Week 6: Python Programing - Consolidation

In this week's session we will focus on putting together all the skills we have learned so far. We will deviate from the normal PRIMM routine (predict-run-investigate-modify-make) so that you can spend plenty of time practising modifying small programs.

Focus for this week: Trouble-shooting

Types of errors

1. **Logical** – the program runs but doesn't do quite what it should do.
2. **Syntax** – the program won't run at all because there is something wrong in the way it is written such as a missing colon, bracket or quote mark, or variables spelled in two different ways, or a function spelled incorrectly.
3. **Run-time** – the program starts to run but when you input something at the keyboard it crashes. This is quite often because your program expects a number and you have not converted it using int(), or because you have tried to divide by zero.

Keep a note of all errors in the red notebook or in a list somewhere. This is important and you can turn it into a poster for your students!

Answer to homework

```
# correct program

def Example2():
    # repeat while carryOn == True
    carryOn = "y"
    print("Name program")
    print()
    while carryOn == "y":

        yourName = input("Please enter your name")
        carryOn = input("Do you want to enter another name? (y/n) ")

        print("Hello "+yourName)
    print("End of program")
```

Exercises – easier

These exercises start by giving you practice with print(), input() and if statements. If you can do these progress to the harder exercises that involve loops.

Download the starter program called starter_easy.py

1. In the downloaded file, run exercise1() and see what it does. Then edit it to create the initial of your first name.	

2. Now run exercise2() and see what it does. Now edit it to ask the user for two numbers, add them together and print the answer.	`>>> exercise2()`
	`Please enter a number ... 45`
	`Please enter another number :35`
	`The sum of the numbers is 80`

3. Now copy exercise2() as exercise3() and edit it to ask the user for three numbers, multiply them together, and print the answer.	`>>> exercise3()`
	`Please enter a number: 5`
	`Please enter another number : 7`
	`Please enter another number : 4`
	`The product of those three numbers is 140`

4. Now copy exercise2(), rename as exercise4() Modify the program so that the user only inputs one number and the program checks if it is more than or equal to 50 and if it is tells the user that they have passed. If the mark is less than 50 the message should say "You have not passed".	`>>> exercise4()`
	`Please enter a number: 60`
	`You have passed!`
	`>>> exercise4()`
	`Please enter a number: 35`
	`You have not passed!`

5. Now run exercise5() several times to see what it does. It uses the elif statement to have several paths in the program. Change exercise5() so that it also gives an D grade if the mark is 40 or over.	`>>> exercise5()`
	`Please enter your mark 46`
	`Your grade is D`
	`>>> exercise5()`
	`Please enter your mark 78`
	`Your grade is A`

6. Now go back to exercise4() and save it as exercise6(). Modify it so that it asks for a number between 1 and 10. If the number given is less than 1 or greater than 10 it should give an error message.	`>>> exercise6()`
	`Please enter a number between 1 and 10: 14`
	`That number is not between 1 and 10`
	`>>> exercise6()`
	`Please enter a number between 1 and 10: 5`
	`Thanks`

7. Now edit exercise6() and save as exercise7(). Change this program so that it keeps asking the user, using a while loop, until the user enters a number between 1 and 10.	`>>> exercise7()`
	`Please enter a number between 1 and 10: 12`
	`That number is not between 1 and 10`
	`Please try again: 0`
	`That number is not between 1 and 10`
	`Please try again: 45`
	`That number is not between 1 and 10`
	`Please try again: 5`
	`Thanks`

Exercises – harder

Start here if you got at least to exercise 5 last week. The first exercise was exercise 7 from last week's tasks, followed by last week's extension task.

Download starter_harder.py from http://teachinglondoncomputing.org/ks3-week-6

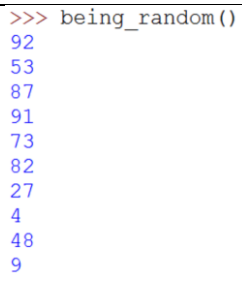
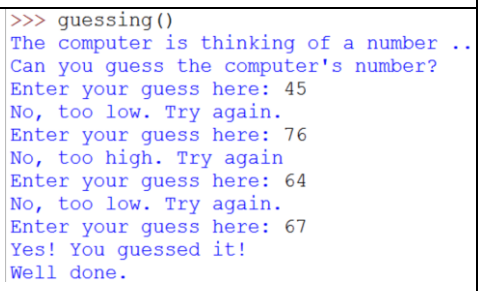
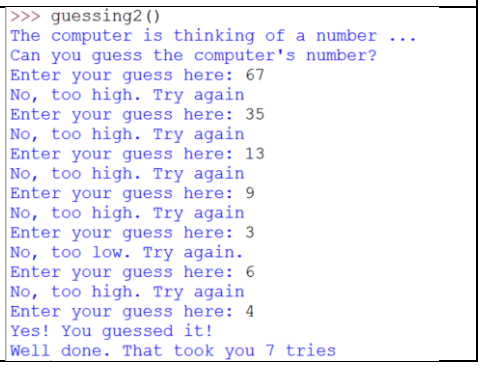
8.	Develop the password program which was exercise 7 last week. Write a program which will ask the user for their password and allow them three tries to get it right. The password is already stored and is "secret".	Enter your password: fred That password is not the one stored Try again! bananas That password is not the one stored Try again! secret Yes that's the correct password! You had 3 tries

Try the following statements out in the Python shell (not in a new file).

```
>>> ord("a")
97
>>> ord("A")
65
>>> chr(97)
'a'
>>> chr(65)
'A'
>>> chr(66)
'B'
>>> chr(67)
'C'
```

ord() and chr() are functions that you can use to work out the ASCII code in decimal for characters that are entered in the keyboard

9.	Write a program that will write out all the letters from "a" to "z" using the chr() function.	a b c d e f g …. etc.
10.	Write a program that will ask the user for a character and output its ASCII value – use the ord() function.	>>> exercise10() Type in one character : t Its ASCII value is 116
11.	Write a new function called change_my_password() which asks the user if they want to change their password. If they want to change it they must type it in twice exactly the same. If they do not type it in twice exactly the same, give an error message.	>>> change_my_password() Do you want to change your password? (y/n)y Type in your new password: chutney Type it in again, please: chutney Thanks, I will reset your password

12.	Look at the program being_random() that you have downloaded in the starter. It uses a random function to generate a random number between 1 and 100. Modify the program so that it generates 10 random numbers and prints them out. Test it to make sure that it doesn't generate the same numbers each time.	
13.	Look at the program guessing() that you have downloaded from the website. Try it out and see what it does. Modify this program so that the user has to keep guessing the computer's number until it gets it right, reporting "Too high!" or "Too low!" each time if it's not right.	
14.	Modify the program guessing() so that it counts how many tries the user has had.	

Homework

Create a debugging activity. You will need to:

- Write a program or use a program that you have used in the course and save the correct, working version.
- Add 5-10 bugs in it.

Email the buggy and the corrected version to me before next Monday.

In the last two weeks we will cover 2 complete programs with different amounts of scaffolding depending on your confidence, including in the last week a simple graphical user interface for Python called EasyGui.

As you do these exercises, make a note of any errors that you come across in the notebook and how you fix them.

Check your answers at http://teachinglondoncomputing.org/ks3-week-6