WILLIAM JESSUP
UNIVERSITY

CSCI251: Programming II (Online)

Course Syllabus

Course Description

Students will learn object-oriented design; encapsulation and information-hiding, data abstraction; separation of behavior and implementation; classes, subclasses, and inheritance; polymorphism; class hierarchies; practices and design of software development. With this information, students will become overall better programmers by making programs that are robust, easier to change, and easier to keep bug free

Course Outcomes

Upon satisfactory completion of this course, you will be able to:

1. Identify and implement Objects and Classes in Python.
2. Take advantage of Inheritance and Polymorphism.
3. Demonstrate common Design Patterns.
4. Employ techniques for improving Cohesion while reducing Coupling.
5. Apply the knowledge to track down problems using debugging tools

Course Materials

Required Material(s)

Students must purchase:

"Python 3 Object Oriented Programming" Dusty Phillips, Packt Publishing Ltd. ISBN 9781849511261

All other required materials are provided as PDFs or links in Moodle. See the weekly schedule for more complete information on course readings.

Recommended (but not required) Additional Reading

None

Class Policies

You are expected to read all assigned readings, view all lecture videos, screencasts, and access any links posted by the professor. Be prepared to discuss the contents of each.

Attendance and Participation

Because online courses require significant interaction between students, you must upload a current photo of yourself to your Moodle profile. The image should be a headshot with your face clearly visible (no pets, group photos, or cartoons).

You are not required to be online at the same times as your classmates. However, you should check in regularly (to access new materials, submit assignments, and/or participate in ongoing threaded discussions).

Each course week includes a threaded discussion focusing on topics related to the course. The discussions are a great place to ask questions, clarify issues, and share insights. You must check in regularly and contribute to the ongoing conversation, posting on the number of required days.

See the Online Discussion Guidelines for more details.

Any student who has not logged in for course participation during the first week will be administratively dropped along with any subsequent courses in the term.

Note: If you are off campus for any Jessup-sponsored extracurricular activity, you are still required to maintain and follow the due dates outlined in this syllabus. If you have an exceptional instance where internet access is not present either in your transportation and/or accommodations, you will need to have your supervising individual (professor, coach, etc.) inform your instructor to receive additional time on an assignment.

Netiquette

Netiquette, or the rules that surround good communication on the internet, is very important in online courses that are based on high levels of interaction and communication between students and professors at a distance.

Some basic rules to guide you in your online communication (see the Online Student Orientation for a longer, expanded list):

1. **Be thoughtful, kind and courteous in your communication.** Avoid language that may offend others and be cautious when using sarcastic language. In addition, respect your classmates' privacy by not asking them to share more than they would be comfortable doing.
2. **Proofread your writing so it is clear and easy to read.** Avoid acronyms (including text speak), do not use ALL CAPS, and do not overuse exclamation marks (use *italics* for emphasis). Write in short paragraphs and use plenty of white space (extra space between paragraphs) as that makes text easier to read on a webpage.
3. **Engage with your classmates.** Make sure your writing communicates what you intend, ask clear questions of your peers and always be aware of your audience when you are writing in the online classroom.

Written Work Guidelines

Written work is graded for content, organization, style, grammar, and formatting. All papers are to be typed, proofread, spell-checked, double-spaced, and prepared in accordance with APA style and format. Basic formatting should be Times New Roman 12 with 1 inch margins. For help with APA formatting, see the APA Tab of the Student Resources link (located on the main page of the course in Moodle).

The Writing Center is available to all Jessup Online students for help with writing papers as well as APA formatting. You can contact them at writingcenter@jessup.edu or schedule a session through the WJU Student Services Scheduler.

Assignments

Submission Format

All assignments must be submitted as an attachment via Moodle no later than 11:59 PM (PST) the day the assignment is due. Unless otherwise specified, you should submit all papers as Microsoft Word documents (.doc or .docx files) via Moodle. Use the “How to Submit Pages Doc (Mac) to Turnitin” link on Moodle when uploading documents in Mac format.

Late assignments

Whether instructors accept late work or not is up to their discretion.

In the case that they do, late work may be penalized 10% of the possible points for the assignment for each day, or part thereof, that it is late. *Work may not be submitted more than a week late.*

If you face particular difficulty meeting a deadline, please contact the professor ahead of time to discuss any options.

NOTE: The professor is not obliged to accept any late work after the final day of the class session unless prior arrangements have been made.

Feedback and Grades

You can expect to receive written feedback and grades on each weekly assignment via Moodle within 72 hours of the due date for submission.

For larger assignments (research papers, projects, etc.), you can expect to receive feedback within a week.

Academic Integrity

The University Catalog states:

> Academic integrity is an essential component of Christian higher education. Instances of plagiarism will not be treated lightly. If it is a student’s first offense, the paper will simply receive a zero. The student may or may not have the option to re-write the assignment for half credit, according to the instructor’s discretion. If evidence of plagiarism exists a second time the student will receive an academic dismissal, which can be appealed by the student.

Plagiarism includes:

- The intentional or unintentional representation of another’s words or ideas as your own in an academic exercise.
- Using the “copy and paste” method to use text found on a Web site without giving credit to the source.
- Copying information from a source without proper citation and without use of quotation marks or block quotation formatting. If any words or ideas used do not represent your original words or ideas, you must distinguish them with quotation marks or an indented block quotation followed by the appropriate citation.
- Paraphrasing statements or paragraphs without proper citation or using someone else’s ideas, data, language, and/or arguments without acknowledgement.
- Presenting work as your own that has been prepared in whole or part by someone other than you.
- Failure to properly cite statistics, data, or other sources of information in your paper.

- Resubmitting a paper that you have already turned in as an assignment for a different course (including a different section of the same course). While the paper may be considered your original work, resubmitting it is considered a form of plagiarism. Your assignments for every class should be unique and original for that course.

Student Complaints

For complete information about WJU and how to file a complaint as a student please see the Consumer Information section of the Jessup website.

If a distance education student who lives outside the state of California believes that the university's internal procedures have not adequately addressed concerns identified under the Program Integrity Rule, there is a link on the Jessup website with Student Complaint information by State and Agency.

Discussion Forums

Discussion Forums are an integral part of every Jessup Online course. A high percentage of learning in an online environment comes through the dialogue that takes place in Discussion Forums. You should think about the discussion questions in this class as an opportunity for you, your classmates, and your instructor to enter into an interesting conversation about what you are studying. Therefore, you are encouraged to jump into the discussion as often as you'd like. This ensures that everyone will benefit from a variety of opinions and insights on the topics at hand. In other words, your contribution is valuable and important! Since this is a conversation, it's also important that you read the *entire* forum; not only are your contributions important, but you'll find that your classmates' contributions are as well!

NOTE: All Discussion due dates/times are for the Pacific Time Zone.

Substantive Posts

You must post **at least 3 substantive responses** each week. A substantive post is one that contributes something significant to the academic conversation using academic language (avoid "text speak" or other informal language in your discussion posts). To be substantive and earn full credit, a post should:

1. **Be of appropriate length** (initial = 250-400 words; secondary = 125-225 words).
2. **Engage with the course materials** (lecture, texts, videos, etc.) in such a way that it is evident that you have integrated the course content into your thinking.
3. **Demonstrate critical thinking skills.** In other words, your substantive posts should reflect that you have carefully considered the discussion question and have put effort into writing a response that makes a relevant contribution to the conversation.

Requirements

Since discussion questions are usually given a lot of weight in terms of the final course grade, there are also academic expectations. These are as follows.

You must be active in the discussion forum **at least 3 days per week**. This means that you must post a response on 3 of the 7 days each week of the course in order to receive full credit. Do not write all of your forum posts on one day – that eliminates the opportunity for dialogue with classmates.

For weeks with one discussion question:

1. You must post your **initial response** to the question by **Wednesday @ 11:59PM PT**
2. By **Sunday @ 11:59PM PT**, you must post (at minimum) **two secondary posts** (posts responding to your classmates' comments or to your instructor's prompts) for a **total of three posts**. All posts must be substantive to receive full credit.

For weeks with two or more discussion questions:

1. You must post your **initial response to DQ#1 by Wednesday @ 11:59PM PT**
2. You must post your **initial response to DQ#2 by Friday @ 11:59PM PT**
3. By **Sunday @ 11:59PM PT.** you must post (at minimum) **four secondary posts** (spread across both questions; responding to your classmates' comments or to your instructor's prompts) for a **total of six posts**. All posts must be substantive to receive full credit.

Grading (Discussion Questions)

You are encouraged to take part in the weekly dialogue as much as you would like. Your instructor will rate your discussion posts according to the following guidelines:

Initial posts = 0 – 4 points each

- Points can be deducted for posting late (after the stated deadline), and/or for your post not meeting the requirements for being substantive (see above).

Secondary posts = 0 – 3 points each

- Points can be deducted for your post not meeting the requirements for being substantive (see above).
- All secondary posts are due each week by Sunday night @ 11:59 p.m. No credit will be given for late discussion posts after this time.

Each discussion question is worth 10 points [4 pts. for your initial post; 3 pts. for each secondary post]. Therefore, for weeks with **one discussion question**, you can earn up to a total of **10 points**. For weeks with **two discussion questions**, you can earn up to **20 points.**

These totals will be accumulated throughout the week in your gradebook as your instructor rates your posts.
Your **final grade** [0 – 10 or 20] for the entire week will be reflected in your gradebook **no later than Wednesday of the following week.**

Services for Students with Disabilities

In accordance with Section 504 of the Rehabilitation Act and the Americans with Disabilities Act, WJU Disability Support Services office (DSS) provides eligible students with a variety of individualized, reasonable accommodations. These accommodations are intended to assist college students with disabilities in having equal access to regular college programs and activities. Accommodations are determined individually for each student through an interactive process and are based on functional limitations resulting from a documented disability. Recent (within 3 years), verifiable documentation must be provided by a medical doctor or appropriately licensed professional.

Approved accommodations will be provided for students who present instructor with a copy of their Faculty Notification Letter (issued by DSS).

For more information, please visit the Disability Support Services website.

Disability Support Services Contact Information:

WJU Disability Support Services
(916) 577-2253
dss@jessup.edu

Technology Requirements

Sufficient technology tools and Internet access are required when taking a course through Jessup Online. The following list will help ensure that you are adequately equipped.

Supported Operating Systems

- Windows 8 and Windows 10
- MacOS is supported for most online course materials

It is highly recommended that you have administrative rights to the computer used for your coursework. If you must use a computer over which you do not have administrative rights (such as a workplace computer), you may experience difficulties with needed functions, such as installing plug-ins. Check with your workplace IT department to ensure that you may access course materials from your company's network.

Productivity Tools

Microsoft Office (this software is available to students at deeply discounted pricing through Microsoft or JourneyEd.com.).

WJU Email Account

All students are provided with a WJU email address. It should be used for all course communication between you and your instructor. This will avoid issues with Spam blockers and other problems that may prevent you from receiving email from your instructors. Use of this email account will also enable you to participate in special student offers that are available only to students with an "edu" email address. You can access your Jessup e-mail account at my.jessup.edu.

Supported Browsers

- Google Chrome
- Mozilla Firefox

Browser Settings

Please refer to your browser's Help features to check these settings.

- Pop-Up Blocker should be disabled
- JavaScript should be enabled
- Java should be enabled
- Cookies should be enabled

Plug-ins

The most recent version of the following plug-ins is required for many of the resources available in your online courses:

- Adobe Acrobat Reader
- Apple QuickTime Player
- Java SE 8 or higher

All plug-ins needed to participate in components of your online classes are available at no additional cost. It is recommended that you review the list of plug-ins and install them prior to beginning your coursework.

Screen Settings

Screen resolution (size) should be set at minimum 1024 x 768 or higher.

HelpDesk

There is a link on every Moodle page for 24/7 technical support through an outside vendor.

You can also contact the Jessup HelpDesk (which is not 24/7) is through WJU. Email helpdesk@jessup.edu or call 916.577.2345.

Course Grading Explanations

Points	Grade
90-100	A
80-89	B
70-79	C
60-69	D
<59	F

A = **Excellent performance**. Work is truly exemplary and worthy of emulation by others. Student exceeds expectations and constructively contributes to the learning environment.

B = **Above average performance**. All assignments are complete and on time and exhibit a complete understanding and an ability to effectively apply concepts.

C = **Average performance**. Student accomplishes only the minimum requirements or does not complete all requirements. Oral and written communication is at an acceptable level for a college student.

D = **Work is below acceptable level for a college student**. Student shows only a very basic understanding of the material or does not meet all assignment requirements.

F = **Work is not passing**. Student's work is incomplete or does not apply information and concepts in a satisfactory

Final Grade Calculation

Assignments		***Value***
Discussion Questions		**30%**
Weekly Assignments		**40%**
Midterm Exam		**15%**
Final Paper		**15%**
	TOTAL:	100%

Course Outline

Week 1	***Details***	***Due***	***Demand Hours***	***Course Objective***
Topics and Learning Objectives	**Topics** ● Course Introduction ● Introduction to Debugging ● Overview of Object-Oriented Programming ● Objects and Modules in Pytho**n** **Learning Outcomes** ● Debug Python code in Jupyter Notebook and VS Code ● Summarize the basics of object-oriented programming ● Practice with Tkinter ● Adapt Python class syntax ● Compare and contrast a class definition and a class instance ● Explain aliasing (references/pointers) ● Use the Python standard for protected and private methods of name mangling ● Create a Python module ● Configure a Module to be able to be run independently ● Import modules by many different methods			
Setup	**Install Tkinter** ● https://wiki.python.org/moin/TkInter ● Do the online "Hello World" tutorial for Tkinter ○ https://realpython.com/python-gui-tkinter/ ● Submit a screenshot of your Tkinter program displaying "Hello World" ● Contact your instructor or IT helpdesk if you have problems with installation ● Google how to take a screenshot and save it if you don't know how	**Sunday 11:59 PM PT**	**1.5 hours**	
Reading Assignments	**Read "Python 3 Object Oriented Programming"** ● Chapter 1 (pages 7 – 23) 16 pages ○ Quiz will be on this reading ● Chapter 2 Section (pages 33 – 40, 43 – 52) 16 pages		**3.0 hours**	

Video Resources	**View** ● Programming is Hard: Introduction ● Course Introduction – roadmap ● How to Debug Code in Jupyter Notebook ● How to Debug Code in VSCode or Visual Studio ● What is Object-oriented Programming ● Class Syntax ● Class Concepts ● Creating and Using Modules **View Demo** ● Demo: “Hello World” with a class ● Demo: “Hello World” with a module		**1.5 hours**	
Discussion	**Discuss** ● **DQ #1: Public, Private, Protected** ○ Give concrete specific examples for cases (not mentioned in class materials) where you would want a class method to be public, private and protected. Explain your choice. ● **DQ #2: Class, Module** ○ Give concrete specific examples for cases (not mentioned in class materials) where you would want your code to be a part of a class, module and neither. Explain why would you want this?	See *Discussion Guidelines*	**6.0 hours**	
Quiz #1 (Pass/Fail)	**Complete** ● Programmer Academic Integrity Quiz ○ 4 - 6 Questions	**Complete/ Incomplete** **Sunday 11:59 PM PT**	**0.5 hour**	
Assignment #1	**Complete** ● Questions on Reading Assignment ○ 14 Questions ○ Multiple choice	**Sunday 11:59 PM PT**	**1.5 hours**	

Assignment #2	**Debugging Challenge** ● Use debugging to solve the problems with the code sample ● Submit the corrected code in Moodle ● Work in Visual Studio Code, Visual Studio, or Jupyter Notebook and submit the file to Moodle.	**Sunday 11:59 PM PT**	**1.5 hours**	
Assignment #3	**Code** ● Create a ComplexNumber class ○ DO NOT make use of Python's built in complex numbers system in any way ○ **Think of this class you are making as a new type of variable you are making** ○ It must store the real and imaginary numbers as floats. ○ Methods to provide: add, subtract, multiply, and divide ○ Do not put anything in the methods other than the calculation and returning the result. For example you better not print anything from within the methods. ○ The returned values should be another instance of ComplexNumber ● Work in Visual Studio Code, Visual Studio, or Jupyter Notebook and submit the file to Moodle.	**Sunday 11:59 PM PT**	**4.0 hours**	
		TOTAL HOURS FOR THE WEEK:	**19.0 hours**	

Week 2	***Details***	***Due***	***Demand Hours***	***Course Objective***

Topics and Learning Objectives	**Topics** ● Inheritance and Polymorphism ● Programming Styles ● Extending Built-ins **Learning Outcomes** ● Implement subclasses ● Override methods ● Describe multiple inheritance ● Use Polymorphism ● Code using each of the different programming styles ● Extend a class with built-in functions ● Overload math operators, Boolean operators, math functions, and casting functions			
Reading Assignments	**Read "Python 3 Object Oriented Programming"** ● Chapter 3 (pages 63 – 80) 17 pages ● Chapter 5 Section (pages 125 – 141) 16 pages ● Chapter 6 "Extending Built-ins" (pages 177-182) 5 pages		**4.0 hours**	
Video Resources	**View** ● Inheritance syntax ● Polymorphism ● Programming Styles ● Math Operator Overloading ● Other Operator Overloading ● Operator Overloading to Simulate Lists ● Operator Overloading with List Iterators **View Demo** ● Create Tkinter class wrappers for Button and Label ● Create a Tkinter class wrapper for the Frame ● Extend some built-ins for our Tkinter wrapper classes ● Extend list built-ins for our Tkinter Frame class		**2.0 hours**	
Discussion	**Discuss** ● **DQ #1: Inheritance** ○ Give concrete specific examples for cases (not mentioned in class materials) where we would want use inheritance and multiple inheritance. Explain your choice. ● **DQ #2: Polymorphism** ○ Give concrete specific examples for cases (not mentioned in class materials) where we would want to use polymorphism. Explain your choice.	See *Discussion Guidelines*	**6.0 hours**	

Assignment #4	**Code** ● Clock assignment ● Write a class inheritance hierarchy of clocks ○ Clock ▪ Sundial ▪ Mechanical ● Cuckoo ● Grandfather ▪ Digital ▪ Atomic ● Give them a function to display the time making proper use of inheritance ● Give them a function to print out the type of clock ● You will be graded on your ability to understand not just the syntax of inheritance but how to do good programming with inheritance by avoiding adding more code than needed, and by making it easy to make changes in the future. ● Work in Visual Studio Code, Visual Studio, or Jupyter Notebook and submit the file to Moodle.	**Sunday 11:59 PM PT**	**1.5 hours**	
Assignment #5	● Create a special list class that ensures that its internal list is sorted ● It is a wrapper for a list ● Do not use the built-in python sorting functions ● The list starts out empty ● When something is added to the list, insert it into the correct sorting position ● Implement the following methods ○ __getitem__ ○ __setitem__ ○ __iter__ ○ __reversed__ ○ __next__ ○ __len__ ○ __contains__ ○ __delitem__ ○ __str__ ● Work in Visual Studio Code, Visual Studio, or Jupyter Notebook and submit the file to Moodle.		**6.0 Hours**	
		TOTAL HOURS FOR THE WEEK:	**19.5 hours**	

Week 3	***Details***	***Due***	***Demand Hours***	***Course Objective***

Topics and Learning Objectives	**Topics** ● Comprehensions ● Generators ● Alternative Overloading ● Enums ● Named Tuples ● Metaclass ● Abstract Base Class ● Built-in Decorators **Learning Outcomes** ● Demonstrate Comprehensions ● Implement Generators ● Code default function arguments ● Restrict argument variable types ● Use a variable length argument list ● Pack and unpack arguments ● Pass a function as a variable ● Override a method with monkey-patching ● Make a class instance callable ● Implement Enums ● Use Named Tuples ● Review the Metaclass ● Make an Abstract Base Class ● Code with the Python built-in decorators			
Reading Assignments	**Read "Python 3 Object Oriented Programming"** ● Chapter 7 "Comprehensions" (pages 197-203) 6 pages ● Chapter 7 "Generators" (pages 203-205) 2 pages ● Chapter 7 "An alternative method to overloading" (pages 205-213) 8 pages ● Chapter 7 "Functions are objects too" (pages 213-220) 7 pages ●		**2.5 hours**	
Reading Assignments	**Skim Read** ● https://docs.python.org/3/library/enum.html 26 pages ● https://docs.python.org/3/library/abc.html 8 pages		**1.5 hours**	

Video Resources	**View** ● Inline “if” statement and “for” loop (comprehension) ● Generators ● Alternative input to methods ● Tricks with Functions ● Working with Enums ● Named Tuples ● Overview of a metaclass ● Abstract base class ● Python built-in decorators **View Demo** ● Add to the Frame class, methods to find widgets by name ● Add to the Button wrapper an option to make another button that is a copy of the current one ● Add a function to the base that returns the type of widget in the form of an Enum ● Make the widget base class into an Abstract Base Class		**3.0 hours**	
Discussion	**Discuss** ● **DQ #1: Yield** ○ Give concrete specific examples for cases (not mentioned in class materials) where we would want use the “yield” statement. Explain your choice. ● **DQ #2: The way you do things** ○ Can the way you do something in code matter as long as you always get the same result? How can it matter? Explain.	See *Discussion Guidelines*	**6.0 hours**	
Assignment #6	**Code** ● Write a function called AddAll that takes an unknown number of parameters ● It will add all the values together and return the result ● It will error if the inputted values are not all integers or all floats. It can also take exactly one argument that is a function pointer. ● If the argument is a function pointer, it calls that function multiple times to get each value to add. ● Write code to test all generic cases of calls to AddAll. ● Work in Visual Studio Code, Visual Studio, or Jupyter Notebook and submit the file to Moodle.	**Sunday 11:59 PM PT**	**2.5 hours**	

Assignment #7	**Code** ● In your code, create an enum with 5 options. ● Then create a Tuple with 5 strings. ● Each string is a message to show the user the response to each enum the user entered. ● Take text input from the user that must be one of the enums. ● Find the variable integer to the enum that the user entered. ● Then use the variable to do a lookup in the tuple in order to output the appropriate response. ● Work in Visual Studio Code, Visual Studio, or Jupyter Notebook and submit the file to Moodle.	**Sunday 11:59 PM PT**	**2.5 hours**	
		TOTAL HOURS FOR THE WEEK:	**18.0 hours**	

Week 4	*Details*	*Due*	*Demand Hours*	*Course Objective*
Topics and Learning Objectives	**Topics** ● Midterm ● Design Patterns Overview ● Design Pattern: Decorator ● Design Pattern: Observer **Learning Outcomes** ● Review material learned so far ● Take the Midterm ● Define what a design pattern is ● Code a decorator ● Show an observer pattern ● Explain an event aggregator ● Adapt a mediator pattern			
Reading Assignments	**Read "Python 3 Object Oriented Programming"** ● Chapter 8 "Design Patterns" (pages 227-229) 2 pages) ● Chapter 8 "Decorator Pattern" (pages 229-235) 6 pages ● Chapter 8 "Observer Pattern" (pages 235-237) 2 pages		**1.0 hour**	

Video Resources	**View** ● Decorator Pattern Overview ● Decorator Pattern Examples ● Observer Pattern Overview ● Observer Pattern Example ● Event Aggregator Example ● Mediator Pattern Example **View Demo** ● Add an optional logging feature to the creation of our widget classes ● Make an event observer for the ‘enter’ key		**1.5 hours**	
Discussion	**Discuss** ● **DQ #1: Observer Pattern** ○ Give concrete specific examples for cases (not mentioned in class materials) where we would want use an observer pattern. In these cases would it be best to use polling or callbacks? Explain your choice. ● **DQ #2: Event Aggregator and Mediator Pattern** ○ From your examples in question #1, what would happen if we instead used an event aggregator to do the same things? What about using a mediator instead? Which would be the optimal solution? Why?	See *Discussion Guidelines*	**6.0 hours**	
Midterm Exam	**Code** ● One coding challenge will be provided ● You have **1.5 hours** to complete it ● Work in Visual Studio Code, Visual Studio, or Jupyter Notebook and submit the file to Moodle.	**Sunday 11:59 PM PT**	**8.0 hours**	
Assignment #8	**Code** ● Write a Tic Tac Toe game played by only AI and only with random picks. ● Detect tie or win. ● Write a decorator to detect how fast a single game is played given a seed for random generation. ● Create another decorator to print out the Tic Tac Toe board at each pick so you can see what the AI is doing. ● Work in Visual Studio Code, Visual Studio, or Jupyter Notebook and submit the file to Moodle.	**Sunday 11:59 PM PT**	**3.5 hours**	
		TOTAL HOURS FOR THE WEEK:	**20.0 hours**	

Week 5	*Details*	*Due*	*Demand Hours*	*Course Objective*
Topics and Learning Objectives	**Topics** ● Design Pattern: Strategy ● Design Pattern: State ● Design Pattern: Template ● Design Pattern: Singleton **Learning Outcomes** ● Apply the Strategy pattern ● Code using a State pattern ● Explain the Template Pattern ● Implement a Singleton			
Reading Assignments	**Read "Python 3 Object Oriented Programming"** ● Chapter 8 "Strategy Pattern" (pages 238-240) 2 pages ● Chapter 8 "State Pattern" (pages 241-246) 5 pages ● Chapter 8 "Template Pattern" (pages 251-255) 4 pages ● Chapter 8 "Singleton Pattern" (pages 247-251) 4 pages		**1.5 hour**	
Video Resources	**View** ● Strategy Pattern Overview ● Strategy Pattern Example ● State Pattern Overview ● State Pattern Example ● Template Pattern Overview ● Template Pattern Example ● Singleton Pattern Overview ● Singleton Pattern Example **View Demo** ● Modify our Label class to be able to take in multiple text strings each for a different language ● Make a menu and options frames ● Make a custom button pair class ● Make a root Frame class that is a singleton		**2.5 Hours**	

Discussion	**Discuss** ● **DQ #1: State vs Strategy** ○ Give a concrete specific example for a case (not mentioned in class materials) where we would want use a strategy pattern. Why not use a state pattern for this instead? Explain ● **DQ #2: Template vs Strategy** ○ Give a concrete specific example for a case (not mentioned in class materials) where we would want use a template pattern. Why not use a strategy pattern for this instead? Explain	See *Discussion Guidelines*	**6.0 hours**	
Assignment #9	**Code** ● Write a program that first asks the user which language they speak: English or Pig Latin. ○ The rest of the dialog must be in that language. ○ Read online about how to convert English to Pig Latin and vice-versa ○ DO NOT do any conversion functions between English and Pig Latin. Instead write and hardcode the messages in each language yourself. ● Then give an introduction and ask them their name and respond with a hello message in the correct language. ● Do all this properly according to the principles of the Strategy Pattern. ● Work in Visual Studio Code, Visual Studio, or Jupyter Notebook and submit the file to Moodle.	**Sunday 11:59 PM PT**	**2.5 hours**	
Assignment #10	**Code** ● Write a game for taking care of your pet puppy. ● Show the state that your puppy is in: sleeping, idle, greeting, wanting attention, playing, hungry, eating, wanting to potty, pottying, barking, fighting. ● The user can input what to do with the puppy: ignore, pet, play, feed, take on walk, etc. ● Each user action has an effect on the state of the puppy depending on the state the puppy was in when the action was taken. ● (You are free to add additional states and user actions.) ● Work in Visual Studio Code, Visual Studio, or Jupyter Notebook and submit the file to Moodle.	**Sunday 11:59 PM PT**	**3.5 hours**	

Assignment #11	**Code** ● Write a class that prints a date and time according to 3 different formats. ● Use the template pattern to break down the printing of each of these components and their separators: month, day, year, hour, minute. ● Work in Visual Studio Code, Visual Studio, or Jupyter Notebook and submit the file to Moodle.	**Sunday 11:59 PM PT**	**2.5 hours**	
		TOTAL HOURS FOR THE WEEK:	**18.5 hours**	

Week 6	***Details***	***Due***	***Demand Hours***	***Course Objective***
Topics and Learning Objectives	**Topics** ● Design Pattern: Adapter ● Design Pattern: Facade ● Design Pattern: Flyweight ● Design Pattern: Proxy ● Design Pattern: Command **Learning Outcomes** ● Make an Adapter ● Create a Facade ● Describe a Flyweight pattern ● Distinguish between Flyweight and Proxy patterns ● Use a Command pattern			
Reading Assignments	**Read "Python 3 Object Oriented Programming"** ● Chapter 9 "Adapter Pattern" (pages 257-260) 3 pages ● Chapter 9 "Facade Pattern" (pages 260-262) 2 pages ● Chapter 9 "Flyweight Pattern" (pages 263-266) 3 pages ● Chapter 9 "Command Pattern" (pages 267-271) 4 pages		**1.5 hour**	

Video Resources	**View** ● Adapter Pattern Overview ● Adapter Pattern Example ● Facade Pattern Overview ● Facade Pattern Example ● Flyweight & Proxy Pattern Overviews ● Flyweight Pattern Example ● Command Pattern Overview ● Command Pattern Example **View Demo** ● Create a class wrapper for the RadioButton widget ● Create a class wrapper for the Canvas widget, but only for use in displaying a single image ● Use the flyweight pattern for Canvas class images ● Make buttons to send a command message to a Button, Label, Canvas, or Frame to change text or image		**2.5 hours**	
Discussion	**Discuss** ● **DQ #1: Flyweight vs Proxy** ○ You have seen code examples of the flyweight pattern. The proxy pattern is quite similar. Based on the description of the differences in purpose in these two patterns, what are the most significant code differences you would expect between the two? Why do you think so? ● **DQ #2: Façade Pattern** ○ Name a Python module that you are familiar with. What changes to the interface would you perform to make that module "better"? How would you go about creating a façade to do just that? (No need to provide code. Just explain your approach.)	See *Discussion Guidelines*	**6.0 hours**	
Assignment #12	**Code** ● Write adapters for math.sin(), math.cos() that take degrees instead of radians. ● Write adapters for math.asin(), math.acos() that return degrees instead of radians. ● Work in Visual Studio Code, Visual Studio, or Jupyter Notebook and submit the file to Moodle.	**Sunday 11:59 PM PT**	**1.5 hours**	

Assignment #13	**Code** ● Some applications keep all their strings in a common resource in order to save memory by avoiding duplicate strings. ● Take string input from the user in a loop. ● First take the string tag, then the string value. ● Each string is stored in a dictionary. ● If an input of a string is already in the dictionary, don’t add it but give an error saying it already exists and show what tag it has. ● If the tag is the same then change the string value of that tag. ● At the end, print out the string list. ● Work in Visual Studio Code, Visual Studio, or Jupyter Notebook and submit the file to Moodle.	**Sunday 11:59 PM PT**	**2.5 hours**	
Assignment #14	**Code** ● Create a text based calculator that allows you to add, subtract, multiply, and divide two numbers. ● The first time, the user enters 2 numbers, but after that the user enters only the second number because the result of the previous calculation will be the new first number. ● Give the user the option to “undo” the previous calculation and keep undoing until the first calculation is undone. ● DO NOT store results of previous calculations. Instead store previous operations and numbers that were used thus making this assignment follow the command pattern. ● Work in Visual Studio Code, Visual Studio, or Jupyter Notebook and submit the file to Moodle.	**Sunday 11:59 PM PT**	**5.5 hours**	
		TOTAL HOURS FOR THE WEEK:	**19.5 hours**	

Week 7	***Details***	***Due***	***Demand Hours***	***Course Objective***

Topics and Learning Objectives	**Topics** ● Design Pattern: Factory ● Design Pattern: Abstract Factory ● Design Pattern: Composite ● Final **Learning Outcomes** ● Implement a Factory ● Explain an Abstract Factory ● Code using a Composite pattern ● Review design patterns ● Take the Final			
Reading Assignments	**Read "Python 3 Object Oriented Programming"** ● Chapter 9 "Abstract Factory Pattern" (pages 271-275) 4 pages ● Chapter 9 "Composite Pattern" (pages 276-280) 4 pages		**1.0 hour**	
Video Resources	**View** ● Factory & Abstract Factory Pattern Overview ● Factory Pattern Example ● Composite Pattern Overview ● Composite Pattern Example **View Demo** ● Make a Factory to create our UI classes based on input string ● Add the option to copy a Frame and the other classes as we did with the Button class		**1.5 hours**	
Discussion	**Discuss** ● **DQ #1: Abstract Factory** ○ Give specific examples for cases (not mentioned in class materials) where we would want use an abstract factory pattern. Explain your choice ● **DQ #2: Composite Pattern** ○ Give specific examples for cases (not mentioned in class materials) where we would want use a composite pattern. Explain your choice.	See *Discussion Guidelines*	**6.0 hours**	

Assignment #15	**Code** ● Make a Factory that will return either the ComplexNumber class that we made earlier or a new Number class that is a parent of ComplexNumber. ● The input to the factory will be a string that is either a regular number or a complex number. ● It returns a Number if the input was a regular number, but a ComplexNumber if the input was a complex number. ● Work in Visual Studio Code, Visual Studio, or Jupyter Notebook and submit the file to Moodle.	**Sunday 11:59 PM PT**	**2.5 hours**	
Final Exam	**Multiple Choice** ● 11 multiple choice questions about design patterns ● Submit your multiple choice answers in a regular text document to Moodle. **Code** ● Four coding challenges will be provided to test your understanding of design patterns ● Do the code parts in Visual Studio Code, Visual Studio, or Jupyter Notebook and submit the file to Moodle. **You have 2 hours total to complete the multiple choice section and all four coding challenges.**	**Sunday 11:59 PM PT**	**8.5 hours**	
		TOTAL HOURS FOR THE WEEK:	**19.5 hours**	