Data Types and Variables

Aim: *To learn about different data types and variables.*

A variable is a space in the computer's memory allocated to a piece of information. You can think of a variable as a box where this information is stored. The box has a label – the variable name. There are different types of boxes for the different types of data, whether text, integers or decimals etc. There are also different size boxes to store different amounts of information.

Note: *In some programming languages, you must declare each variable before you use it in your program (basically say 'this variable exists and this is what it is for'). In Python, you do not need to declare variables but you do need to remember what type of data you have assigned to each. You will create errors in your programs if you mix up your data types.*

Task 1 – Data Types

A few of the data types you might use are shown in the table below. Try and match each type to its description. Common sense should get you through here but if you need help, search for 'Python data types'.

	Variable Type				Desciption
1	bool (Boolean)	•	•	a	A group of unordered items that are all different (each is unique).
2	list	•	•	b	Positive or negative whole numbers.
3	set	•	•	c	Positive or negative numbers with decimal points.
4	float (decimal)	•	•	d	This has two possible values, True or False (1 or 0).
5	int (integer)	•	•	e	Essentially, a piece of text of any kind.
6	str (string)	•	•	f	An ordered sequence of items.

Task 2 – Naming Variables

There is really only one common convention in place for the naming of variables in Python. This is to use all lower-case letters separated by underscores e.g. *first_name*, *last_name* etc.

Naming variables is otherwise down to personal style. Descriptive words are encouraged, for example, *number_of_rows* or *last_letter*.

One convention that is sometimes used is to end the variable name with the type of data that the variable will hold. For example, *first_name_str* for a string, or *highest_int* for the highest integer. This can help you remember what each variable is being used for. However, this is only useful if the words chosen don't make it obvious anyway (*first_name* will clearly hold a string, so nothing more is required).

For Boolean variables, try and state a fact that can be true or false, such as *is_paid*, or *is_registered*.

Suggest variable names for the following pieces of data. There are no right or wrong answers.

a. A variable holding the age of someone in years ____________________

b. A variable holding a street address ____________________

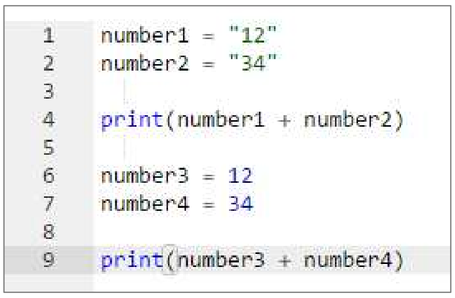
c. A variable holding the value *True* if someone is a member ____________________

d. A variable holding a *Date of Birth* ____________________

Data Types and Variables (page 2)

Task 3 – Text or Number?

Look at the program on the right. It takes two numbers and adds them together, then another two numbers and adds them together.



```
number1 = "12"
number2 = "34"

print(number1 + number2)

number3 = 12
number4 = 34

print(number3 + number4)
```

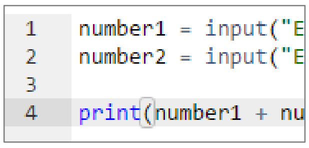
a. What do you think you will get for the two lines of output?

____

____

b. Create the program and find out what actually happens. Save as "***08.3 Text Number 1***".

c. Copy the code and paste it into a new session. Adapt it so that it accepts two numbers from the user through the console and adds these together. Are the numbers inputted through the console strings or integers? Save as "***08.3 Text Number 2***". We have shown part of our solution on the right.



```
number1 = input("E
number2 = input("E

print(number1 + nu
```

Task 4 – Data Type Conversions

It is often necessary to convert variables from one type of data to another. For example (as you found in the last task) any data entered into the console is accepted as a string. These strings may be fine if you are working with names or addresses, but they are not so good if you are trying to accept numbers for a calculation.

Strings can be converted into integers using the ***int*** function.

integer = int(string)

If, for example, you have a variable *first_number* holding the string "4", then the code *int(first_number)* will give you the integer 4. This can then be used in a calculation. The program below demonstrates this.

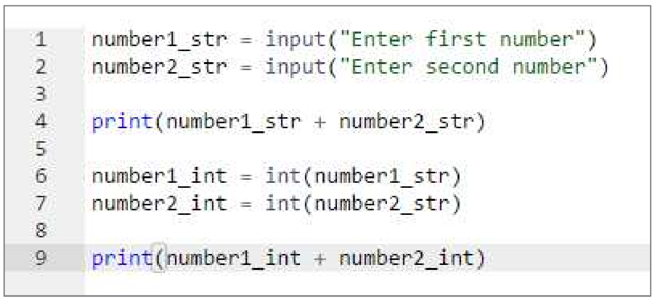
a. Create the program and test it with and without the *int* function on lines 6 and 7 i.e. try:

```
number1_int = int(number1_str)
number2_int = int(number2_str)
```

and

```
number1_int = (number1_str)
number2_int = (number2_str)
```

Save as ***08.4 int***.

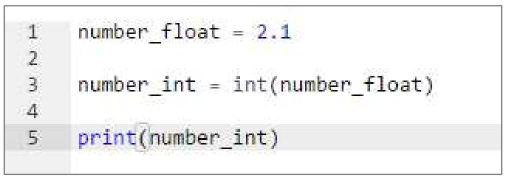


```
1  number1_str = input("Enter first number")
2  number2_str = input("Enter second number")
3
4  print(number1_str + number2_str)
5
6  number1_int = int(number1_str)
7  number2_int = int(number2_str)
8
9  print(number1_int + number2_int)
```

Note: In this case, we have named the variables with the extra _str or _int parts so that we remember what type of data is being held at different points in the program.

b. Write a similar program that uses the ***str*** function to change integers to strings. Display the effect that the function has using outputs to the console. You can fix the starting numbers in the program. Save as "***08.4b str***".

c. The program on the right changes a decimal number (called a floating-point number) into an integer. Create the program and see what happens when decimals such as 2.1, 2.5 and 2.9 are assigned in line 1. Save as "***08.4c int***".



```
1  number_float = 2.1
2
3  number_int = int(number_float)
4
5  print(number_int)
```