# Lesson 4 - Functions, Strings and Dates

|  |  |
| --- | --- |
| **Lesson Outcomes**  In this lesson you will learn:   * how to use basic common functions; * to incorporate functions into your programs; * how to format strings and number; * how to use and manipulate dates using the date object. | **C:\Users\Graham\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.IE5\X6CHINOH\MC900441498[1].png** |

## Introduction to Functions and Strings

Functions in any programming language are powerful tools. They allow us to perform common complex operations without having to write all the extra code. A function should have a **name** and with the option of adding **parameters** and a **return value**. For example, if we loop at the common function **str** in Python:

**Output = str ( 1203 )**

The PARAMETER e.g. a number

The function name

The above function takes a number as an input and will return the number as a **STRING**. This returned value is assigned to the variable Output. Further examples of functions we use all the time:



In this example the following functions are used: **float, input, print and str. “Float**” converts the text input to a decimal number, **“print”** outputs a string, **“str”** converts a number to a string and finally **“round”** takes a decimal number and rounds up to the given number of digits.

Python has many functions which are useful, here are just a few commonly used:

#### Maths functions:

|  |  |
| --- | --- |
| **Function** | **Description** |
| round() | round (value to round, [optional number of digits]) |
| math.floor() | Removes the digits past the decimal place. To use this function you have to use the import command. |
| math.trunc() |  |

#### String Functions / Handling

|  |  |
| --- | --- |
| **Function** | **Description** |
| len() | used to get the length of a string |
| *S.*upper()  (S is the string) | Converts string into ALL upper case |
| *S.*lower()  (S is the string) | Converts string into ALL lower case |
| string[2:2] | Extracting elements of a string |
| ‘Hi” in “Hi, how are you” | Find text in a string using the ‘in’ command: |
| string.find(‘test’) | Find text in a string and returning the index |

#### Random Numbers:

|  |  |
| --- | --- |
| **Function** | **Description** |
| random() and randomint(x,y) | To use this function you have to import random.    (Note the “#’ command, this tells Python to ignore anything after and is used for the programmer to add their own comments) |

### Tasks

4.1 Write a program that asks the user to type in a number with decimal places. The program should then display the rounded number to 2 decimal places.

4.2 Write a program that reads in a string and displays the number of characters in the string.

4.3 Write a program that asks the user for their surname and displays the surname in uppercase letters;

4.4 Write a program that will display random numbers between 1 and 6 until a six is generated

**HINT:** you will need to use a conditional loop.

4.5 Write a program that will display six random numbers between 5 and 10.

4.6 Write a game in which the user guesses what number between 1 and 1000 the computer has “thought of”, until he or she has found the correct number. The computer should tell them whether their guess is too high, too low or spot on.

## String Manipulation

In the above exercises you used some useful functions for numbers and Strings. However, Python has some extra functions which are unique to Strings and are used extensively for manipulating and extracting content.

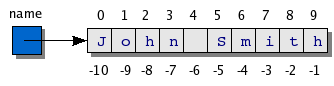
For example sometimes it is useful to only extract individual CHARACTERS of a string:



Character position

This program will output the first and last character of the string. The position of the character indicated by the number in square brackets.

You can imagine a string as a series of characters with each character given a number starting from 0 e.g. name=”John Smith” would be represented in the computer as:

Each character stored as an ASCII value.

As well as returning single characters you may also retrieve subsets of the string:



In this example the surname and first name are extracted from the name string. Surname starts at character position 5 and ends at 10, first name starts at 0 and ends at 4 (not including 4):



Alternatively, you can count from the end backwards to get the first name.

Python also provides a useful facility to duplicate characters, for example:



This outputs the character “-” sixty times.

### Tasks

4.7 Write a program that reads in a date of birth in the format **dd/mm/yy** and displays the year of their birth.

4.8 Write a program that asks the user for their first name and surname. The program then displays the full name, surname all uppercase followed by first name capitalised, regardless what case the user entered.

4.9 Write a program that asks the user to enter a sentence, terminated by a full stop and the pressing of the Enter key. The program should count the number of words and display the result.

**HINT:** create a loop which looks at the string character by character, keeping a running total of words. The word count is incremented when a space is detected.

4.10 Write a palindrome tester. A palindrome is a word or sentence that reads the same backwards as forwards. The user should enter a string and the program should display whether the string is a palindrome or not.

## Formatting Strings

Python provides facilities to allow you to format the output of your string, this is particularly useful when you want to produce layouts or tables in your outputted data. The formatting works by replacing placeholders within the string with values, for example:



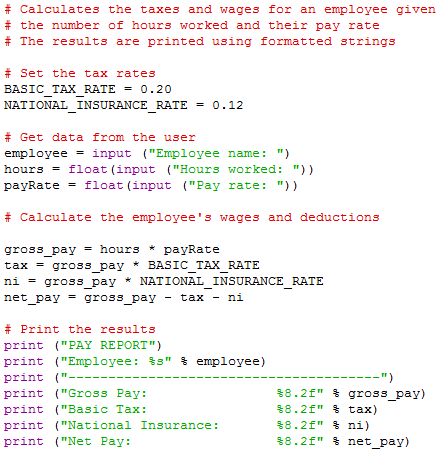
In the above the place holders are “%d” i.e. replace these values with the integers 300 and 400. The “\n” tells Python to add a carriage return e.g. next line.

The general structure is (after the % sign):

%[flags][width][.precision]code

Where

|  |  |
| --- | --- |
| *flags* | Used to indicate zero fills (0) which fills preceding blank spaces within the field with 0 and optional justification within the given field width: + for right-justification or - for left-justification. |
| *width* | An integer value indicating the number of spaces in the current field used when formatting the replacement value. |
| *precision* | The number of digits to be printed after the decimal place when printing a real value. |
| *code* | One of the format codes:   |  |  | | --- | --- | | **Code** | **Description** | | %s | String (or any object) | | %c | Character (from an ASCII value) | | %d | Decimal or integer value | | %i | Integer value (same as %d) | | %u | Unsigned integer | | %o | Octal integer | | %x | Hexadecimal integer | | %X | Same as %x but uppercase | | %e | Floating-point with exponent (decimal numbers) | | %E | Same as %e but uppercase | | %f | Floating-point no exponent (decimal numbers) | | %g | Same as %e or %f | | %G | Same as %g but uppercase | | %% | Prints a literal % | |

Whilst this looks fairly complicated it is relatively straightforward. Just remember the placeholders are codes which can be any of the ones listed above. An example program:

The first placed holder is a simple string (%s) and the other values (%8.2f) are displayed in decimal, with 8 digit width followed by 2 decimal places, the f indicating a floating point number (decimal number).

### Tasks

4.11 Write a program which prompts the user for a number between 1 and 12, and then displays the times table to 12 of that number e.g. user enters 2, so display the 2s times table. The table should be formatted with one column for the times table numbers and the other column for the answers.

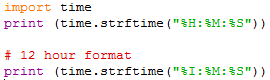
4.12 Write a program which ask for their surname, first name, DOB, tel. no and address. Once the data has been entered display in the following format (us – and| for basic table formatting):

|  |  |  |
| --- | --- | --- |
| Surname:  Forename:  Address: | Dob:  Town/ City: | Tel: No:  Postcode: |

## Dates

As you have seen from previous exercises, the ability to process strings is important. As well as these built-in functions sometimes it is useful to import MODULES. One such module is **DateTime** for processing and manipulating dates. As part of the **DateTime** module there are different classes (built in parts of the module):

* date – Just a date. *(Month, Day, Year)*
* time – Time independent of day. *(Hour, Minute, Second, Microsecond)*
* datetime – Combination of date and time. (Month, Day, Year, Hour, Second, Microsecond)
* timedelta – A duration of time used for manipulating dates
* tzinfo – An abstract class for dealing with timezones

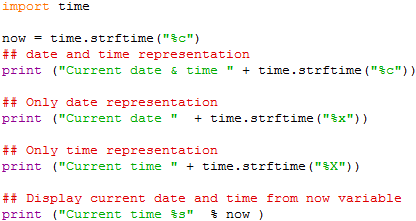
To use a module simply use the import command at the beginning of your program. The following example displays the current time:

This should produce the following outputs:

23:46:08

11:46:08

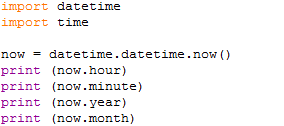
To extract data and time you can use the ***strftime*** function as part of the time import. This is similar to formatting text using codes to display different parts of date/time.

The format the date output you uses the following codes:

|  |  |  |
| --- | --- | --- |
| **Directive** |  | **Meaning** |
| %a |  | Weekday name. |
| %A |  | Full weekday name. |
| %b |  | Abbreviated month name. |
| %B |  | Full month name. |
| %c |  | Appropriate date and time representation. |
| %d |  | Day of the month as a decimal number [01,31]. |
| %H |  | Hour (24-hour clock) as a decimal number [00,23]. |
| %I |  | Hour (12-hour clock) as a decimal number [01,12]. |
| %j |  | Day of the year as a decimal number [001,366]. |
| %m |  | Month as a decimal number [01,12]. |
| %M |  | Minute as a decimal number [00,59]. |
| %p |  | Equivalent of either AM or PM. |
| %S |  | Second as a decimal number [00,61]. |
| %U |  | Week number of the year (Sunday as the first day of the week) as a decimal number [00,53]. All days in a new year preceding the first Sunday are considered to be in week 0. |
| %w |  | Weekday as a decimal number [0(Sunday),6]. |
| %W |  | Week number of the year (Monday as the first day of the week) as a decimal number [00,53]. All days in a new year preceding the first Monday are considered to be in week 0. |
| %x |  | Appropriate date representation. |
| %X |  | Apropriate time representation. |
| %y |  | Year without century as a decimal number [00,99]. |
| %Y |  | Year with century as a decimal number. |
| %Z |  | Time zone name (no characters if no time zone exists). |
| %% |  | A literal '%' character. |

Sometime it is useful to store parts of the Date/ Time into variables so they can be used in calculations:

This can be done easily:

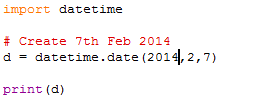


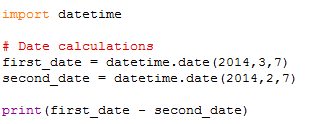
So using any of the above in a program:

**print** ("Current year = %s" %now.year)

More information on time/date functions can be found: <http://pymotw.com/2/datetime/>

## Manipulating Dates

You can also use the date/time object to create your own dates e.g.:

Using this method allows you to manipulate dates using simple formula e.g.

Should output:

Let’s add 3 days to a date object.

|  |  |
| --- | --- |
|  |  |

This will output:

|  |  |
| --- | --- |
|  |  |

### Tasks

4.13 Write a program that displays today’s date in the format dd/mm/yy.

4.14 Write a program that reads in a date of birth in the format dd/mm/yy and displays the user’s age.

4.15 Write a program that reads in a date, converts into date format, adds a day and then displays the next day’s date.