BT furtHER - Curriculum Overview

Module 2, Python Fundamentals
(This module covers all of the foundations of Python with many tasks and small program design practices.)

Learning element	Learning chapters	Key learning outcomes on completion of module	Demonstration of learning	Delivery requirements

Part 1: Fundamentals	Chapter 1: Introduction to Python	• How we'll be using Python on this course.
		• Experience coding through Python consoles.
		• Creating and running your first Python script.
			Open spyder
			• show me the steps to run a python file (e.g. print('hello CFG'))
			  ○ save file -- naming it
			  ○ configure
			  ○ run -- shortcut
			• Open online Python editor to run some code: (e.g. (3.2 / 2.0) *7)
				Your first python file ch1_name.py: conation print hello and simple operation.

	Chapter 2: Operations, Strings and Variables	• Operators in Python, relating with module 1 web course. • Number types. • Names & Variables. • Strings in Python. • String formatting. • Comments in Python.	• what does ** mean in python? (how to do square root as well?) • explanation what is variable and how to use them (including rules for naming variables). • what need to beware of while doing string manipulation. • How to commenting code -- shortcut?	One file ch2_name.py, including all class practise code: • Task 1 simple operations • Task 2 variable practise. • Task 3 basic string manipulation. • Task 4 print formatting. • Homework code
	Chapter 3: Functions & Importing Modules	• Getting input from a user. • Introducing functions. • Calling a function. • Arguments. • More fun with functions. • Returning stuff from functions. • Importing modules and	Open bash • run T1 with command line what is a function in python • why function (including clean code tips) • syntax • what to beware of when write a function	One file ch3_name.py, including all class practise code: • Task 1 input from a user • Task 2 first function • Task 3 use range built-in function • Task 4 -- return version • Task 5 convert_temperature() -- return version

		functions	• how to use them (call a function) • what are arguments • how to storage output from a function • what to beware of when write return • demo different ways to import math library and pi function -- advantages and disadvantages of the different import method	• Homework
	Chapter 4: Conditionals	• Concept of programs and algorithms. • Using “logic” in programming. • Control structures. • Boolean type and expressions. • Conditional statements (“if” statements). • Else statements. • Elif statements.	• Explain what is a programme and algorithm (and algorithm should be?) • What is conditionals? • What is boolean type? • Logic operators quiz • demo using logic operators to get boolean results (true/false quiz) • difference between == and = • what need to beware of when using if, elif, else?	One file ch4_name.py, including all class practise code: • Task 1 and Task 2, but comment out (control 1 or “”” xxx “”” that python can skip those words) • Task 4 (else version) • Task 5 (also add some comment for why order is important) • Homework

	Chapter 5: Object Oriented Programming	• What are objects in Python.
		• Motivation: code is more reusable & easier to read in the future.
		• Name/type of class and attributes.
		• Apply basic object-orientation concepts such as inheritance and composition.
		• UML with OOP.
			• explain difference between
			  ○ class and function.
			  ○ class and object
			• why OOP? (explain:)
			  ○ modularity
			  ○ abstraction
			• what is:
			  ○ self
			  ○ __init__ (what need to beware of)
			  ○ inheritance
			  ○ composition
				One file ch5_name.py, including all class practise code:
				• Task 1 (with comments of how to test the code at the bottom of the file. e.g. jason.balance; jason.withdraw(100.0))
				• Task 2 (with comments of how to test the code at the bottom of the file. e.g. **cat version** of Snoopy = Dog(); Snoopy.bark())
				• Task 3 (with comments of how to test the code at the bottom of the file. e.g. **cook Robot version** of Nana = CleanRobot(); Nana.clean())
				• Task 4 (with comments of how to

				test the code at the bottom of the file. e.g.machineDog = SuperRobot(), machineDog.clean()) • Homework: Task 5
	Chapter 6: Command Line and Git with Python	• Run Python using the command line. • Steps to run a Python file with command line. • Navigating using the command line. • Working with git in the command line. • Setting up a new repository. • Managing conflicts in git. • Avoiding conflicts in git. • Branches.	Using Bash: • navigate to your python file dir • print the file path with your current dir • make a new dir, go to that dir and back • what you need to beware of when use command line. Github: • Start repository. • how to clone a repository. • Push code to git and commit with comments. • explain why git? • how to solve conflict? • what routine can help to avoiding conflict?	One file ch6_name.txt • Summaries what you have learnt so far • What you have struggled so far • What you proud of/enjoyed so far
Part 2: Design a simple programme.	Chapter 7: Debugging	• use print function for debugging • use breakpoints for debugging • Best practice. • A little bit about software project management. • it is a iterative process.	Debugging • how to set print to check variable within a function? • how to set breakpoint and how does it work? • demonstrate how to use step-in and step-out? Project management • What are the key point of project management, and what are you going to do with it? • summarise your understanding of how to design a program.	One file ch7_name.py, including all class practise code: • Task 1, how to use breakpoint, step in and out to debug.

	Chapter 8: Mobile Data Bundle Purchase Program	• Design a real world scenario program. • Development folder and test file. • Think through the problem logic. • Writing steps based on the logic flow. • A more realistic definition: using return. • Breaking the task down, using functions as subroutines. • Passcode testing. • Mobile data bundle purchase function. • Get used to testing alone with development. • How to design a test file to test the whole program as a whole. • Designing functionalities based on story cards and logic.	Explain your code to me: • The whole flow, design of the program • What is your understanding of the requirement • How many functions you design ○ What are their input, output and functionalities • How those functions interact with each other • How did you test each function • How did you test the whole program together	Submit three files: Two files of the final version of: • SimpleBundlePurchase.py • test_DataBundlePurchase.py Another file that contain all the tasks: • Task 1, return with conditionals
Part 3: Data Types.	Chapter 9: Lists	• Intro to lists (also called arrays). • Index of list. • Make changes to a list. • Slice of a list. • Sorting a list. • Modifying sort behaviour. • Introduction of Tuples. • Lambda function, used for a more complicated sort.	• Describing the difference between mutable and immutable data type • Explain you submitted code to me, which parts you struggled, which parts you enjoyed.	One file ch9_name.py, including all class practise code: • Task 1 first list • Task 2 make change of list • Task 3 slice list • Task 4 sort list • Task 5 tuples • Task 6 lambda sort
	Chapter 10: Dictionaries	• Intro to dictionaries. • Keys and values. • Creating and using operations in	• Describing the difference between dictionary and other data type. • Is dictionary mutable?	One file ch10_name.py, including all class practise code: • Task 1 create dict, assign, retrieve and update values.

		dictionaries. • Creating and assigning values to a dictionary. • Getting dictionary values through a key. • Overwriting dictionary values or doing operations with values. • The 'print format' for dictionaries. • Other operations. • Get keys and values from a dictionary. • Avoiding key errors. • Sorting a dictionary by value. • Sorting dictionary values in descending order.	• Explain you submitted code to me, which parts you struggled, which parts you enjoyed.	• Task 2, Task 3 phoneBook dict • Task 4 key, value data type • Task 5 get items in key and value • Task 6 avoiding key errors • Task 7 sorting dict • Task 8 sorting dict with a task
Part 4: Iterations	Chapter 11: Iterations 1 'While loop' and Guessing Number Game Programme.	• Loops in control structures. • The while loop. • Early exit and continuation. • 'Break' statement. • 'Continue' statement. • Using loops to solve useful problems in Python.	• Describe your understanding of a conditional loop. • What is the syntax of while loop? • Explain you submitted code to me, which parts you struggled, which parts you enjoyed.	One file ch11_name.py, including all class practise code: • Task 1, repeat division • Task 2, triangular numbers • Task 3, students' mark level responds • Task 4, learn when to use break in while. • Task 5-Task 6, Design the guessing number game program and testing it. • Task 7, How to make the guessing number game as robust as possible.
	Chapter 12: Iterations 2, 'For	• Introduction of "for" loops. • The "for" loop vs the "while" loop	• Describe your understanding of a counting loop.	One file ch12_name.py, including all class practise code:

	loop'.	(intuitively). • Looping through a list. • Looping through other types. • Looping through a dictionary with sorted order. • Search and operations inside a "for" loop. • Modifying range for special cases of counting loops. • Loop Controls. • Nested loops. • A multiplication table.	• What is the syntax of for loop? • Your understanding of the difference between "for" and "while" loops. • Explain you submitted code to me, which parts you struggled, which parts you enjoyed.	• Task 1 loop through my_shopping_cart list. • Task 2 update list values. • Task 3 loop through your own list with any features you added in the loop. • Task 4 loop through str type. • Task 5 loop through tuple type. • Task 6, Task 7 loop using "for" loops to get different keys and values through dictionaries. • Task 8, combine counting loop and conditionals to filter out values. • Task 9, design your own sum function. • Task 10, loop with index values. • Task 11, loop with range function. • Task 12, break "for" loops. • Task 13, nested loops • Task14, multiplication table.
Part 5:More on OOP	Chapter 13: OOP project	• Revise OOP with examples • Combine knowledge of chapter 1-12 and used in the OOP project • Distinguish between python modules and python class	• What are the syntax of defining a class? • What is 'self' used for? • What are the differences between superclass and subclass? • Run your final version of task 7/ OOP project code and explain the design of your code. • Why testing while developing is important? • Explain what are the key features when design tests for a project? • Explain the difference and similarities between python module and class.	One file ch13_name.py, including all class practise code: • Task 1, initialise person class • Task 2, more functionalities for person class • Task 3, greeting filter • Task 4, subclass - Wizard! • Task 5, redefine __init • Task 6, new method for Wizard! • Final version of your task 7

Part 6: Real world applications	Chapter 14: Database	• Understand database structure. • Tables, columns, keys in database. • Learn mySQL and SQLite. • Online and offline databases. • Learn how to create a database and read a database.	• Explain what is database and why it is useful? • What is the syntax to create a database? • How to insert data to database table, what you need to beware of in this process? • What is the difference between static data and stream data? • How to select data from database tables? • Show me your homework and explain how you did it	One file ch14_name.py, including all class practise code: • Task 1, create table and insert data. • Task 2, insert data automatically with variables. • Task 3, read/select data from database. • Homework:, phonebook data table.
	Chapter 15: APIs	• Use an API to trigger an email to be sent and get weather information. • Get data from a user using a HTML form and use it in an API call. • Get JSON formatted data from a payload response.	• What are APIs used for? • What's an API key? • What's JSON, and why might we want to use it?	One file ch15_name.py, including all class practise code: • Task 1, testing on Mailgun API with Python. • Task 2, get current weather information from API, return in JSON and print with normal sentence format.
	Chapter 16: Web App Development	• Learn from examples: Flask. • Understand how Python, Flask, and HTML interact with each other. • Routes & decorators. • Pass variables between Python and HTML using render_template(), a HTML form and Jinja2 syntax.	• what is web framework? • How does decorator works? • Explain your code	One file ch16_name.py, including all class practise task 1-5 code.
	Chapter 17: Deploying your code	• What it means to deploy code. • Where's my website and how do I update it? • Getting your own project set up on Heroku.	• Why deploy? • What is cloud-based platform? • What are the steps to deploy your web to Heroku?	One file ch17_name.py, including all class practise code. Phonebook project dev file and test file.