Al al-Bayt University

Prince Hussein Bin Abdullah College for Information Technology
Computer Science Department

0901212
Python Programming

1st Semester 2014/2015

Course Catalog
This course introduces the student to the Python language. The course provides insight into the features of Python such as lists, functions, working with files, dictionaries and sets, errors and exception handlings, using modules, GUI, that make it an excellent choice for developing projects of any size.

Textbook(s)	
Title	How to Think Like a Computer Scientist (Learning with Python)
Author(s)	Allen Downey, Jeffrey Elkner and Chris Meyers
Edition	2nd
Publisher	Green Tea Press (Wellesley, Massachusetts)
Year	2008
Number of copies in university library	0

References	
Books	• *Core Python Programming* by Wesley Chun is a large book at about 750 pages. The first part of the book covers the basic Python language features. The second part provides an easy-paced introduction to more advanced topics including many of those mentioned above. • *Python Essential Reference* by David M. Beazley is a small book, but it is packed with information both on the language itself and the modules in the standard library. It is also very well indexed. • *Python Pocket Reference* by Mark Lutz really does fit in your pocket. Although not as extensive as Python Essential Reference it is a handy reference for the most commonly used functions and modules. Mark Lutz is also the author of Programming Python, one of the earliest (and largest) books on Python and not aimed at the beginning programmer. His later book Learning Python is smaller and more accessible.
Internet Links	• The Python home page at *www.python.org* is the place to start your search for any

	Python related material. You will find help, documentation, links to other sites and SIG (Special Interest Group) mailing lists that you can join. • The Open Book Project *www.ibiblio.com/obp* contains not only this book online but also similar books for Java and C++ by Allen Downey. In addition there are Lessons in Electric Circuits by Tony R. Kuphaldt, Getting down with ..., a set of tutorials on a range of computer science topics, written and edited by high school students, Python for Fun, a set of case studies in Python by Chris Meyers, and The Linux Cookbook by Michael Stultz, with 300 pages of tips and techniques. • Finally if you go to Google and use the search string "python-Snake-Monty" you will get about 750,000 hits.

Instructors	
Coordinator	Dr. Saad Bani-Mohammad
Office Location	Computer Science Department (Vice Dean of IT College)
Office Phone	0096 2 2 6297000 ext. 3390
Email	bani@aabu.edu.jo

Class Schedule and Locations
Section 1: **Lecture Times:** Sunday, Tuesday and Thursday: 10-11. **Location: CS** Department Hall

Office Hours
Dr. Saad Bani-Mohammad: Sunday and Thursday: 11-12, 13-14

Teaching Assistants

Names of lab instructors	Day	Time	Section No.	Lab No.
Areej Shatat	Tuesday	14:00-16:00	1	IT 4

Course Objectives	**Assessment Method**
Objective 1: Familiarization with Python's Idle programming environment, working with values, variables, expressions and statements, programs as a sequence of statements, input-process-output program style, solving very small problems, handling errors in programs.	Exams and Lab Assignment #1.
Objective 2: Writing programs that make use of conditional execution, writing functions that return values.	Exams and Lab Assignment #2.
Objective 3: Understanding and writing recursive functions.	Exams and Lab Assignment #3.
Objective 4: Writing programs that make use of	Exams and Lab Assignment #4.

conditional and iterative execution.	
Objective 5: Practice with drawing shapes (computer graphics and animation).	Exams and Lab Assignment #5.
Objective 6: Practice with string operations, solving problems using string manipulation.	Exams and Lab Assignment #6.
Objective 7: Practice with creating and manipulating lists, solving problems requiring lists.	Exams and Lab Assignment #7.
Objective 8: Working with nested lists (lists of lists).	Exams and Lab Assignment #8.
Objective 9: Learning about and working with dictionaries.	Exams and Lab Assignment #9.
Objective 10: Introduce event driven Graphical User Interface (GUI) programming.	Exams.
Objective 11: Further work with file processing, dictionaries, and problem-solving.	Exams and Lab Assignment #10.
Objective 12: Writing programs that make use of object oriented concepts (Classes and objects, Classes and functions, Classes and methods, Overloading, Overriding, and Inheritance).	Exams.

Prerequisites	
Prerequisites by course:	Object-Oriented Programming (0901210)

Topics Covered		
Chapter 1: The Way of the Program		
Topic	**Chapter(s) in Text**	**Week(s)**
• The way of the program. • Basic concepts: program, interpreter, compiler, programming languages, solving a problem. • What is debugging? • Program errors: syntax, semantic and runtime errors. • Experimental Debugging. • Formal and natural languages. • The first program.	1	1
Chapter 2: Variables, expressions, and statements		
Topic	**Chapter(s) in Text**	**Week(s)**
• Values and Types. • Variables. • Variable Names and Keywords. • Python Statements. • Evaluating Expressions. • Operators and Operands. • Order of Operations.	2	2

• Operations on Strings. • Composition. • Comments.		
Chapter 3: Functions		
Topic	**Chapter(s) in Text**	**Week(s)**
• Function Calls. • Type Conversion. • Type Coercion. • Math Functions. • Composition. • Adding New Functions. • Function Definitions and Use. • Flow of Execution. • Parameters and Arguments. • Variables and Parameters are Local. • Stack Diagrams. • Functions with Results.	3	3
Chapter 4: Conditionals and Recursion		
Topic	**Chapter(s) in Text**	**Week(s)**
• The Modulus Operator. • Boolean Expressions. • Logical Operators. • Conditional Execution. • Alternative Execution. • Chained Conditionals. • Nested Conditionals. • The *return* Statement. • Recursion. • Stack Diagrams for Recursive Functions. • Infinite Recursion. • Keyboard Input.	4	4
Chapter 5: Fruitful Functions		
Topic	**Chapter(s) in Text**	**Week(s)**
• Return Values. • Program Development. • Composition. • Boolean Functions. • More Recursion. • Checking Types.	5	5
Chapter 6: Iteration		
Topic	**Chapter(s) in Text**	**Week(s)**
• Multiple Assignments. • The while Statement. • The while Statement (Drawing Iteratively). • The while Statement (Tables). • The while Statement (2D Tables). • Encapsulation and Generalization. • Local Variables. • More Generalization.	6	6
Chapter 7: Strings		
Topic	**Chapter(s) in Text**	**Week(s)**

• A compound data type. • String Length. • Traversal and the *for* Loop. • String Slices. • String Comparison. • Strings are Immutable. • *find* Function. • Looping and Counting. • The string Module. • Character Classification.	7	7
Chapter 8: Lists		
Topic	**Chapter(s) in Text**	**Week(s)**
• Lists. • List Values. • Accessing Elements. • List Length. • List Membership. • List and *for* Loops. • List Operations. • List Slices. • Lists are Mutable. • Lists Deletion. • Objects and Values. • Aliasing. • Cloning Lists. • List Parameters. • Nested Lists. • Matrices. • Strings and Lists. • Drawing shapes.	8	8
Chapter 9: Tuples		
Topic	**Chapter(s) in Text**	**Week(s)**
• Mutability and Tuples. • Tuple Assignment. • Tuples as return Values. • Random Numbers. • List of Random Numbers. • Counting. • Many Buckets.	9	9
Chapter 10: Dictionaries		
Topic	**Chapter(s) in Text**	**Week(s)**
• Dictionaries. • Dictionary Operations. • Dictionary Methods. • Aliasing and Copying. • Sparse Matrices. • Hints. • Long Integers. • Counting Letters. • Aside.	10	10
Chapter 11: Files and Exceptions		

Topic	Chapter(s) in Text	Week(s)
• Files and Exceptions. • Text Files. • Writing Variables. • Directories. • Pickling. • Exceptions.	11	11
Chapter 12: GUI Programming		
Topic	**Chapter(s) in Text**	**Week(s)**

• Graphical User Interfaces.
• The main ideas.
• The simplest GUI program in Python
• Event-driven programming.
• Terminology.
• Changing the layout.
• Getting input from the user.
• GUI Examples:
  • Designing a GUI.
  • Setting up the window and widgets.
  • A variable for the Entry widget.
  • A callback for the Check button.
  • Defining the check function.
  • Improving the output.
  • Stylistic points.
	Instructor Notes	12

Chapter 13: Classes and Objects		
Topic	**Chapter(s) in Text**	**Week(s)**
• User-defined Compound Types. • Attributes. • Instances as Parameters. • Sameness. • Rectangles. • Instances as return values. • Objects are Immutable. • Copying.	12	13
Chapter 14: Classes and Functions		
Topic	**Chapter(s) in Text**	**Week(s)**
• Time. • Pure Functions. • Modifiers.	13	14
Chapter 15: Classes and Methods		
Topic	**Chapter(s) in Text**	**Week(s)**
• Object-Oriented Features. • *Print*Time. • Object Oriented Examples. • Optional Arguments. • The initialization method. • Points revisited. • Operator overloading. • Polymorphism.	14	15
Chapter 16: Inheritance		
Topic	**Chapter(s) in Text**	**Week(s)**

• A simple class def: student. • Creating and Deleting Instances. • Instantiating Objects. • Constructor: __init__. • Self. • Deleting instances: No Need to “free”. • Access to Attributes and Methods. • Definition of student. • Traditional Syntax for Access. • Accessing unknown members. • getattr(object_instance, string). • hasattr(object_instance,string). • Attributes: Two Kinds of Attributes • Data Attributes. • Class Attributes. • Data vs. Class Attributes. • Subclasses. • Redefining Methods. • Definition of a class extending student. • Extending __init__. • Built-In Members of Classes. • Special Methods. • Special Methods – Example. • Special Data Items. • Special Data Items – Example. • Private Data and Methods.	Instructor Notes	16

Course Outcomes:
1. The students will be able to develop their own programs in Python, based on a simple problem description.
2. The students will in particular be trained in using the computer to solve problems from their own life and visualize the solutions (design GUI for their solutions).
3. Students' experience in this course will put them in a good position to use the computer to solve exercises in other university courses and to exploit this experience to develop their graduation projects.
4. The students will understand the built-in objects of Python.
5. The students will understand the GUI programming.
6. The students will be able to deal with drawing and animation.
7. The students will be able to deal with lists, dictionaries, tuples and files in python.

Evaluation		
Assessment Tool	**Expected Due Date**	**Weight**
Test 1	6^{th} week	15%
Test 2	12^{th} week	15%
Lab	Weekly	20 %
Final Exam	TBA	50 %

Policies	
Attendance	It is strongly recommended that students attend all lectures. Also, university regulations mandate that students may not miss more than 12.5% of classes without valid excuses. In all cases, they may not miss more than 25% of classes. Should they do, they will be not be allowed to take course exams.
Homework/Lab	Students are expected to attend lab sessions and submit assignments on time.
Exams	Exams will be close-book. Exam dates will be announced later according to departmental and university schedules.
Plagiarism	You should not copy other people's work and claim it is yours. Detected plagiarism will be dealt with as per university regulations.

Last updated : 25/09/2014

I'm thinking I need to modify the for loop so that if the output character count is less than the original string value inputted, it adds the letter 'U'. How do I go about relating the length of the string value inputted for each name and the 'prefixes'? I'm not sure how I'd go about doing this, or if I should persue a different strategy. I apologize in advance I've butchered any of the terminology, I'm two days into this book, and I have no prior computer science background. python. Share.Â They are processed by a list comprehension, which you will definitely want to learn if you don't already know: result = [''.join(pair) for pair in zip(infixes, suffixes)] result = [''.join(pair) for pair in zip(prefixes, result)].Â I just came across this question myself, and then checked the web to see how others did. Mine looked like this Learning with Python. Allen Downey Jerey Elkner Chris Meyers. Green Tea Press.Â The process of translating and using How to Think Like a Computer Scientist for the past two years has conrmed Pythonâ€™s suitability for teaching beginning students. Python greatly simplies programming examples and makes impor-tant programming ideas easier to teach. The rst example from the text illustrates this point. Scientist Think Python: How to Think Like a Computer Scientist How to Think Like a Computer Scientist â€" How to Think Like a. 354 PagesÂ·2012Â·2.73 MBÂ·12,594 Downloads. How to Think Like a Computer Scientist. Learning with Python 3 (RLE). V Think Python: How to Think Like a Computer Scientist. 483 PagesÂ·2015Â·3.11 MBÂ·5,893 DownloadsÂ·New!Â Master Python Programming with a unique Hands-On Project Have you always wanted to learn computer Learning to â€œThink Like a Lawyerâ€. 327 PagesÂ·2007Â·3.14 MBÂ·179,896 Downloads. The language of law school : learning to â€œthink like a lawyerâ€ / Elizabeth Mertz grounded Load more similar PDF files. PDF Drive investigated dozens of problems and listed the biggest global issues facing the world today. The answer key to "How to think like a computer scientist: learning with python 3". 3 stars. 1 fork.Â This anwer key only shows that answers that require python code. Each file is named exercise_iiyy.py. ii corresponds with the chapter. For example 02 corresponds to chapter 2. yy corresponds with the exercise number. For example 03 refers to the third exercise. About. The answer key to "How to think like a computer scientist: learning with python 3". Resources. Readme.

Learning with Python 3 (RLE). Version date: November 2011. by Peter Wentworth, Jeffrey Elkner, Allen B. Downey, and Chris Meyers.Â Tips, Tricks, and Common Errors GNU Free Document License. Search Page. How to Think Like a Computer Scientist: Learning with Python 3 Â». next | index. Â© Copyright 2011, Peter Wentworth, Jeffrey Elkner, Allen B. Downey and Chris Meyers. Created using Sphinx 1.0.7. http://openbookproject.net/thinkcs/python/english3e/[1/4/2012 9:36:22 PM]. Index â€" How to Think Like a Computer Scientist: Learning with Python 3. How to Think Like a Computer Scientist: Learning with Python 3 Â». Index. Symbols | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W. Like mathematicians, computer scientists use formal languages to denote ideas (specically computations). Like engineers, they design things, assembling components into systems and evaluating tradeoffs among alternatives. Like scientists, they observe the behavior of complex systems, form hypotheses, and test predictions. The single most important skill for a computer scientist is problem solving. Problem solving means the ability to formulate problems, think creatively about solutions, and express a solution clearly and accurately.Â In Python, the script looks like this: (For scripts, weâ€™ll show line numbers to the left of the Python statements.) 1 print("Hello, World!") This is an example of using the print function, which doesnâ€™t actually print anything on paper. The answer key to "How to think like a computer scientist: learning with python 3". 3 stars. 1 fork.Â This anwer key only shows that answers that require python code. Each file is named exercise_iiyy.py. ii corresponds with the chapter. For example 02 corresponds to chapter 2. yy corresponds with the exercise number. For example 03 refers to the third exercise. About. The answer key to "How to think like a computer scientist: learning with python 3". Resources. Readme. Connect and share knowledge within a single location that is structured and easy to search. Learn more. Think Python (How to think like a computer scientist) - Excercise 8.4. Ask Question. Asked 10 years, 2 months ago.Â I just came across this question myself, and then checked the web to see how others did. Mine looked like this: prefixes = "JKLMNOPQ" prefixes = list(prefixes) prefixes[5] = "Ou" prefixes[7] = "Qu" suffix = "ack". for p in prefixes: print(p + suffix). 3rd Edition (Using Python 3.x). by Jeffrey Elkner, Peter Wentworth, Allen B. Downey, and Chris Meyers. illustrated by Dario Mitchell. Copyright Notice. Contributor List. Chapter 1 The way of the program. Chapter 2 Variables, expressions, and statements.