

Lab 3 – Flow

if/elif/else
while/break/continue/else
operators

In programming and scripting languages, conditional statements or conditional constructs are used to perform different computations or actions depending on whether a condition evaluates to *true* or *false*. (Please note that true and false are always written as True and False in Python.)

The condition usually uses comparisons and arithmetic expressions with variables.

These expressions are evaluated to the Boolean values True or False.

The statements for the decision taking are called conditional statements, alternatively they are also known as conditional expressions or conditional constructs.

The if-then construct (sometimes called if-then-else) is common across many programming languages, but the syntax varies from language to language.

Donald Keidel, Ph.D. 2016

The general form of the if statement in Python looks like this:

```
if condition_1:  
    statement_block_1  
elif condition_2:  
    statement_block_2  
else:  
    statement_block_3
```

If the condition "condition_1" is True, the statements in the block statement_block_1 will be executed.

If not, condition_2 will be executed.

If condition_2 evaluates to True, statement_block_2 will be executed.

If condition_2 is False, the statements in statement_block_3 will be executed.

In order for comparisons to be made, you will need to learn and use a few more operators.

1. Boolean Or, Boolean And, Boolean Not (Logical Operators)

Operator	Description	Example
and	Called Logical AND operator. If both the operands are true, then the condition becomes true.	(a and b) is true.
or	Called Logical OR Operator. If any of the two operands are non zero, then the condition becomes true.	(a or b) is true.
not	Called Logical NOT Operator. Use to reverse the logical state of its operand. If a condition is true, then Logical NOT operator will make it false.	not(a and b) is false.

2. The usual comparison (relational) operators

Operator	Description	Example
==	Checks if the value of two operands is equal or not, if yes then condition becomes true.	(a == b) is not true.
!=	Checks if the value of two operands is equal or not, if values are not equal then condition becomes true.	(a != b) is true.
<>	Checks if the value of two operands is equal or not, if values are not equal then condition becomes true.	(a <> b) is true. This is similar to != operator.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(a > b) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(a < b) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(a >= b) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(a <= b) is true.

```

1  '''
2  dog_years_if_elif_else.py:
3
4  asks the user to input age of a dog and converts them
5  to human years
6
7  '''
8
9  age_input = raw_input('Age of dog?')
10 try:
11     age = int(age_input)
12 except ValueError:
13     print 'The number input',age_input,'is not a valid integer'
14 else:
15     print 'The number input',age_input,'is a valid integer'
16
17     if age <= 0:
18         print "This can hardly be true!"
19     elif age == 1:
20         print "about 14 human years"
21     elif age == 2:
22         print "about 22 human years"
23     else:
24         human = 22 + (age - 2) * 5
25         print "Human years: ", human
26

```

Donald Keidel, Ph.D. 2016

```

>>> ===== RESTART =====
>>>
Age of dog?5
The number input 5 is a valid integer
Human years:  37
>>> |

```

In some programs the programmer will need to carry out a sequence of statements *repeatedly*.

The code within the loop, i.e. the code carried out repeatedly is called the body of the loop.

Python supplies two different kinds of loops: the *while* loop and the *for* loop.

Most loops contain a *counter* variable that changes in the course of the calculation.

These variables have to be *initialized* (declared) before the loop is started.

The counter or other variables, which can be altered in the body of the loop, are *contained* in the condition.

Before the body of the loop is executed, the condition is evaluated.

If it evaluates to True, the body gets executed.

After the body is finished, the condition will be evaluated again.

The body of the loop will be executed as long as the condition yields True.

Donald Keidel, Ph.D. 2016

```
1 '''
2 sum_1_to_100_while_loop.py:
3
4 The following small script calculates the sum of
5 the numbers from 1 to 100
6
7 '''
8
9 number_to_count_to = 100
10
11 sum_of_numbers = 0
12 initial_value = 1
13
14 while initial_value <= number_to_count_to:
15     sum_of_numbers = sum_of_numbers + initial_value
16     initial_value = initial_value + 1
17
18
19
20 print "Sum of 1 until", number_to_count_to, "is", sum_of_numbers
21
```

```
>>> ===== RESTART =====
>>>
Sum of 1 until 100 is 5050
>>> |
```

Donald Keidel, Ph.D. 2016

```
while_loop_with_true_false_break.py - /Users/donaldk/Desktop/Python_Beginners_Course_Materials/while_loop_with_true_false_break.py
1 '''
2 while_loop_with_true_false_break.py:
3
4 This is a script containing a while loop
5 and it also uses the keywords:
6
7 1. True or False
8 2. break
9
10 '''
11
12 while True:
13     age = raw_input('Give me your age; type "quit" to end.\n')
14
15     if age == 'quit':
16         break
17     try:
18         int_age = int(age)
19         print 'Your age is', int_age
20     except:
21         ValueError
22         print 'Your age must be an integer value'
23
24
25
```

Code Browser Ln: 13 Col: 0

```
Python 2.7.5 Shell
>>>
Give me your age; type "quit" to end.
10.5
Your age must be an integer value
Give me your age; type "quit" to end.
23
Your age is 23
Give me your age; type "quit" to end.
45
Your age is 45
Give me your age; type "quit" to end.
quit
>>>
```

GUI: OFF (TK) Ln: 14 Col: 4

Lab 3 – Exercises

1. Conditional Statements combined with operators (comparison or relational, assignment, arithmetic, and logical). For each sentence, translate the sentence to Python code using your operators:

if i is from 2 to 78, subtract 2 from i

if j is not between 10 and 15 but is the number 11, then k equals j times 2

if k is not equal to 7 and i equals 5 or j is greater than 8, set x equal to 200 divided by k

if i is the sum of j and k, and j is not equal to sum of i and k, then x equals j times k divided by j

Donald Keidel, Ph.D. 2016

2. Write a Python program that asks the user to enter an integer that is greater than 0. The program will keep on asking the user for the number until it is valid.
3. Review the following statements. If there is an error, please indicate where the error exists and how to correct it.

if x < 5 or x = 6:

if x < 0 and x > 5:

if i == 12 or i not 6:

Lab 4 – Functions

A function is a block of *organized, reusable* code that is used to perform a single, related action.

Functions provide better *modularity* for your program and they make your program more reusable.

Your code is more easily debugged, code will not repeat in many locations in program, allows for changes to be made in one location of code, code will be more organized, and you can use this function in other locations in program.

There are many *built in* functions in Python, but you can also create your own functions. These are called *user-defined functions*.

Here are some simple rules to define a function in Python.

1. Function blocks begin with the keyword **def** followed by the function name and parentheses (`()`).
2. Input parameters (arguments) are placed within these parentheses. Parameters can also be defined in the parenthesis. Not all functions require arguments. If there are no arguments, the parenthesis are empty.
3. The first statement of a function can be optional.
4. The code block within every function starts with a colon (`:`) and is indented.
5. The return statement exits a function. It can return a value. A return statement with no arguments is the same as return None. A return is not needed and will return None if not present.

Syntax:

```
def FunctionName( parameters ):
    """ docstring here is optional – describes what function does"""
    function code block

    return [expression] - optional
```

By default, parameters have a *positional* behavior. The programmer needs to inform them in the same order that they were defined.

Example:

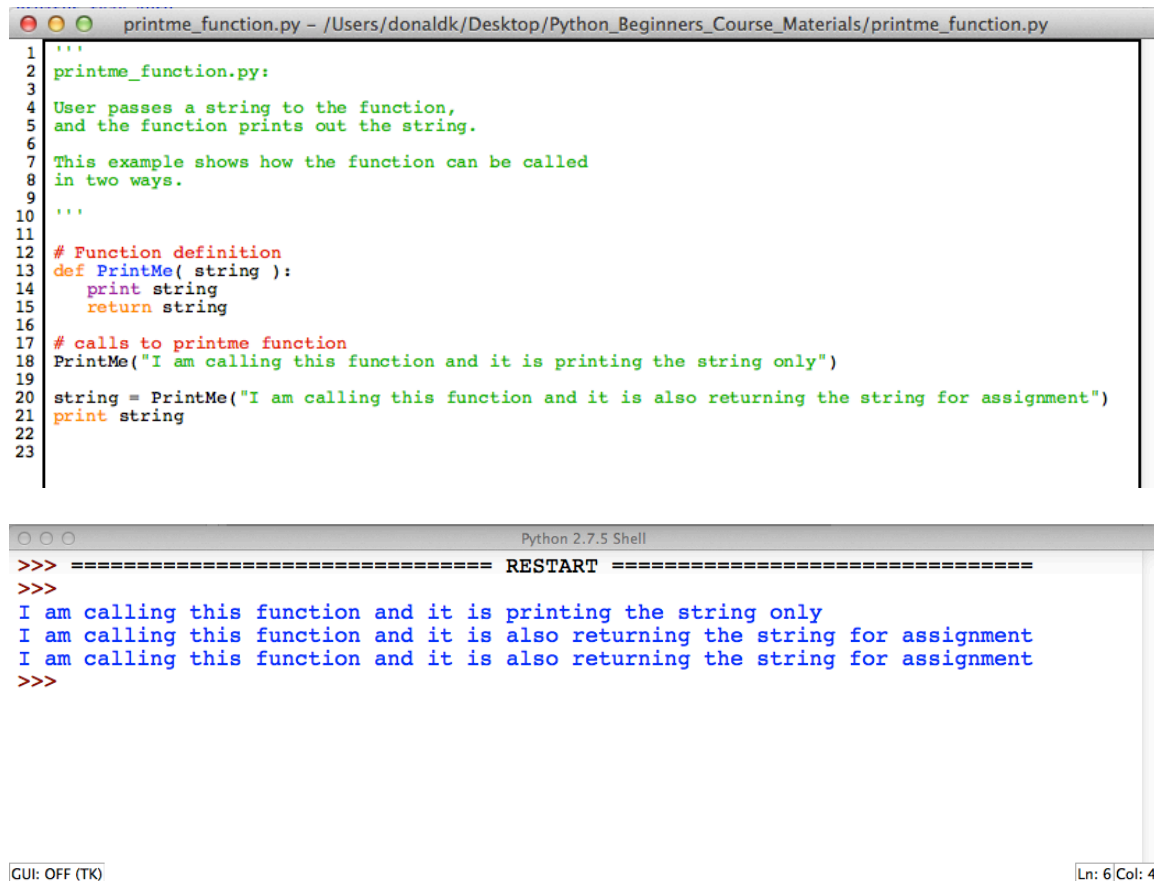
The function takes a string as input parameter and prints it to standard out.

```
def PrintMe( string ):
    "This prints a passed string into this function"
    print string

    return
```

Calling a Function:

Once the basic structure of a function is complete, you can execute it by calling it.



The image shows two windows. The top window is a text editor titled 'printme_function.py - /Users/donaldk/Desktop/Python_Beginners_Course_Materials/printme_function.py'. It contains a Python script with a function definition and two calls to the function. The bottom window is a 'Python 2.7.5 Shell' showing the output of the script after a restart. The output consists of three lines of text, each preceded by a prompt 'I am calling this function and it is printing the string only'.

```
1 '''
2 printme_function.py:
3
4 User passes a string to the function,
5 and the function prints out the string.
6
7 This example shows how the function can be called
8 in two ways.
9
10 '''
11
12 # Function definition
13 def PrintMe( string ):
14     print string
15     return string
16
17 # calls to printme function
18 PrintMe("I am calling this function and it is printing the string only")
19
20 string = PrintMe("I am calling this function and it is also returning the string for assignment")
21 print string
22
23
```

```
>>> ===== RESTART =====
>>>
I am calling this function and it is printing the string only
I am calling this function and it is also returning the string for assignment
I am calling this function and it is also returning the string for assignment
>>>
```

GUI: OFF (TK) Ln: 6 Col: 4

Remember, when calling the function you must have the *same number* of variables as are required in the function definition. Also, the *order* in which they are passed in the call is important also.

```
convert_dollars_to_euros_functions.py - /Users/donaldk/Desktop/Python_Beginners_Course_Materials/convert_dollars_to_euros_functions.py
1 '''
2 convert_dollars_to_euros_functions.py:
3
4 Use of docstrings: A docstring is a string
5 literal that occurs as the first statement in
6 a module, function, class, or method definition.
7
8 Program using functions that repeatedly asks user
9 for valid US dollars amount (float) and once valid
10 amount given will convert that to euros using the
11 current exchange rate at time of writing
12
13 '''
14
15 EURO_CONVERSION_RATE = 1.36
16
17 def GetDollars():
18     """Asks user for dollar amount as float
19     and returns a verified float
20     """
21     while True:
22         us_dollars = raw_input('Enter dollar and cents value to convert.\n')
23         try:
24             float_us_dollars = float(us_dollars)
25         except:
26             ValueError
27             print us_dollars, "is not a valid dollar amount. Try again."
28             continue
29         return float_us_dollars
30
31 def ConvertDollarsEuros(us_dollars):
32     """Converts the parameter us_dollars to euros
33     """
34     euros = us_dollars / EURO_CONVERSION_RATE
35     return euros
36
37
38
39 def Run():
40     """Run function that will run GetDollars and ConvertDollarsEuros
41     functions and print out the result
42     """
43     verified_us_dollars = GetDollars()
44     print 'You will be converting', verified_us_dollars, 'dollars to
45     euros'
46     print verified_us_dollars, 'converted to euros is', ConvertDollarsEuros(verified_us_dollars)
47
48 Run()
49
50
51
```

Code Browser Ln: 11 Col: 40

```
Python 2.7.5 Shell
>>> ===== RESTART =====
>>>
Enter dollar and cents value to convert.
five dollars
five dollars is not a valid dollar amount. Try again.
Enter dollar and cents value to convert.
$4.50
$4.50 is not a valid dollar amount. Try again.
Enter dollar and cents value to convert.
4.50
You will be converting 4.5 dollars to
euros
4.5 converted to euros is 6.12
>>> |
GUI: OFF (TK) Ln: 15 Col: 4
```

Compare to `convert_dollars_to_euros_functions_out_of_order.py` (shown in class). This program has the call to `Run()` before the final function is defined. The interpreter will complain that: `NameError: name 'Run' is not defined`.

Functions can also call other functions.

```
function_calling_other_functions.py - /Users/donaldk/Desktop/Python_Beginners_Course_Materials/function_calling_other_functions.py
1 '''
2 function_calling_other_functions.py:
3
4 Example of functions called by other functions. This
5 script will take a number from the user, add them
6 together, multiple the addition result together, and then
7 try to print the result.
8
9 Also introduced is the stack trace.
10
11 A stack traceback is the result of determining the sequence
12 of nested calls a program has made up to a
13 certain point in its execution.
14 '''
15
16 def PrintResult(result):
17     print result
18
19 def MultiplyNumbers(mult_number):
20     mult_result = mult_number * mult_number
21     PrintResult(mult_result)
22
23 def AddNumbers(add_number):
24     add_result = add_number + add_number
25     MultiplyNumbers(add_result)
26
27 number = raw_input('Give me a number: ')
28 AddNumbers(number)
29
30
31
32
33
34
35
36
37
38
```

Code Browser Ln: 6 Col: 57

```
Python 2.7.5 Shell
>>>
Give me a number: 5

Traceback (most recent call last):
  File "/Users/donaldk/Desktop/Python_Beginners_Course_Materials/function_calling_other_functions.py", line 34, in <module>
    AddNumbers(number)
  File "/Users/donaldk/Desktop/Python_Beginners_Course_Materials/function_calling_other_functions.py", line 30, in AddNumbers
    MultiplyNumbers(add_result)
  File "/Users/donaldk/Desktop/Python_Beginners_Course_Materials/function_calling_other_functions.py", line 24, in MultiplyNumbers
    mult_result = mult_number * mult_number
TypeError: can't multiply sequence by non-int of type 'str'
>>>
```

GUI: OFF (TK) Ln: 13 Col: 4

Lab 4 – Exercises:

1. Write a Python script or program that contains two functions. The first one when called will ask the user for a price. It will check to make sure that what is input is a number. If it is not, it will ask for a valid number. When it is determined that a valid number is input, a second function will be called that will output only the dollar value. e.g. if the price is \$2.65, 2 will be printed to screen.
2. Write a function that you will call in a Python script that will ask the user for the number of people that will attend a party you are throwing. Also ask the user for the number of drinks you will provide and the number of pizzas. Have this function call another function that will determine if you can have the party or not. This function should be called the following way: `is_a_party(num_people, num_drinks, num_pizzas)`. The criteria to determine if you can have the party is the following:

4 drinks minimum for each guest

One quarter of a pizza for each guest

Donald Keidel, Ph.D. 2016

3. In the program in number 2 indicate what happens if you call `is_a_party(num_pizzas, num_drinks, num_people)`?

Homework 2:

Write a program that asks the user for a letter. The program should then determine if the letter is a vowel or not.

In this homework you will, of course, be writing functions to perform the individual components of the program.

For this assignment, you should have the following functions:

1. First function: `AskForLetter()`: This function will repeatedly ask the user for a single letter until the user types 'quit' to exit the program or they have entered a vowel. This function will use the built-in python function called `len`. `len` is used to determine the length of an input. For example, if the input from the user is 'i', `len('i')` will return 1. Try this in python interpreter. If the `len` of the user input is 1, then this function should call the second function below. Finally this function outputs the input letter if it is a vowel.
2. Second function: `IsVowel(letter)`: This function will take as input a variable called `letter` and will determine if this letter is a vowel or not. If it is a vowel it will return `True` and if not it will return `False`. To determine if letter is vowel or not, this function will call a third and a fourth function below.
3. Third function: `IsLowercaseVowel(letter)`: this function will take a single variable as input called `letter` and will return `True` if the letter is an lowercase vowel.
4. Fourth function: `IsUppercaseVowel(letter)`: this function will take a single variable as input called `letter` and will return `True` if the letter is an uppercase vowel.

Make sure you account for user input error. Test your code by trying to cause errors in input.

Print out a copy of your homework and bring it to class. Make sure to include the output.