# Python Programming (CIS1028-N) Tutorial 2: Program Flow Control

## Before You Start

Ensure you have completed all tasks from Tutorial 1. Any issues seek help from your tutors.

**Hint:** It is good practice to plan your programs first on paper using pseudocode. When we say pseudocode, we mean code that is halfway between English and a programming language, such as Python.

Read more about it here: <https://en.wikipedia.org/wiki/Pseudocode>

Attempt to complete this set of tasks before your next tutorial.

## Introduction

This session aims to familiarise you with flow control, lists and iteration.

## Question 1: Comparing Numbers

Write a program to compare two numbers (integers).

1. On paper, devise a routine using pseudocode to ask the user for two numbers, compare them and then print a message indicating which one is bigger than the other (or if they are both the same) with an appropriate message.
2. Carefully work out the structure of your conditional (if) statement before you begin implementation of your procedure.
3. Now implement your algorithm in Python!

## Question 2: Eligible Voters

Write a program to check if someone is eligible to vote.

1. Create a program to ask the user for their name, age and country of citizenship. Your program should then process this information to determine if the named person can vote according to the following rules:
   1. If the person is under 18, they are not allowed to vote. Print out a message which includes their name and states that they are not yet old enough to vote.
   2. If the person is over 18 and their country of citizenship is the UK or Britain, print out a message including the name that states they are entitled to vote.
   3. Otherwise, if they are over 18 but their country of citizenship is not the UK or Britain, print out a message stating that they are not entitled to vote.
2. Test your program with suitable test cases.

## Question 3: Fizz Buzz

Create a program that asks the user for number and then outputs a message based on the properties of that number:

1. Ask the user for a number and store it in a variable.
2. If the number is divisible by 3, print the message Fizz.
3. If the number is divisible by 5, print the message Buzz.
4. If the number is divisible by both 3 and 5, print the message Fizz Buzz.
5. If the number is not divisible by either 3 or 5, print the number itself.

**Hint:** Think about how you will test if a number is divisible by another number. Do some research on the modulo operator (%) and think carefully about how it can help us here. Plan your conditional statements out on paper before you get started coding.

## Question 4: Sorting

You’re tasked with writing a program that sorts 3 numbers.

1. Create a program to ask the user for three numbers (called first\_num, second\_num, third\_num) and print them out in order from smallest to largest.
2. Test your code using the following test cases. You must enter the numbers in the exact order given for each test case. So, for the first test case (*test case a*); first\_num = 87, second\_num = 99, and third\_num = 11. Run your program for each test case to ensure it works correctly.
   1. 87, 99, 11
   2. 87, 11, 99
   3. 99, 87, 11
   4. 99, 11, 87
   5. 11, 99, 87
   6. 11, 87, 99

## Question 5: Grades

|  |  |
| --- | --- |
| Mark | Grade |
| 0 | N/S |
| 1-39 | F |
| 40-49 | D |
| 50-59 | C |
| 60-69 | B |
| 70-79 | A |
| 80-100 | A\* |

Write a program that assigns grades to students based on their marks:

1. Create a program that asks the user for a mark from 0 to 100 and prints a corresponding grade using the table to the left.
2. Devise a test case scheme and test that your program produces the correct output!
3. This might be trickier than you think. What if the user enters a negative number? What if they enter a number greater than 100? What if the thing they enter isn’t a number at all?

## Question 6: Who’s that Pokémon?

Write a program that asks the user for the type of a Pokémon and the first letter of its name, then makes suggestions about what that Pokémon might be called.

|  |  |  |
| --- | --- | --- |
| Type | First letter | Suggestion |
| Water | S | Squirtle |
| Water | T | Tentacool |
| Fire | C | Charmander |
| Fire | M | Moltres |
| Grass | B | Bulbasaur |
| Grass | O | Oddish |
| Electric | P | Pikachu |
| Electric | V | Voltorb |

1. Ask the user for the type of the Pokémon.   
   This can be one of:
   1. Fire
   2. Water
   3. Grass
   4. Electric
2. Ask the user for the first letter of that Pokémon’s name.
3. Based on the information you’ve got, output suggestions based on the table to the right.

## Question 7: Comparing Many Numbers

Improve your answer to question 1 by allowing multiple comparisons per program run.

1. Copy your answer to question 1 and save it in a different file.
2. Adjust your answer to use a loop. Keep asking for pairs of numbers and comparing them until the user types “quit” instead of a number.
3. Pay particular attention to your casts here. Can you wrap the casts around your input() calls and still have the program work as expected? If not, where should you put them instead?

## Question 8: Sum of Numbers at Even Indices

Write a program that sums up only the numbers in a list that are at even indices.

1. Start a fresh file.
2. Declare a list of integers containing the values 14, 5, 19, 20, 21, 66 and 89 in that order.
3. Using a loop, add up only the values at even indices in this list. So that’s:
   1. 14 (index 0)
   2. 19 (index 2)
   3. 21 (index 4)
   4. 89 (index 6)
4. Do not forget to make use of the modulo operator for determining number evenness. Use a *while* loop, consult this week’s lecture for a hint.

## Extension Activity: Bank Consultancy

Write a program to make decisions on loan applications.

1. Your program must ask the user for 4 values:
   1. Credit score (0 to 10 inclusive)
   2. Address term (months at present address)
   3. Income (£s)
   4. Request (£s) (loan amount requested)
2. Your program must then process this data, then print out the input data and a message stating whether the loan request has been granted or not.
3. In conversation with the bank manager, you were given the rules below.
4. You will need to think about organising these and devising an algorithm before you begin coding.

### The Rules

If income or credit score is zero, or time at present address is less than 12 months, the request is refused.

If the requested loan amount is greater than the applicant’s income, the loan is only granted if the request is less than twice the income and the applicant has lived at their present address for 60 months or more and the credit score is 5 or greater.

If the credit score is 7 – 10 inclusive, and time at present address is greater than 12 months but less than 60, and the request is less than the applicant’s income, grant the loan.

If the credit score is between 2 and 5 inclusive and the applicant has been at their present address for 60 months or more and the requested loan is less than the applicant’s income, grant the loan.

If the credit score is 1 and the applicant has been at their present address more than 12 months and the requested loan is less than 20% of the applicant’s income, grant the loan.

In all other cases, the application is refused.

## Extension Activity: Bubble Sort

Improve your answer to Question 4: Sorting, by asking a user to input a sequence of integers into a list and implement a Bubble Sort algorithm <https://en.wikipedia.org/wiki/Bubble_sort>

Resist the urge to Google “Python Bubble Sort” and instead use your problem solving skills.

## Document History

Revision 0 (22-Sep-20): This is the initial version of the 2020/21 exercise.