

Python Programming

Tutorial 03 – Inputs and outputs

UEE60411- Advanced Diploma in Computer System Engineering.

UEENEED111A Develop, test and implement object- orientated code.

For further information:

Leanne Matheson

ICT – Creative Business Enterprise

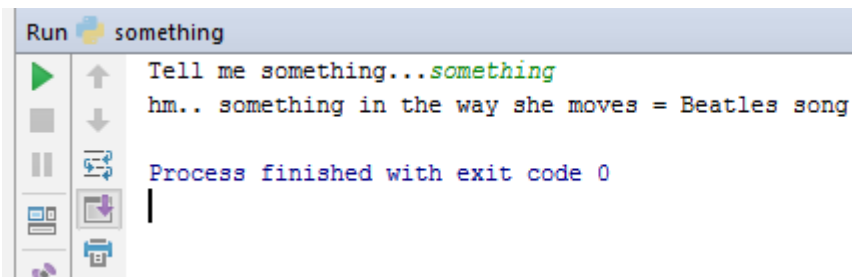
Ph: (03) 5225 0616

Email: lmatheson1@gordontafe.edu.au

The Input() function

- The program below **prompts the user** to input some data from the console (most likely using a keyboard)
- the `input()` function is **invoked without arguments** (this is the simplest way of using the function); the function will switch the console to input mode; you'll see a blinking cursor, and you'll be able to input some keystrokes, finishing off by hitting the Enter key; **all the inputted data will be sent to your program** through the function's result;
- note: you need to assign the result to a variable; this is crucial – missing out this step will cause the entered data to be **lost**;
- we use the `print()` function to output the data we get, with some additional remarks.

```
something = input("Tell me something...")  
  
print("Hm...", something, "in the way she moves=Beatles song")
```



- **something** is a variable, it is populated when it is collected from the user via the `input()` function.
- the `input()` function is invoked with one argument – it's a string containing a message;
- the message will be displayed on the console before the user is given an opportunity to enter anything;
- `input()` will then do its job.
- the `print` function can display strings of text as well as the contents of the `something` variable

A few data Types in Python

Data type	Description
String	"The text between two sets of inverted commas"
integer	Whole numbers both positive and negative the minus and plus can used as + and dash – on the keyboard
float	Numbers with a decimal point e.g. 3.1416

Operators in Python

Operator	Meaning
+	addition
-	subtraction
*	multiplication
/	division
**	to the power of
%	modulus-The remainder after division
=	= is an assignment operator (a = b assigns a with the value of b)
==	== is the question <i>are these values equal?</i> (a == b compares a and b) are they the same?
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
!=	Not equal to
//	Integer division – rounds to a whole number.

Short-cut Operators

$i = i + 2 * j$	$i += 2 * j$
$Var = Var / 2$	$Var /= 2$
$Rem = Rem \% 10$	$Rem \% = 10$
$j = j - (i + Var + Rem)$	$j -= (i + Var + Rem)$
$x = x ** 2$	$x ** = 2$

Operators and their priorities

You probably remember from school that **multiplications and divisions precede additions and subtractions**. (The **BOMDAS** or **BODMAS** rule)

For an expression print $(2 + 3 * 5)$

You should first multiply 3 by 5 and, keeping the 15 in your memory, then add it to 2, thus getting the result of 17.

The phenomenon that causes some operators to act before others is known as the **hierarchy of priorities**.

Python precisely defines the priorities of all operators, and assumes that operators of a **larger (higher) priority perform their operations before the operators of a lower priority**.

So, if you know that $*$ has a higher priority than $+$, the computation of the final result should be obvious.

Naming variables

- A variable name must begin with a letter
- A variable name cannot begin with a number
- A variable name can contain only numbers, letters and underscores
- There cannot be any spaces in a variable name
- The name of a variable is case-sensitive when it is put into use.

Reserved Keywords in Python

and	del	from	not	while
as	elif	global	or	with
assert	else	if	pass	yield
break	except	import	print	
class	exec	in	raise	
def	for	lambda	try	

The above words are not available to be used as names, neither for variables, nor functions, nor any other named entities that you are generating. The reason for this is: Python already has a pre-determined meaning for these words.

Declaring integers and float variables.

Variables are automatically set with a data type when the data type is not specified.

If you have a variable called var,

```
var="2"           #is a string
var=2             #is interpreted as an integer
var=2.0
or var=2.         #is interpreted as a float.
```

You can specify data types for variables, by writing their data type outside brackets.

e.g. `result = int(a/b)`

The variable called result has been defined as an integer.

Collecting integers and float variables

The input function alone only collects strings, so we need to be able to convert strings to numbers. Numbers can be expressed in Python as integers and floats.

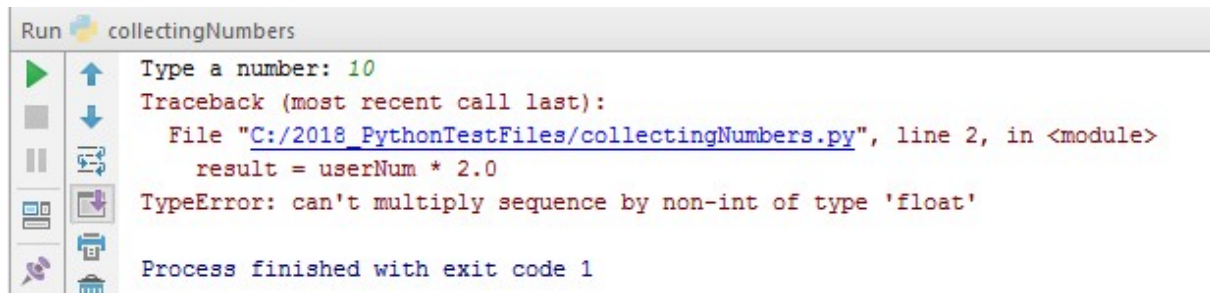
If we do the following: the content entered by the user will be interpreted as a string.

```
userNum = input("Type a number: ")

result = userNum * 2.0

print(result)
```

The output will show up an error in line 2:

A screenshot of a Python IDE window titled 'Run collectingNumbers'. The console shows the user input '10' for 'Type a number:'. A traceback error is displayed: 'File "C:/2018_PythonTestFiles/collectingNumbers.py", line 2, in <module> result = userNum * 2.0 TypeError: can't multiply sequence by non-int of type 'float''. The message 'Process finished with exit code 1' is at the bottom.

```
Run collectingNumbers
Type a number: 10
Traceback (most recent call last):
  File "C:/2018_PythonTestFiles/collectingNumbers.py", line 2, in <module>
    result = userNum * 2.0
TypeError: can't multiply sequence by non-int of type 'float'
Process finished with exit code 1
```

The **userNum** variable will only collect **string** data in the above example, so it will be unable to do the calculation `userNum * 2.0`. An error statement will be generated.

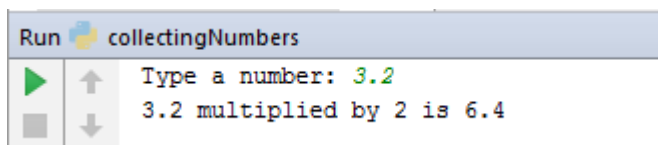
So to collect a number from the user we need to specify the **data type** along with the `input()` function. In the next example we have specified that the **userNumber** input data is a variable of the **float** type.

```
userNumber = float(input("Type a number: "))

result = userNumber * 2.0

print(userNumber, "multiplied by 2 is ",result)
```

The user's input can be interpreted as a float (or number with a decimal point). The result is then successfully calculated.

A screenshot of a Python IDE window titled 'Run collectingNumbers'. The console shows the user input '3.2' for 'Type a number:'. The output is '3.2 multiplied by 2 is 6.4'.

```
Run collectingNumbers
Type a number: 3.2
3.2 multiplied by 2 is 6.4
```

Printing a multi-line output

The use of the `print` function, but with three sets of inverted commas and each end of the content will allow multiline outputs.

Exercises

1. Expand the code below, so that you can collect an input from the user to work out their year of birth if you collect their current age. Collect the age as a float. Express the year born as an integer. Save as **yearBornCalculator.py**

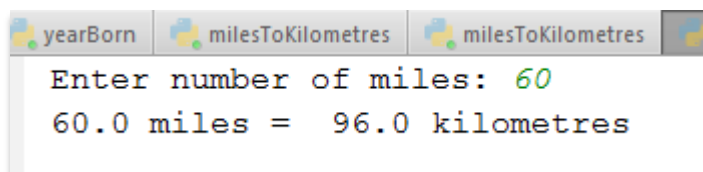
```
age=50
yearBorn = 0
yearBorn = 2018- age
print(yearBorn)
```

2. Calculating hours. Calculate the user's age in hours. You will need to ask the user to enter a whole number here. Collect it as a float in case they get contrary.

```
hours = age * 365 * 24
Print("\nYou're over", hours, "hours old")
```

Save as **hoursOld.py**

3. Alter the file **jokes3.py** to add a "Dad" joke of your own. Include inputs and outputs. Save as **yourNameJokes.py**.
4. Write your own python file that displays large text, similar to the Game Over example shown above. Type your **first name** using a display that is 6 lines high (GAME in the example is 6 lines high) Call the file **firstNameLarge.py**
5. Write a program that converts miles to kilometres. (There is approximately 5/8 of a mile in one kilometre) Include an input that collects the number of miles from the user- as a float, then displays the result of the conversion with a statement. The result may be expressed as a float also. Sample output could be:



Call the file **milesToKilometres.py**

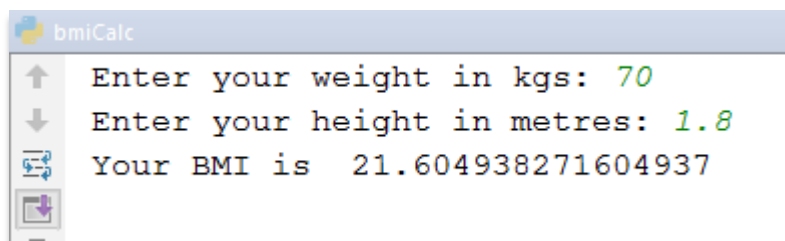
6. Your BMI or Body Mass Index is calculated by dividing your weight in kilograms by your height (squared) in metres.

e.g.
$$\frac{\text{weight(kg)}}{\text{height (m)}^2}$$

or
$$\frac{\text{weight(kg)}}{\text{height x height (m)}}$$

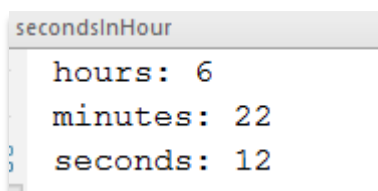
Build a calculator that determines a BMI value for a given height (in metres) and weight (in kilograms). Display the result. Call the file: **BMI_Cal.py**

7. Modify BMI_Cal.py to collect data from the user, then display their individual result. Call this file **BMI_Cal_User.py**



```
Enter your weight in kgs: 70
Enter your height in metres: 1.8
Your BMI is 21.604938271604937
```

8. Write a program that converts 6.37 hours into hours, seconds and minutes. It should work for any decimal value of hours that you load into the first hours variable. Your output should express:



```
hours: 6
minutes: 22
seconds: 12
```

Call the file **timeConverter01.py**

9. Enhance the TimeConverter program so that it collects a decimal value (called **hrsEntered**) of hours from the user, then converts it and displays a message that states.

hrsEntered Hours = nn Hours, nn Minutes and nn Seconds

e.g.

```
secondsInHour secondsInHour
Type hours with a decimal point: 6.37
6.37 Hours = 6 Hours 22 minutes and 12 seconds
```

Call this file **timeConverter02.py**

Files for submission:

yearBorn.py
hoursOld.py
firstNameLarge.py
milesToKilometres.py
BMI_Cal_User.py
timeConverter02.py