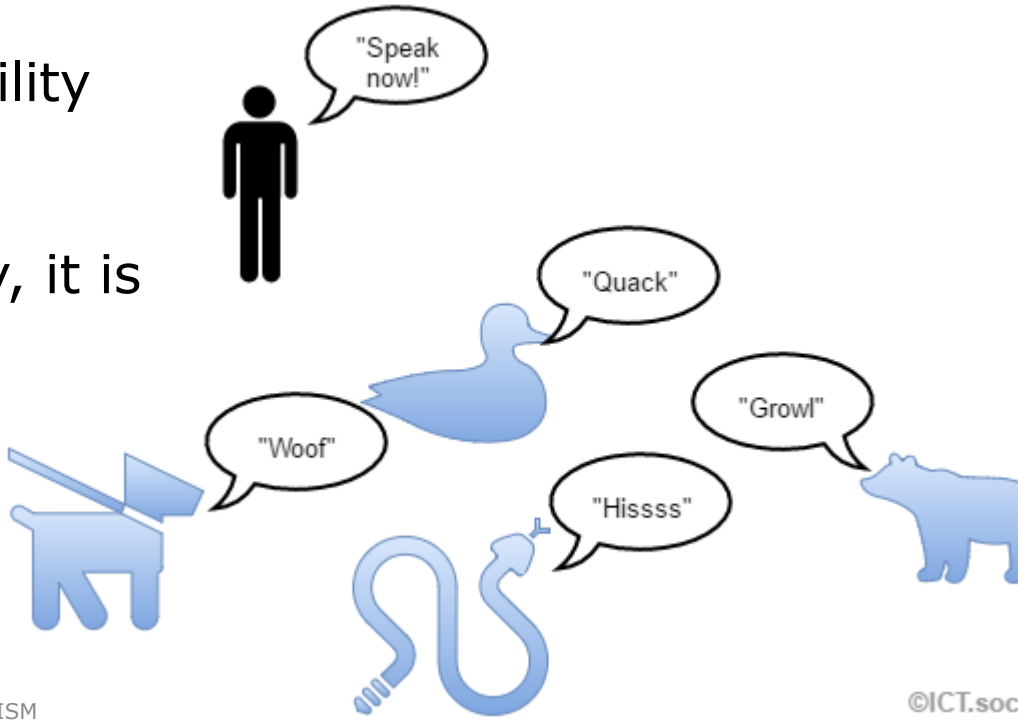
HTTPS://4.BP.BLOGSPOT.COM/-SMG1TY6CO1W/VRCWM2ERF5I/AAAAAAAAET4/0BM6AQSIN8M/S640/OOP%2B-%2BINHERITANCE.JPG

Recap of the Lecture 5

- OOP!

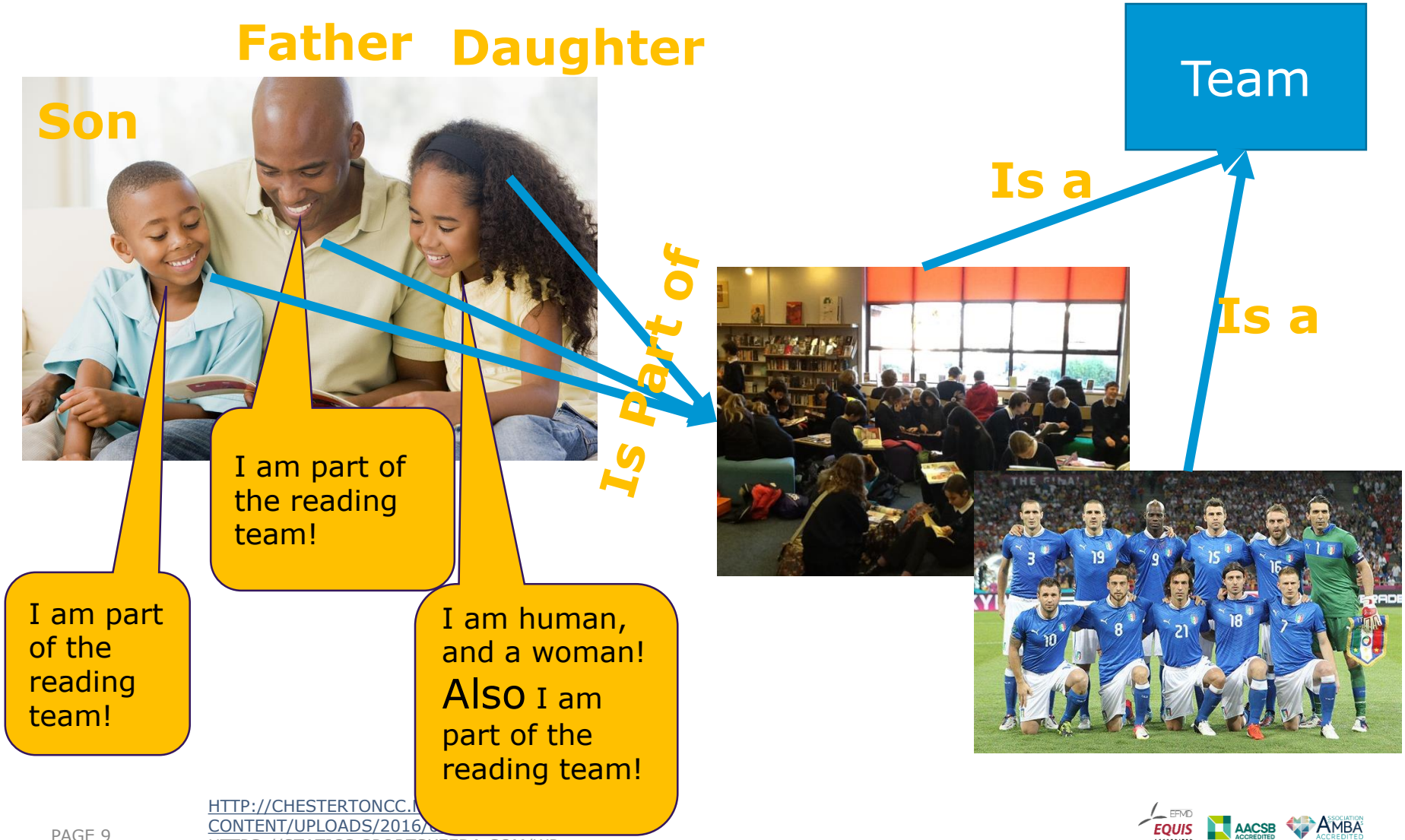
1. Encapsulation
2. Abstraction
3. Inheritance
4. Polymorphism

polymorphism refers to a programming language's ability to process objects differently depending on their data type or class. More specifically, it is the ability to redefine *methods* for *derived classes*



HTTPS://WWW.WEBOPEDIA.COM/TERM/P/POLYMORPHISM.HTML

“Is a”, “is part of” (composition)



HTTP://CHESTERTONCC.
CONTENT/UPLOADS/2016/
HTTPS://STATICS.SPORTSKEEDA.COM/WP-CONTENT/UPLOADS/2015/04/ITALY-1428968210.JPG

Short recap of what has just happened!

1. Hardware – Software (Intro)
2. Basics of programming (Operators, Variables, Functions, Packages)
3. Control flow and lists (if-else, loops (for, while), list)
4. Classes (Objects, Instances, Constructors, Method overloading)
5. Object Oriented Programming (Encapsulation, Abstraction, Inheritance, and Polymorphism)

Short recap of what has just happened!

1. Hardware – Software (Intro)
2. Basics of programming (Operators, Variables, Functions, Packages)
3. Control flow and lists (if-else, loops (for, while), list)
4. Classes (Objects, Instances, Constructors, Method overloading)
5. Object Oriented Programming (Encapsulation, Abstraction, Inheritance, and Polymorphism)
6. today????? Advanced topics about classes, lists, files

Short recap of what has just happened and will happen!

1. Hardware – Software (Intro)
2. Basics of programming (Operators, Variables, Functions, Packages)
3. Control flow and lists (if-else, loops (for, while), list)
4. Classes (Objects, Instances, Constructors, Method overloading)
5. Object Oriented Programming (Encapsulation, Abstraction, Inheritance, and Polymorphism)
6. Today: Advanced topics about classes, lists, files
7. Solving all the EXAMs!! (Thursday)
8. Grand recap! (Next Tuesday)
9. Wahlfach (Extra!!): Exceptions, Dynamic data structures, Algorithms

Today is in black

- **Programming:** algorithms, syntax and semantics, programming, compiler, interpreter
- **Basics and types:** variables, operations, primitive data types, Strings, static vs dynamic typing, explicit vs implicit type casting
- **Control flow and functions:** if-else branches, loops, functions (parameters, return values)
- **Lists:** Arrays (lists), create and fill Arrays, multidimensional Arrays
- **Classes:** Class vs Instance of class, objects, create objects, instance variable, constructor, method overloading
- **Inheritance:** inherit classes, method overriding, problems and solutions for multiple inheritance
- **Information hiding:** variable access, access modifier (Java) and naming conventions, get- and set-methods
- **Object oriented programming:** why OOP, inheritance ("is-a"- and "is-part-of"-relations), information hiding/encapsulation, abstract classes, Super-constructor, polymorphism

Today

1. Files, read and write
2. Multidimensional array
3. Access modifier (java)
4. Abstract classes, Super constructor
5. Multiple-inheritance problems

Opening files

CSV File Test

File

	Column 1	Column 2	Column 3	Column 4	Column
	531	4/10/2007	20:39:00	108	TMZ095
	530	4/10/2007	20:07:00	107	TMZ095
	529	4/10/2007	18:57:00	96	TMZ095
	528	4/10/2007	18:28:00	99	TMZ095
	527	4/10/2007	18:01:00	109	TMZ095
	526	4/10/2007	10:33:00	117	TMZ095
	525	4/10/2007	10:21:00	124	TMZ095
	524	4/10/2007	8:58:00	103	TMZ095
	523	4/10/2007	8:16:00	103	TMZ095
	522	4/10/2007	7:34:00	106	TMZ095
	521	4/9/2007	23:16:00	99	TMZ095
	520	4/9/2007	17:51:00	99	TMZ095
	519	4/9/2007	16:11:00	94	TMZ095
	518	4/9/2007	10:59:00	105	TMZ095

What You Need In Order To Read Information From A File

1. Open the file and associate the file with a file variable.
2. A command to read the information.
3. A command to close the file.

Slide from: http://kremer.cpsc.ucalgary.ca/courses/cpsc231/lectureNotes/files.ppt

1. Opening Files

Prepares the file for reading:

A. Links the file variable with the physical file (references to the file variable are references to the physical file).
B. Positions the file pointer at the start of the file.

Format:[1]

```
<file variable> = open(<file name>, "r")
```

Example:

(Constant file name)

```
inputFile = open("data.txt", "r")
```

OR

(Variable file name: entered by user at runtime)

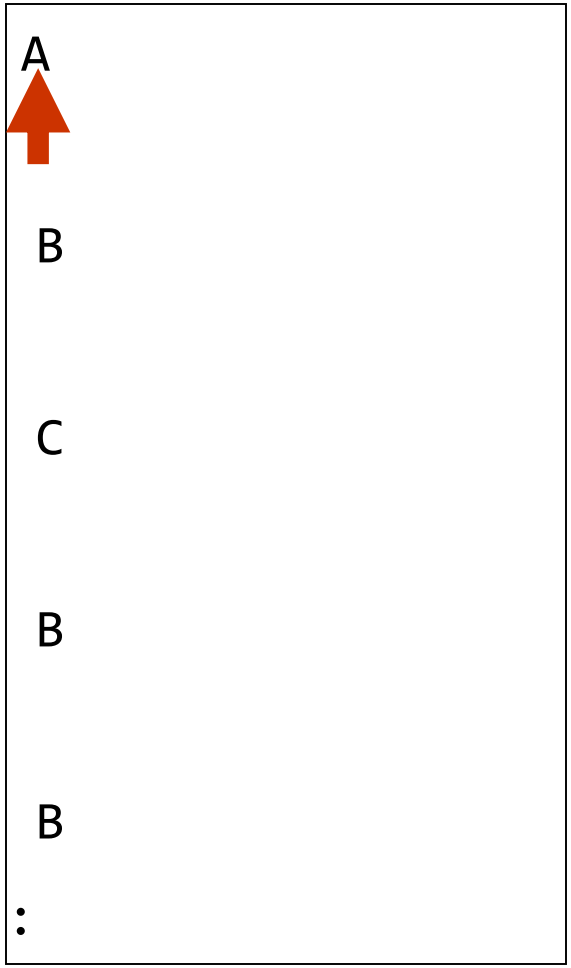
```
filename = input("Enter name of input file: ")
inputFile = open(filename, "r")
```

Slide from: http://kremer.cpsc.ucalgary.ca/courses/cpsc231/lectureNotes/files.ppt

1 Examples assume that the file is in the same directory/folder as the Python program.

B. Positioning The File Pointer

letters.txt



Slide from: http://kremer.cpsc.ucalgary.ca/courses/cpsc231/lectureNotes/files.ppt

2. Reading Information From Files

- Typically reading is done within the body of a loop
- Each execution of the loop will read a line from the file into a string

Format:

```
for <variable to store a string> in <name of file variable>:
    <Do something with the string read from file>
```

Example:

```
for line in inputFile:
    print(line)  # Echo file contents back onscreen
```

Slide from: http://kremer.cpsc.ucalgary.ca/courses/cpsc231/lectureNotes/files.ppt

Closing The File

- Although a file is automatically closed when your program ends it is still a good style to explicitly close your file as soon as the program is done with it.
  - What if the program encounters a runtime error and crashes before it reaches the end? The input file may remain 'locked' an inaccessible state because it's still open.
- **Format:**

```
<name of file variable>.close()
```

- **Example:**

```
inputFile.close()
```

Slide from: http://kremer.cpsc.ucalgary.ca/courses/cpsc231/lectureNotes/files.ppt

Writing to the file

```
file = open("testfile.txt","w")

file.write("Hello World")
file.write("This is our new text file")
file.write("and this is another line.")
file.write("Why? Because we can.")

file.close()
```

Slide from: http://kremer.cpsc.ucalgary.ca/courses/cpsc231/lectureNotes/files.ppt

Lists

- Creating list
  - list_grades = list()
- Adding elements
  - list_grades.append(element)
- Access elements of a list
  - list_grades[0], list_grades[-1],. …

Exercise 1

Goto: Jupyter notebook

In lecture_students folder find file lecture_6_names.txt

"lecture_students/lecture_6_names.txt"

Read names from the file into a list

Output on the screen (print) only the names starting with 'X'

#hint#

1. Use two loops
2. A name will be a string -> you can check if first letter is "X"

https://i.ytimg.com/vi/Z91tys7dYp4/hqdefault.jpg

Multidimensional arrays

- List that contains lists

- Create:
- multi_list = list()
- for i in range(10):
  - multi_list.append([10,10,15])

Will have a shape 10x3

Example with average-salary

Self parameter of the class!

```
class Restaurant(object):
    bankrupt = False
    def open_branch(self):
        if not self.bankrupt:
            print("branch opened")
```

HTTPS://STACKOVERFLOW.COM/QUESTIONS/6990099/EXPLAINING-THE-PYTHON-SELF-VARIABLE-TO-A-BEGINNER?UTM_MEDIUM=ORGANIC&UTM_SOURCE=GOOGLE_RICH_QA&UTM_CAMPAIGN=GOOGLE_RICH_QA

Self parameter of the class!

```
class Restaurant(object):
    bankrupt = False
    def open_branch(self):
        if not self.bankrupt:
            print("branch opened")

x = Restaurant()
```

Self parameter of the class!

```
class Restaurant(object):
    bankrupt = False
    def open_branch(self):
        if not self.bankrupt:
            print("branch opened")

x = Restaurant()
x.bankrupt
```

Self parameter of the class!

```
class Restaurant(object):
    bankrupt = False
    def open_branch(self):
        if not self.bankrupt:
            print("branch opened")

x = Restaurant()
x.bankrupt
Restaurant().bankrupt
```

Self parameter of the class!

```
class Restaurant(object):
    bankrupt = False
    def open_branch(self):
        if not self.bankrupt:
            print("branch opened")

x = Restaurant()
x.bankrupt
Restaurant().bankrupt
```

```
>>> x = Restaurant()
>>> x.bankrupt
False

>>> y = Restaurant()
>>> y.bankrupt = True
>>> y.bankrupt
True

>>> x.bankrupt
False
```

ACCESS modifier!!!!! (java vs python)

- Java has explicit access state of object variables

	Class	Package	Subclass (same pkg)	Subclass (diff pkg)	World
public	+	+	+	+	+
protected	+	+	+	+	
no modifier	+	+	+		
private	+				

+ : accessible
blank : not accessible

ACCESS modifier!!!!! (java vs python)

- Python has explicit access state of object variables

- All public!
- Single underscore '_variable' to say that variable **"private"**
- You can use two underscores '__variable' to make **private**

Abstract class

- An **abstract** method is a method that is declared, but contains no implementation. **Abstract classes** may not be instantiated, and require subclasses to provide implementations for the **abstract** methods.

- *Animals -> make_noise()*

Abstract class

```python
from abc import ABC, abstractmethod

class AbstractClassExample(ABC):
    @abstractmethod
    def do_something(self):
        pass
```

```python
class DoAdd42(AbstractClassExample):
    pass

x = DoAdd42()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-66-258055179d5f> in <module>()
      2     pass
      3
----> 4 x = DoAdd42()

TypeError: Can't instantiate abstract class DoAdd42 with abstract methods do_something
```

Super constructor

- Super() is reference to the base class

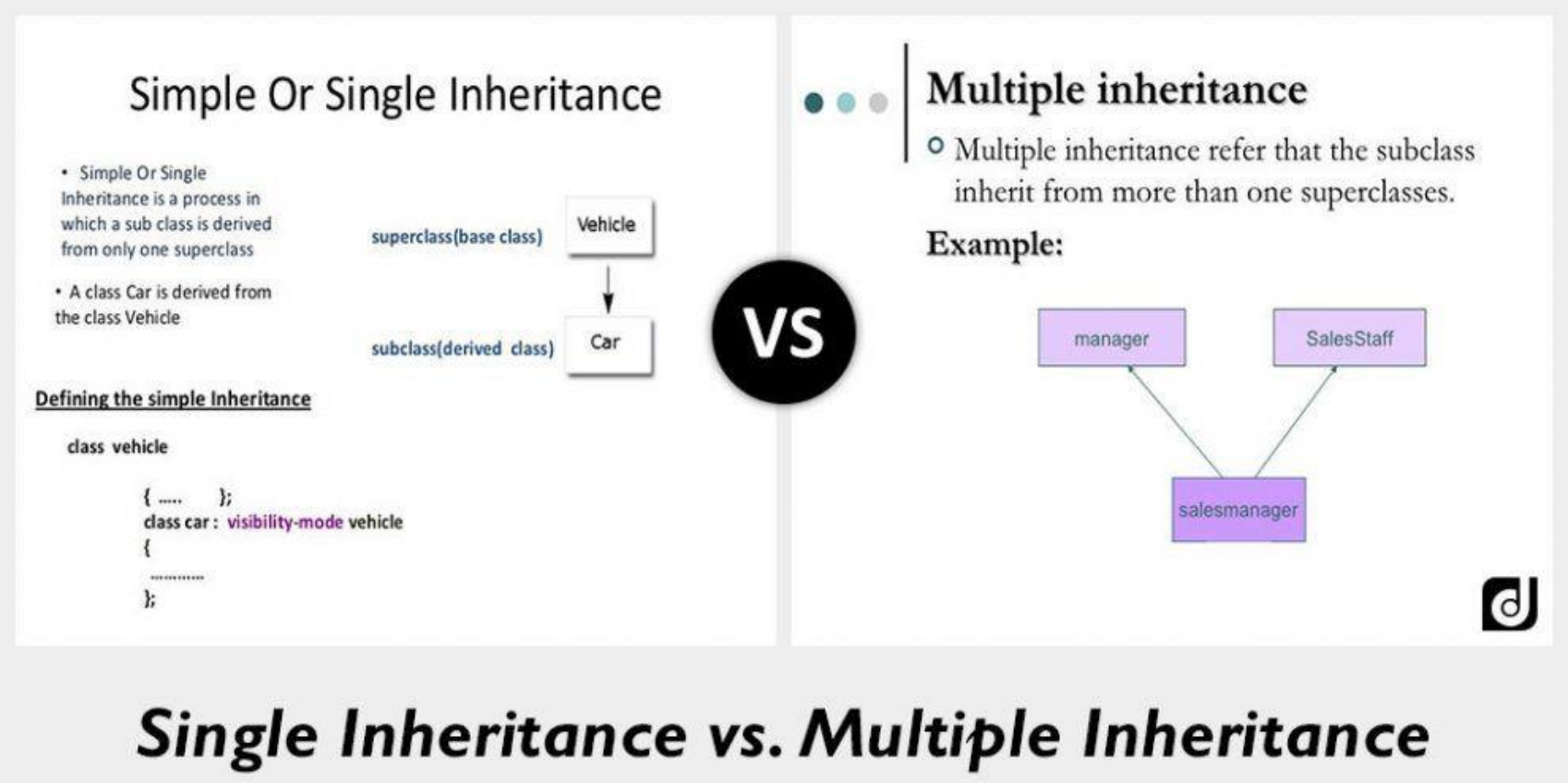
```
class A(object):
    def __init__(self):
        print("world")

class B(A):
    def __init__(self):
        print("hello")
        super().__init__()
```

```
B()
```

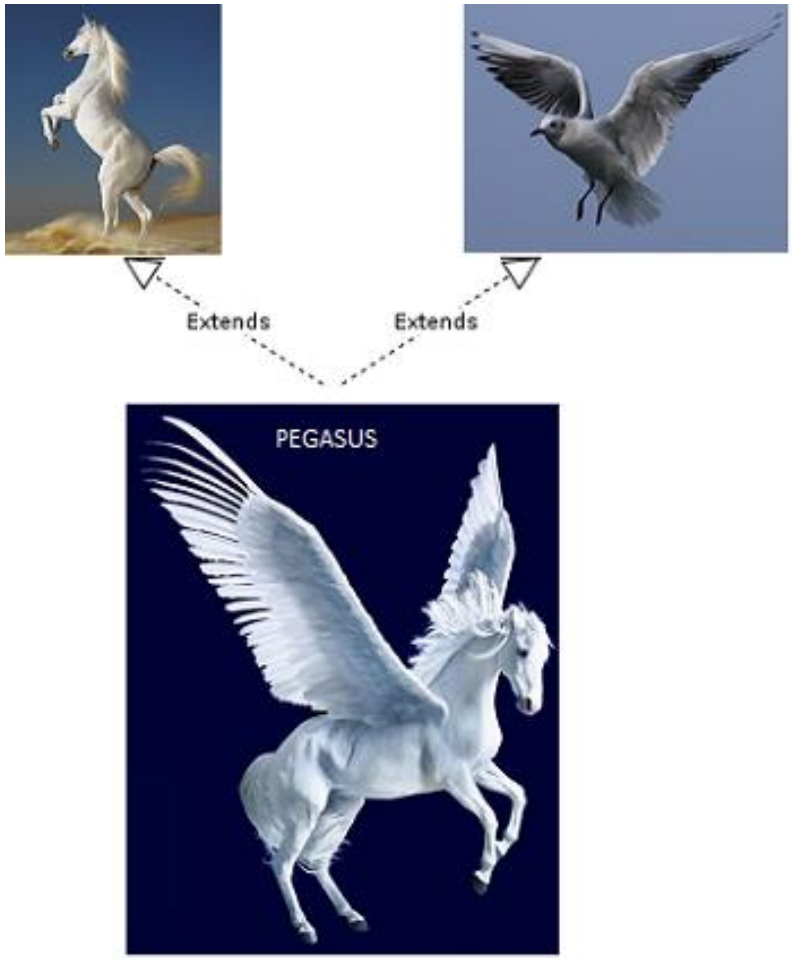
```
hello
world
```

5. Multiple-inheritance problems (and solutions)



HTTPS://WWW.DIFFERENCEBTW.COM/WP-CONTENT/UPLOADS/2017/06/SINGLE-INHERITANCE-VS-MULTIPLE-INHERITANCE-990X495.JPG

5. Multiple-inheritance problems (and solutions)



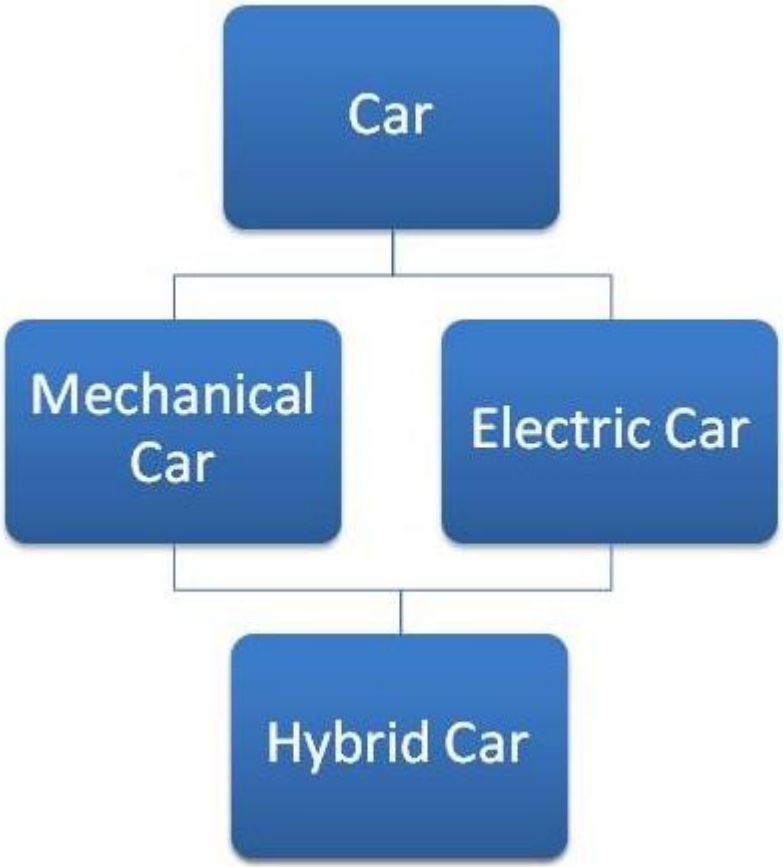
HTTPS://JAVAPAPERS.COM/WP-CONTENT/UPLOADS/2012/09/MULTIPLE-INHERITANCE.PNG

5. Multiple-inheritance problems (and solutions)

Car has a method change_gears()

Mechanical vs Electric

What method will Hybrid Car have?



```python
class Car:
    def m(self):
        print("m of A called")

class ManualCar(Car):
    def m(self):
        print("m of B called")

class ElectricCar(Car):
    def m(self):
        print("m of C called")

class HybridCar(ManualCar,ElectricCar):
    pass
```

```python
x = HybridCar()
x.m()
```

```
m of B called
```

```python
class Car:
    def m(self):
        print("m of A called")

class ManualCar(Car):
    pass
    #def m(self):
    #    print("m of B called")

class ElectricCar(Car):
    def m(self):
        print("m of C called")

class HybridCar(ManualCar,ElectricCar):
    pass
```

```python
x = HybridCar()
x.m()
```

```
m of C called
```

Method Resolution Order (python)

- the **order** in which base classes are searched when looking for a **method**

```python
class Car:
    def info(self):
        print("I am a car")

class ManualCar(Car):
    def info(self):
        print("I am manual")

class ElectricCar(Car):
    def info(self):
        print("I am Electric")

class HybridCar(ManualCar,ElectricCar):
    def info(self):
        print("I am hybrid")
        Car.info(self)
        ManualCar.info(self)
        ElectricCar.info(self)
```

```python
a = HybridCar()
a.info()
```

```
I am hybrid
I am a car
I am manual
I am Electric
```

```python
class Car:
    def info(self):
        print("I am a car")

class ManualCar(Car):
    def info(self):
        print("I am manual")
        Car.info(self)

class ElectricCar(Car):
    def info(self):
        print("I am Electric")
        Car.info(self)

class HybridCar(ManualCar,ElectricCar):
    def info(self):
        print("I am hybrid")
        ManualCar.info(self)
        ElectricCar.info(self)
```

```python
a = HybridCar()
a.info()
```

```
I am hybrid
I am manual
I am a car
I am Electric
I am a car
```

SUPER solution in python (thanks to Method Resolution Order)

```python
class Car:
    def info(self):
        print("I am a car")

class ManualCar(Car):
    def info(self):
        print("I am manual")
        super().info()

class ElectricCar(Car):
    def info(self):
        print("I am Electric")
        super().info()

class HybridCar(ManualCar,ElectricCar):
    def info(self):
        print("I am hybrid")
        super().info()
```

```python
a = HybridCar()
a.info()
```

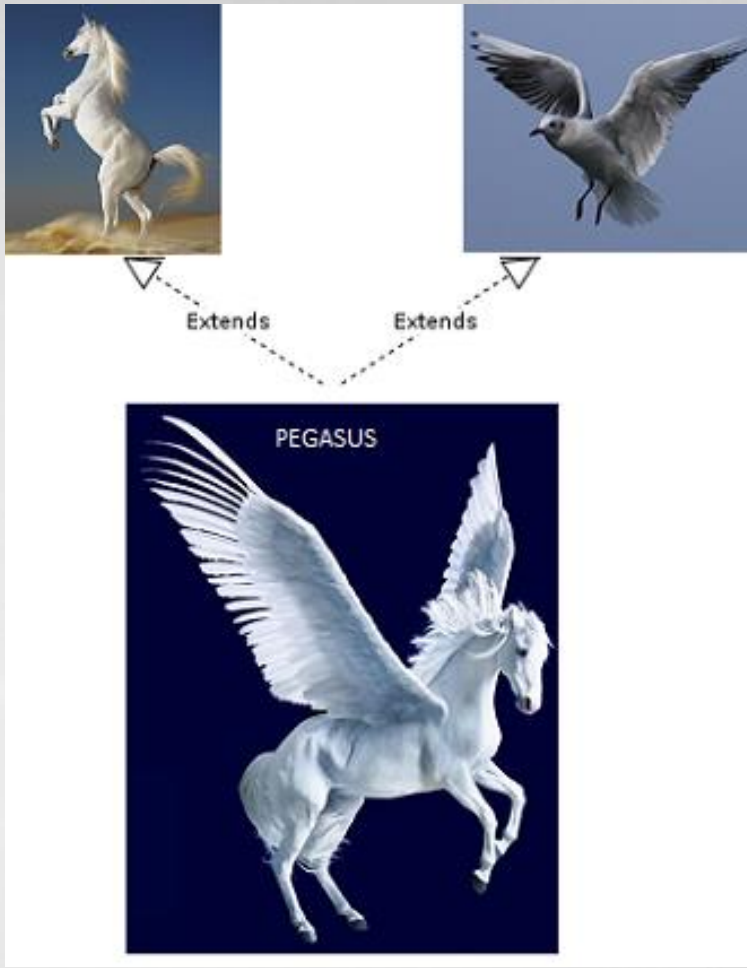
```
I am hybrid
I am manual
I am Electric
I am a car
```

Exercise 2

1. Make class Animal
2. Make class Horse, Seegull as subclasses of Animal
3. Make Pegasus class to inherit from Horse and Seegull

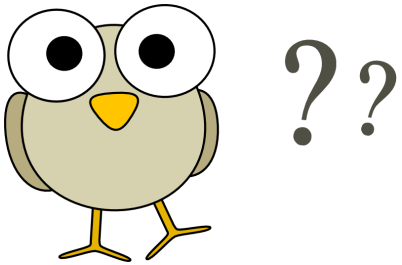
4. All classes have function move()

Horse prints “gallop”
Seegull “fly”
Pegasus should use both super classes Seegull and Horse to print “I can Fly and Hallop”



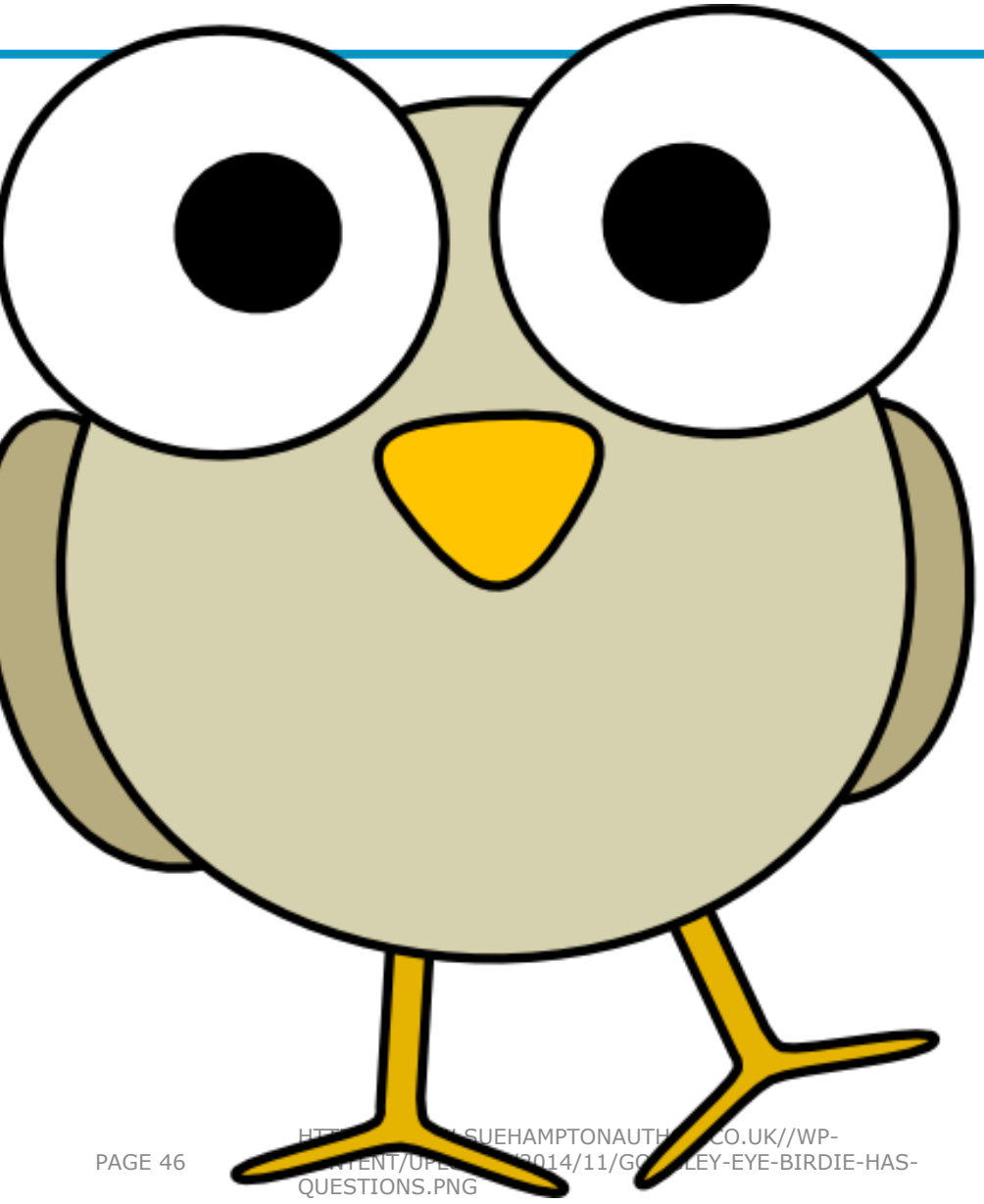
https://i.ytimg.com/vi/Z91tys7dYp4/hqdefault.jpg

TIME FOR!



HTTPS://WWW.SUEHAMPTONAUTHOR.CO.UK//WP-CONTENT/UPLOADS/2014/11/GOOGLEY-EYE-BIRDIE-HAS-QUESTIONS.PNG

TIME FOR! KAHOOT quiz!



HTTPS://SUEHAMPTONAUTHORS.CO.UK//WP-CONTENT/UPLOADS/2014/11/GOOGLEY-EYE-BIRDIE-HAS-QUESTIONS.PNG

Recap today!

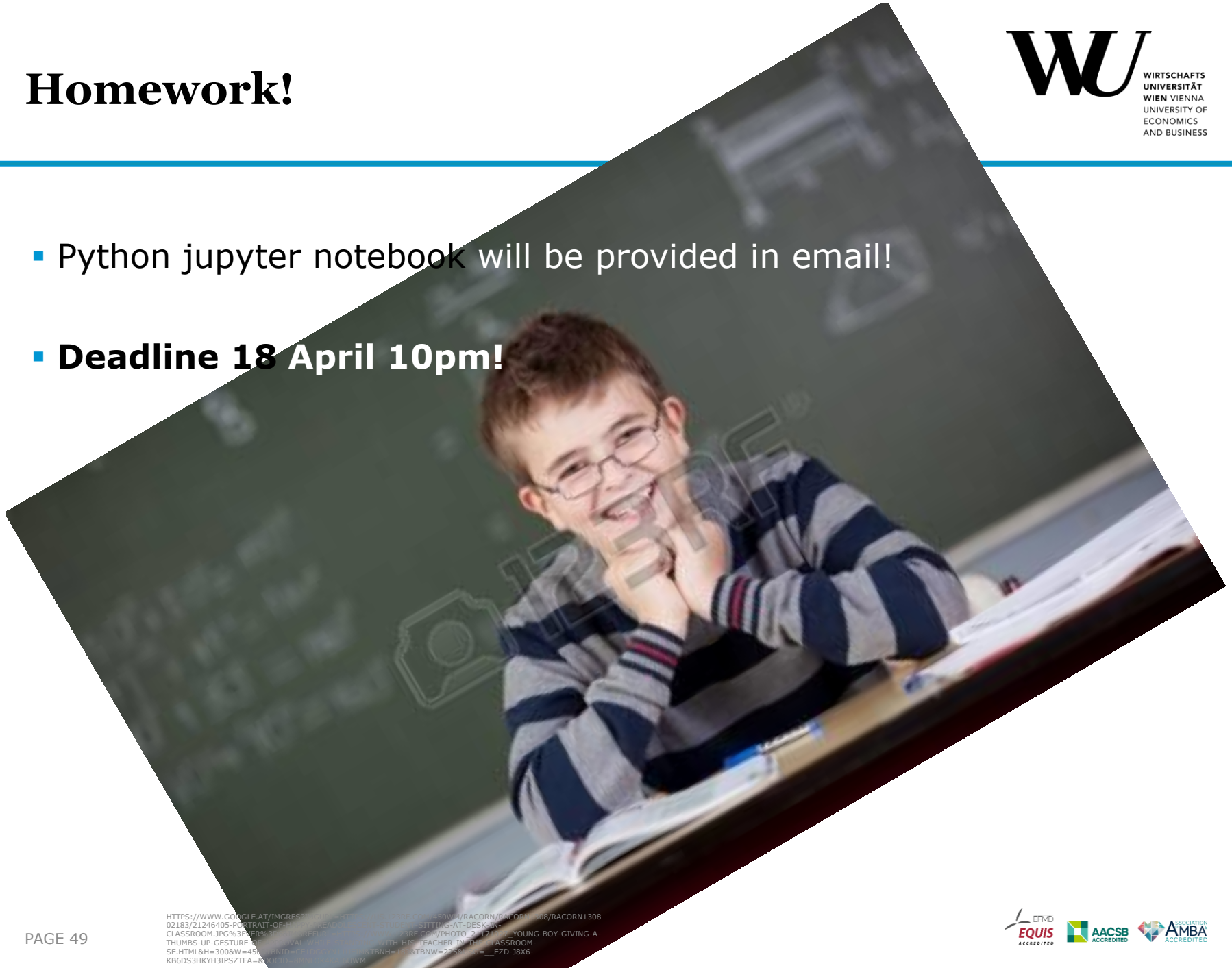
1. Files, read and write
2. Multidimensional array
3. Access modifier (java)
4. Abstract classes, Super constructor
5. Multiple-inheritance problems

Short recap of what has just happened!

1. Hardware – Software (Intro)
2. Basics of programming (Operators, Variables, Functions, Packages)
3. Control flow and lists (if-else, loops (for, while), list)
4. Classes (Objects, Instances, Constructors, Method overloading)
5. Object Oriented Programming (Encapsulation, Abstraction, Inheritance, and Polymorphism)
6. Advanced topics about classes, lists, files

Homework!

- Python jupyter notebook will be provided in email!
- **Deadline 18 April 10pm!**



HTTPS://WWW.GOOGLE.AT/IMGRES?IMGURL=HTTPS://[illegible]123RF.COM/450WM/RACORN/RACORN1308/RACORN130802183/21246405-PORTRAIT-OF-HAPPY-[illegible]-STUDENT-SITTING-AT-DESK-IN-CLASSROOM.JPG%3FVER%3D[illegible]&REFURL=[illegible]123RF.COM/PHOTO_21171[illegible]_YOUNG-BOY-GIVING-A-THUMBS-UP-GESTURE-[illegible]OVAL-WHI[illegible]WITH-HIS-TEACHER-IN-THE-CLASSROOM-SE.HTML&H=300&W=45[illegible]&NID=CE1DG-[illegible]&TBNH=[illegible]&TBNW=27[illegible]&S=__EZD-J8X6-KB6DS3HKYH3IPSZTEA=&[illegible]WM

See you next Thursday!

Wrapping up **classes**!

```python
class BachelorStudent(Student):
    def speak(self):
        return('You guys have no idea.')

some_bsc_student = BachelorStudent()

class Teacher:
    def speak(self):
        return('I am a teacher and i love it!!')

some_teacher = Teacher()

class PhD(Teacher, Student):
    def speak(self):
        return(str(Teacher.speak(Teacher)) +
               ' ' + str(Student.speak(Student)))

some_phd = PhD()
```