Lab 3 – Condition & Loop statement and Defining Function

This lab will cover the semantics of the Python language. As opposed to the syntax, the semantics of a language involve the meaning of the statements.

Ref.: https://jakevdp.github.io/WhirlwindTourOfPython/

As with Lab 2, we don't expect you to finish all of the material here in one class period. This lab is designed to be long! You are encouraged to work through the extra material at your own pace - it explores interesting and intriguing aspects of Python functions.

Above all, have fun playing with Python! Enjoy.

Control Flow

Control flow is the ordering in program execution. Without it, a program is simply a list of statements that are sequentially executed. Using control flow, you can execute certain code blocks conditionally and/or repeatedly. Here we will cover conditional statements and loop statements.

Conditional statements: `if-elif-else`

The if-then statements allow the programmer to execute certain pieces of code depending on some Boolean condition.

a) Let's have a look at a simple if-then statement. Copy the following program into a number.py file. Use breakpoints and step through the code to examine the program.

```
number = input("Enter a number: ")

if number >= 22:
     print("if")
else:
     print("else")
```

Now add another condition that print "elif" if the number is greater than or equal to 21. Use breakpoint and step through the code to trace the variables.

Note that the elif and else blocks are optional. You may opt to include as few or as many elif statements as you would like.

b) Practice: Write a program called colour.py that asks a user to enter a colour name (i.e. blue, red, yellow). If the colour entered is your favourite colour, print a message to that effect. Otherwise, print a different message when it is not your favourite. The program should also print a final message for every user regardless of what they entered.

`while` loops

Loops in Python enable us to perform repetitive tasks or in another word, to execute one or more statements repeatedly.

a) Copy the following program into a whileLoops.py file. Use breakpoints and step through the code to examine the program. Note the value of `count` variable at each iteration.

```
count = 1

while count <= 10:
    print(count, end=" ")
    count +=1
```

What will happen if you remove the statement “`count +=1`”? Remove the statement to verify your prediction.

*In programming, a code performing similar task can be written in multiple ways. Note that the above code can be written using the `for` loop construction as well.

Now modify the code by making use of `continue` to print only the odd integers.

b) Practice: Write a program called squares.py that prints a list of the first nine positive integers and their squares. Your output should look similar to the following:

```
Number      Square
1              1
2              4
3              9
4              16
5              25
6              36
7              49
8              64
9              81
```

`for` loops

The other type of loop is a `for` loop. For example if you want to print each of the items in a list, consider using for loop.

a) Copy the following program into a forLoops.py file. Use breakpoints and step through the code to examine the program. Note the similarity with the while loop example code.

```
for integer in range(10):
    print (integer, end=" ")
```

In order to view the sequence of numbers generated by range, use list function. Example: `list(range(5)), list(range(1,5)), list(range(1,5,2))`
You might notice that the range starts at zero by default.

b) Loop over the elements of a list (Recall: Lab 2 Data Structure)
`for` loop is awesome where you can use it to cycle through a list. For example:

```
fruits = ['apple', 'orange', 'durian']
for fruit in fruits:
    print(fruit)

num = [1, 2 , 3]
squares = []
for n in nums:
    squares.append(n**2)
print(squares)
```

Tips: You can make use of the `enumerate` build-in function in Python to access the index of each element within a loop. Try it out yourself.

List comprehensions: you can simplify this code by using list comprehension.

```
num = [1, 2 , 3]
squares = [n**2 for n in nums]
print(squares)
```

Do explore comprehension for other data structures (i.e. tuples, sets and dict) on your own.

Defining Function

Just like any other programming languages, user-defined functions in Python are useful to separate your code. You can also put a block of code in a function if you think you will be using it anywhere else. Python functions are defined using the `def` keyword.

a) Let's have a look at an example user-defined Python function that determines the sign of an integer. Use breakpoints and step through the code to examine the program.

```
def sign(num):
    if num > 0:
        return 'positive'
    elif num < 0:
        return 'negative'
    else:
        return 'zero'

for x in [-1, 0, 1]:
    print(sign(x))          Calls to function
```

The function named `sign` takes a single argument `num`, does something with this argument and returns the corresponding string.

Now try calling the function using `sign(1, 2).` It will give you an error because the function is only defined to take one arguments.

b) Default argument value
Sometimes there are certain values that we want the function to use most of the time, but we would like to give user some flexibility. We can provide this flexibility as follows:

```
def welcome(name, loud=False):          Default/Optional argument
    if loud:
        print('WELCOME, %s!' % name.upper())
    else:
        print('Welcome, %s' % name)

welcome('Jane')
welcome('John', loud=True)
```

There is a lot more information about Python functions in the documentation. Do explore!

Lab Task

1 Write a program called combi.py that prints the output shown below. The program should print the numbers 1 through 10, except that between 5 and 6 it should print the word "Python".

```
1
2
3
4
5
Python
6
7
8
9
10
```

2 Write a program which prompts the user for 10 floating-point numbers and calculates their sum, product, and average. Your program should only contain a single loop.

What to submit
Submit only **ONE** Python file (i.e. just the .py file). The filename should follow the following standard: <StudentMatrixNumber>.py

Written by Kok Ven Jyn 2018 (rev. 2022)