www.thegordon.edu.au

the Gordon

Python Programming

Section 3 – Calculation with Variables

UEE60411 Advanced Diploma of Computer Systems Engineering

UEENEED103A	Evaluate and modify Object Oriented code programs

For further information:

Steve Gale
ICT – Science and Technology
Ph: (03) 5225 0940
Email: sgale@gordontafe.edu.au

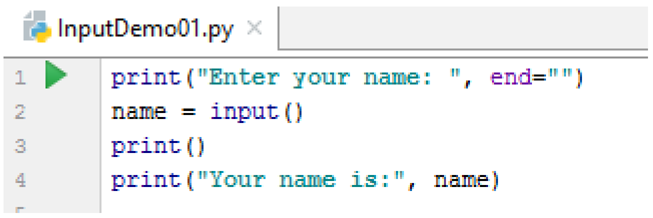
1 – Console Input

So far we have looked at **output** to the console. Each time we have used a variable it has been hard-coded into the program. All well and good, but what if we want to get input from the user?

In our “Hello Phil” program, for example, where we print out the name of the user to the console, we still have to write the person’s name into the program. This is obviously not possible if we want a *user*, not a *programmer*, to use the program.

It would be nice to be able to get the user to type their *own* name into the program, place that into a variable, and then display the variable. How would we do that?

Like this:



```
1  print("Enter your name: ", end="")
2  name = input()
3  print()
4  print("Your name is:", name)
```

Let’s look at each line, one-by-one.

Line	Purpose
1	Displays the command “Enter your name: “ on the console. The **end=””** command means that the line ends with an empty space, rather than a carriage return or new line marker. This means the user can type their name at the end of the line. Remove it, and see what happens.
2	This is the important bit. Here we invoke the **input()** function, which takes the user’s keyboard entry and assigns it to the **name** variable.
3	Print an empty line
4	Print out the message, with the content of the **name** variable (which the user entered via the keyboard) tacked onto the end.

Create the above program (calling it **InputDemo01.py**) and run it to see what happens.

NOTE: In PyCharm Edu, when the program executes, make sure you click down into the console before you start entering data! If you don’t, you will start editing the actual program!

If you want to save a bit of coding (and not have to worry about the **end=””** command), you can actually incorporate your prompt into the actual input() function. The same program as **InputDemo01.py** can be coded like this:

```
name = input("Enter your name: ")
print()
print("Your name is:", name)
```

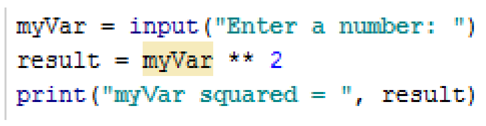
This does *exactly* the same as the previous code, but in three lines instead of four.

String Input

It's also important to remember that *all* input from the keyboard like this is in the form of a *character string*. If you enter the number **6**, for example, it will enter the variable as the *character* "6", **not** the numeric value of 6.

If you need to enter numbers you will need to **cast** the variable into number format (we looked at the reverse of this in the last lesson, where we **cast** a number into a string for *output*).

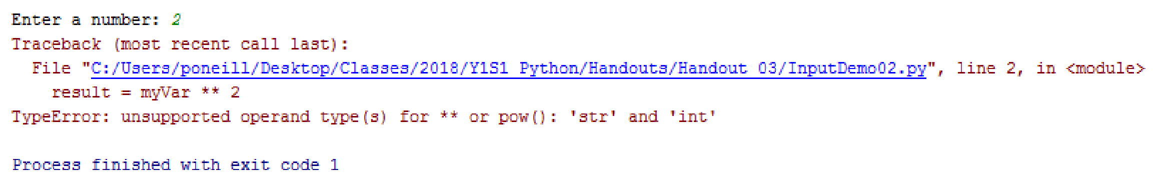
Let's create a new program and try this out:



```
myVar = input("Enter a number: ")
result = myVar ** 2
print("myVar squared = ", result)
```

(you can actually see where PyCharm Edu has highlighted `myVar` because it has picked up that it is an error)

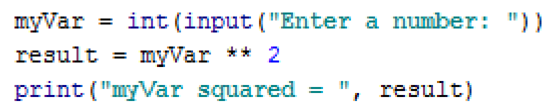
When you run this program, you will get a result something like this:



```
Enter a number: 2
Traceback (most recent call last):
  File "C:/Users/poneill/Desktop/Classes/2018/Y1S1 Python/Handouts/Handout 03/InputDemo02.py", line 2, in <module>
    result = myVar ** 2
TypeError: unsupported operand type(s) for ** or pow(): 'str' and 'int'

Process finished with exit code 1
```

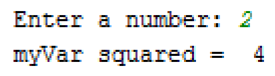
It has asked for the input, and when you enter 2 and hit the ENTER key, it produces an error. In this case, it is telling us that it cannot raise a string (`myVar`) to the power of an integer (`2`).

To make it work, you need to convert (or **cast**) the string into a numeric datatype, either an **int** or a **float**. We can use a couple of functions to do this, as shown in this modification to the program:



```
myVar = int(input("Enter a number: "))
result = myVar ** 2
print("myVar squared = ", result)
```

If you enter the number 2 at the prompt, you will get:



```
Enter a number: 2
myVar squared =  4
```

Because we have converted the entered data (a string) into an integer, the program can now perform the calculation and raise it to the power of two.

Changing a Variable's value

You can change a variable's value by simply assigning it a new value: `num = 1`

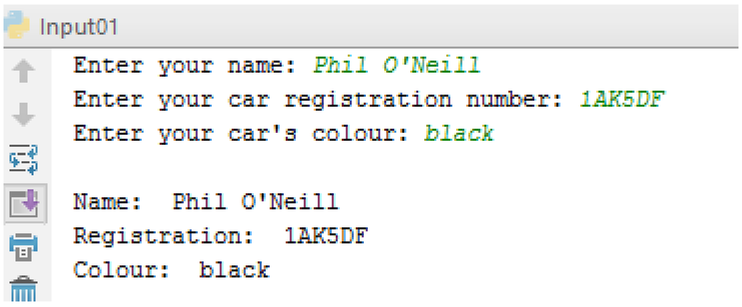
Sometimes, however, you want to increase or decrease a variable's value. If you wanted to add 1 to a variable called `counter`, for example, you could use: `counter = counter + 1`

To reduce the amount of coding, you could also use: `counter += 1`

String Input Exercises

Complete the following Python exercises, ensuring you create and save your files in a Python Project.

1. Create a Python program called **Input01.py**. This program should ask a user for their name, car registration number, and the car colour. You will need three variables called `name`, `rego`, and `colour`, all of which will be initialised by reading input from the user. Once it has all three variables, print out a result as follows:



```
Input01
Enter your name: Phil O'Neill
Enter your car registration number: 1AK5DF
Enter your car's colour: black

Name:  Phil O'Neill
Registration:  1AK5DF
Colour:  black
```

2. Save the above program and call it **Input02.py**. Modify the output so that it produces the output shown below. You will need to use **string formatting** for the best way of doing this. As an example, the second line of output ('You have a black car") would be coded as:

```
print("You have a {0} car".format(colour))
```

The last line would be coded like this:

```
print("{0} has a {1} car, whose registration is {2}".format(name, colour, rego))
```

Note how the positions of the variables are a list starting with the number 0, and they correspond with the variables in that order. Try rearranging the order of the variables to experiment with the results.

```
Enter your name: Phil O'Neill
Enter your car registration number: 1AK5DF
Enter your car's colour: black

Phil O'Neill
You have a black car
Registration number is 1AK5DF

Phil O'Neill has a black car, whose registration is 1AK5DF
```

3. Create a program called **Input03.py**. It should read the name of an Australian state (which is input by the user) into a variable called `state`, and the names of three towns in that state into three variables called `town1`, `town2`, and `town3`.
   Create another variable called `message`.
   Use these variables (and string formatting, as in #2 above) to display the data the user enters as shown below. The final line of output should be only `print(message)` – you will need to figure out how to create the message variable in your code.

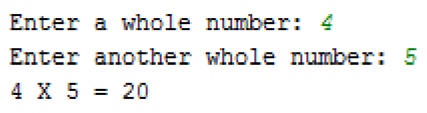
```
Enter a state: Victoria
Enter a town: Geelong
Enter another town: Ballarat
Enter a third town: Warrnambool

Victoria
Geelong
Ballarat
Warrnambool

Geelong, Ballarat, and Warrnambool are all towns in Victoria
```

4. Create a program called **Input04.py** that uses the `input()` function to read the names of three months. Display the months in the following order:
   The order in which they were entered
   The reverse order
   The second month, followed by the first, followed by the last (like the three acts of the movie *Pulp Fiction*)

```
Enter a month: January
Enter a second month: February
Enter a third month: March
The months in entry order are January, February, and March
The months in reverse order are March, February, and January
The months in "Pulp Fiction" order are February, January, and March
```

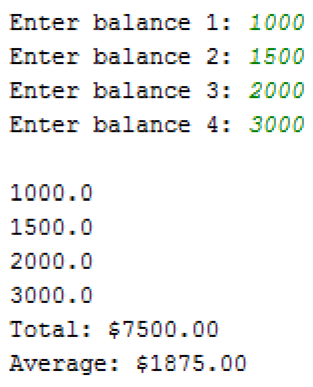
5. Open up **Sample06.py** and examine the code. Run the program and see if you can trace how it works. Save it as **Input05.py**
   Modify it so that the program:
   Requests the user enters two numbers (converting them to integers)

Multiplies them together
Puts the answer into a third variable and displays it as shown here:



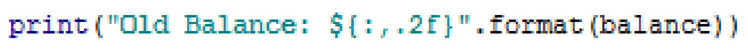
```
Enter a whole number: 4
Enter another whole number: 5
4 X 5 = 20
```

6. Modify **Input05.py** and save it as **Input06.py**
Allow the user to enter four numbers representing bank balances (these should be **floats** – that is, numbers with decimal places).
You should declare another two variables called `total` and `average`.
The four bank balances should be added together and the result assigned to `total`.
The `total` should be divided by four to give the `average`.
The program should then display the four balances, the `total`, and the `average`, as shown below.

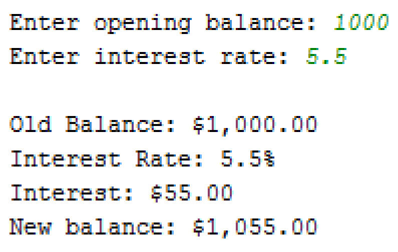


```
Enter balance 1: 1000
Enter balance 2: 1500
Enter balance 3: 2000
Enter balance 4: 3000

1000.0
1500.0
2000.0
3000.0
Total: $7500.00
Average: $1875.00
```

7. Create a program called **Input07.py**
This will ask for an investment balance, and an annual interest rate (the user should enter the interest rate as a double: eg. Entering 5.5 is equivalent to 5.5%. Remember that 5.5% is actuall 0.055, so you will need to divide the entered interest rate by 100 before you perform your calculations).
The program should calculate the new balance (old_balance + interest = new_balance).
You should display the results as shown:
(HINT: To get a thousands separator, you would use:



```
print("Old Balance: ${:,.2f}".format(balance))
```



```
Enter opening balance: 1000
Enter interest rate: 5.5

Old Balance: $1,000.00
Interest Rate: 5.5%
Interest: $55.00
New balance: $1,055.00
```

Numeric Input Exercises

Complete the following Python exercises, ensuring you create and save your files in a Python Project.

1. Create a Python program called **PInput01.py**. It should take two numbers from the keyboard, convert them into **integers**, and display the product (multiplication) of them, as shown below:

```
Enter first number: 5
Enter second number: 6
5 * 6 = 30
```

2. Make copy of **Pinput01.py** and call it **Pinput02.py**
Ask the user for two **integers** and *add* them together, displaying the result as below. Then add 10 to he first and 20 to the second number, add the new numbers again, and display the results with the appropriate message, as shown:

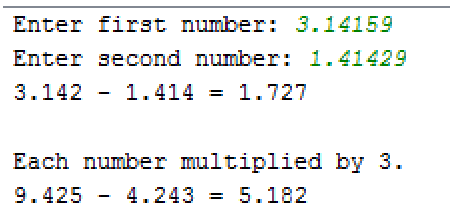
```
Enter first number: 5
Enter second number: 6
5 + 6 = 11

10 added to the first
and 20 to the second number.
15 + 26 = 41
```

3. Make copy of **Pinput02.py** and call it **Pinput03.py**
Ask the user for two decimal numbers (remember to convert the keyboard inputs into **floats**). Display the sum of the two numbers, then add 12.53 to each number and display the sum again with an appropriate message, as shown:

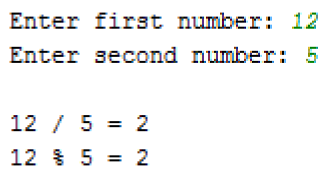
```
Enter first number: 11.0
Enter second number: 23.0
11.00 + 23.00 = 34.00

12.53 added to number.
23.53 + 35.53 = 59.06
```

4. Make a copy of **Pinput03.py** and call it **Pinput04.py**
Input two decimal numbers from the keyboard and display the difference (subtract) of the two numbers.
Multiply each number by 3 and again display the difference, with an appropriate message (make all the numbers correct to 3 decimal places).

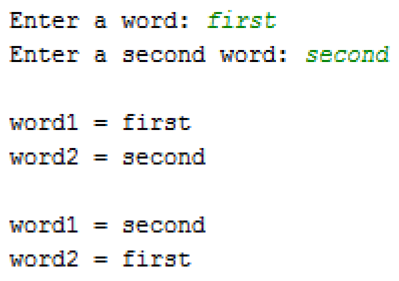
(And yes, for any math-heads out there, I used π (3.14159) and Pythagoras' constant (1.41429) ☺)


```
Enter first number: 3.14159
Enter second number: 1.41429
3.142 - 1.414 = 1.727

Each number multiplied by 3.
9.425 - 4.243 = 5.182
```

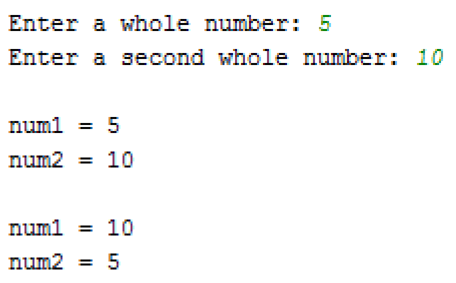
5. Make a copy of **Pinput04.py** and call it **Pinput05.py**
Input two integers from the keyboard.
Display the whole part, and the remainder part of the result when `num1` is divided by `num2`
(HINT: Check your **arithmetic operators** table from Handout 02)


```
Enter first number: 12
Enter second number: 5

12 / 5 = 2
12 % 5 = 2
```

6. Pinput06.py
Input two strings, `word1` and `word2`, from the keyboard. Display them in that order. Then swap the contents of `word1` and `word2` (that is, put the string that was in `word1` into `word2`, and vice versa).
Display `word1` and `word2` again to prove that the contents were exchanged.
(HINT: Remember that to swap two variables in a computer program, you need a *third* variable to temporarily store one of the values)

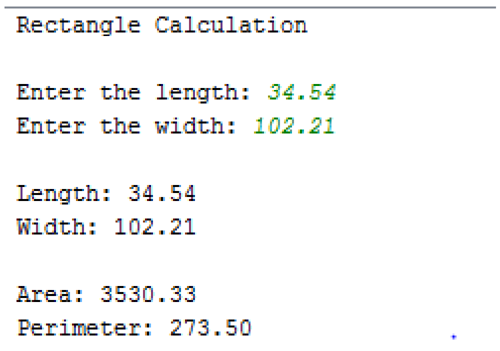

```
Enter a word: first
Enter a second word: second

word1 = first
word2 = second

word1 = second
word2 = first
```

7. **Pinput07.py**
   Enter two integers into the variables **num1** and **num2**. Display these numbers. Swap the contents of the variables and display them again (similar to **Pinput06.py**, above)



```
Enter a whole number: 5
Enter a second whole number: 10

num1 = 5
num2 = 10

num1 = 10
num2 = 5
```

8. **Pinput08.py**
   Enter two decimal numbers (floats) to represent the length and width of a rectangle. Calculate and display the area and the perimeter.



```
Rectangle Calculation

Enter the length: 34.54
Enter the width: 102.21

Length: 34.54
Width: 102.21

Area: 3530.33
Perimeter: 273.50
```