

## COM4013 – Introduction to Software Development

### Week 4 – More Loops, User Input Validation

*Always select “Enable Editing” if prompted by Microsoft Word*

#### Lab Exercises

The exercises in this lab follow the topics in the lecture notes. Refer to the lecture notes for examples and guidance.

There are two "stages" marked in the lab sheet. Stage 1 is the *absolute minimum* point you should reach. Hopefully, you can reach it in the lab session. Reaching stage 2 suggests you are on target for a top-class mark for the module. Set your goals sensibly - do not just aim for the minimum or you may struggle to pass the module.

**I use the term integer, string and float, and Boolean to infer what the input/output looks like. Unlike many other languages Python does not expect a data type before the variable names.**

So if I ask that you declare an integer num1 to 1, then do as follows:

```
num1 = 1
```

For declaring a string and assigning the value hello

```
greeting = "hello"
```

For floats we can declare it like this

```
Money = 2.50
```

For Booleans (bool) we can declare it like this

```
isHeavy = True
```

#### 'for' Loops

- ◆ Create a new Jupyter Notebook project called **Lab 4 Exercises**. Refer to Week 1's lecture/worksheet/video if you have forgotten how to create a new project/file/program or use the shell code.
- ◆ Write a 'for' loop that displays the numbers 1 to 10.  
**Refer to the lecture notes, they contain examples of how to write a 'for' loop.**

- ◆ Using `end= ' '` syntax (that I introduced in the last session) in your print statement, to make the output look like this (all numbers on the same line separated by commas):

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

### More Simple 'for' Loops

- ◆ Write the following two 'for' loops in the same cell. Separate each answer using a new print statement (outside of the for loops scope) with a newline Unicode character:
- ◆ Display the odd numbers from 9 to 49 – comma separated
- ◆ Display the even numbers from 100 down to 50 in reverse order, *but not including 50* – comma separated

### Text Input Validation

Create a new cell in Jupyter Notebook (use the shell code).

- ◆ It is very common to ask the user a Yes/No question:

```
Are you sure you want to delete the record? (y/n)
```

Write a program to ask this question and read the answer from the user. The program should repeatedly ask again if the user types something invalid. **There is an example of text input validation in the lecture notes.**

- ◆ After the loop display an appropriate message depending on the user's choice. This needs an 'if' statement. The output for your program might read like this:

```
Are you sure you want to delete the record? (y/n) xyz
Sorry, you must answer 'y' or 'n'.
```

```
Are you sure you want to delete the record? (y/n) n
Record was not deleted.
```

```
Are you sure you want to delete the record? (y/n) y
Record was deleted.
```

- ◆ We should allow the user to type 'y' or 'n' in lower case or upper case. Make the appropriate changes to your program to allow this. Again read the lecture notes.

Make sure you test your program thoroughly with all possible inputs. Try to break your program with unexpected inputs!

## Numeric Input Validation

Create a new cell in Jupyter Notebook (use the shell code).

- ◆ Ask the user for the year of their birth and put the result in an integer variable. The valid range for the years is from 1900 (inclusive) to 2011 (inclusive). If the user types a year outside that range *or* if the user types something that is not a number, then the program should ask the user to try again. **Again refer to the lecture notes for an example.**
  - HINT: Use a do-while like construct in conjunction with a try-except block.
  - This may be difficult to complete

- ◆ When you have a valid input display their age (approximate as we don't know their exact birthday). Example output:

```
What year were you born? 2048
Sorry, please enter a valid year.
```

```
What year were you born? Not telling you...
Sorry, please enter a valid year.
```

```
What year were you born? 1990
You are about 21 years old.
```



Stage 1

Keep Going...!

## Calculator

Create a new cell in Jupyter Notebook (use the shell code).

This exercise tests your ability to work with a program as it becomes complex. It is easy to make small mistakes when copying earlier work. Tips:

- Be careful with variable names.
- Comment sections of code, you can put lots of blank lines between parts if you like

- Try not to get confused as the program becomes longer and more complicated!
- ◆ Copy this code into the shell code. It's an example of adding two numbers together.

```
while (True):
    userNum1 = float(input("Enter first number: "))
    userNum2 = float(input("Enter second number: "))

    userTotal = userNum1 + userNum2
    print(f"{userNum1} + {userNum2} = {userTotal}")

    yesNo = input("Do you want another go (y/n): ")

    if (yesNo == 'n'):
        break
```

- ◆ Add comments to this code – I suspect **MOST** of you will not take the time to do this...
- ◆ Add user input validation to the program (read the two points below before beginning).
- ◆ The numbers that are input must be checked with try-except, but can be in any range. Again reuse the appropriate parts of the earlier solution.
- ◆ The yes/no user input is almost identical to the earlier exercise, but I would like for you to use a regular while loop to do this. Be careful, as you may find that you are not able to escape from the loop.
  - Apply the .lower() function to your yesNo input variable.
  - Use a break statement to do this.
- ◆ At the start of the program, ask the user what operation they want to perform:
- ◆ Which operation do you want + or - ?
- ◆ Validate their input (a string containing "+" or "-") – use a simple while loop here.
- ◆ Update the calculation and answer display to use the user's choice of operator.
- ◆ As an extra challenge, modify the code to also allow for multiplication and division.

## Calendar Program - Prototype

Now we will make a program to display a calendar of a given year. This has some very tricky parts, so we start with a prototype (a simple test version).

We will need to know the number of days in a particular month / year. The Python calendar library contains a method to get this, so we don't have to program this part ourselves. The following code returns the number of days in October (month 10) of 2011, the result 31 will be put into the `numberDays` variable:

```
import calendar
numberDays = calendar.monthrange(2011, 10)[1] # Year and month
print(numberDays)
```

- ◆ Create a new cell in Jupyter Notebook (use the shell code).
- ◆ Copy and run the code to see that it works.
  - Calendar is a part of the Python standard library so you do not need to conda/pip install the library.
- ◆ Create a new cell in Jupyter Notebook (use the shell code).
- ◆ Ask the user for a year. Validate that the year is a number and in the range 1 to 9999. Repeatedly ask the user until they enter a valid value.

```
Enter a year: t
Sorry, please enter a valid year.
```

```
Enter a year: 10000
Sorry, please enter a valid year between 1 and 9999.
```

- ◆ Use a 'for' loop to display for each month, the month number and the number of days in the month:

```
Enter a year: 2011
Month 1
31 days
```

```
Month 2
28 days
...
```

## Names for Months and Days

We want to show the month and day as names rather than numbers. We could do this with lots of 'if' statements, but we'll use the Python libraries again to save us that work.

- ◆ `datetime` is also a part of the Python standard library so you do not need to conda/pip install the library.

This is how to get the text name of the month and day from a given date (assuming you have set up the `year`, `month` and `day` variables):

```
from datetime import datetime

year = 2023
month = 10
day = 15      # This can be any day

# Always using day = 1 when getting month name
monthName = datetime(2023, 10, 1).strftime("%B")

# Short day name (e.g. Mon Tue Wed)
shortDayName = datetime(year, month, day).strftime("%a")

# Full day name (e.g. Monday Tuesday Wednesday)
dayName = datetime(year, month, day).strftime("%A")

print(f"Month Name:      { monthName }")
print(f"Short Day Name: { shortDayName }")
print(f"Full Day Name:   { dayName }")

# Output
# Month Name: October
# Short Day Name: Sun
# Full Day Name: Sunday
```

- ◆ Create a new cell in Jupyter Notebook (use the shell code) and copy the month for loop that you created before.
- ◆ Within the month 'for' loop create another 'for' loop that count through the days in that month. You have already found the number of days, so you have the value that this new 'for' loop should count up to.
- ◆ Remember: For loop range ends are exclusive so you do need to account for this.
- ◆ For each day, display its name and number. Use simple and sensible variable names as you build up this program.

The output required is this, follow it carefully:

```
Enter a year: 2011
January
Sat 1, Sun 2, Mon 3, Tue 4, ...

February
Tue 1, Wed 2, ...
```



## Stage 2

~Boss Battle~

- ♦ Find the day of the week you were born...
  - Ask for the day, month, and year of birth
  - Validate the inputs – whichever way you'd like
  - Print their day of birth (Monday, Tuesday, etc.). I've give you the code to retrieve this using the year, month, and day variables.

You were born on a **Tuesday**

### Calendar Program - To Completion

- ♦ Rest assured that we will return to this in our next session...
- ♦ (I have two more advanced tasks in mind for this   ).

**THAT IS ALL FOR NOW**

### Submission

1. Please refrain from submitting your work to the Teams area. Last week marked the final instance in which I will request you to submit your practical session assignments. If you have any doubts about your work, you can continue submitting, but kindly seek feedback during the session. Given that some of you possess advanced programming skills, it might be more beneficial to review my solutions and compare them to your own understanding.