Temasek Junior College
2022 JC1 H2 Computing
Practical 6 – Selection Control Structures in Python I

Session Objectives
By the end of this session, you will learn:
(i) what the comparison operators are.
(ii) what the logical (Boolean) operators are.
(iii) the use of the `if`, `if-else` and `if-elif-else` in Python programming.

§6.1 Comparison Operators

Comparison operators are used to compare the statement written on the left of the operator to that written on the right of the operator. This results in a Boolean result of either `True` or `False`.

[illegible]

In Python, there are 6 comparison operators:

• Equal to	`==`
• Not equal to	`!=`
• Greater than	`>`
• Greater than or equal to	`>=`
• Lesser than	`<`
• Lesser than or equal to	`<=`

Exercise 1
Enter and execute the following comparison statements on the REPL. Observe what happens and fill in the results.

(a) `2 == 3`	
(b) `2 != 3`	
(c) `2 > 3`	
(d) `2.0 >= 3.0`	
(e) `2.0 < 3.0`	
(f) `2.0 <= 3.0`	

Besides the number data type, the comparison operators may also be used to compare strings. Python compares string lexicographically i.e using ASCII values of the characters. We will learn more about ASCII code in the course of our study.
You may find the ASCII value of a character using the `ord()` in Python. For example `ord('a')` returns 97 since the ASCII value of 'a' is 97.

Exercise 2

Enter and execute the following comparison statements on the REPL. Observe what happens and fill in the results.

(a)	`'A' == 'a'`	
(b)	`'A' < 'a'`	
(c)	`'123' == '321'`	
(d)	`'Hello, World' < 'Github'`	

§6.2 Logical (Boolean) Operators

Python has 3 logical (Boolean) operators.

- `and`
- `or`
- `not`

All 3 operators return Boolean results of `True` or `False` when used.

§6.2.1 Boolean operator - `and`

The `and` operator returns `True` when statements on the left and right of the operator are all true. If either or both of the statements are false, then it will return `False`.

Exercise 3

Enter and execute the following code into the REPL. Observe what happens and fill in the results.

(a)	`True and True`	
(b)	`False and True`	
(c)	`5 >= 5 and 6 > 3`	
(d)	`'123' == "123" and 5 < 5`	
(e)	`2 > 3 and 3 > 4`	

§6.2.2 Boolean operator - `or`

The `or` operator returns `True` when the statement on the left of the operator or that on the right or both are true (i.e. at least one of the statements is true).

Exercise 4

Enter and execute the following code into the REPL. Observe what happens and fill in the results.

(a)	`False or True`	
(b)	`True or False`	
(c)	`5 >= 5 or 6 > 3`	
(d)	`'123' == "123" or 5 < 5`	
(e)	`2 > 3 or 3 > 4`	

§6.2.3 Boolean operator - `not`

The `not` operator reverses the outcome of a statement.
The `not` operator should be evaluated first, followed by the `and` operator and lastly the `or` operator

Exercise 5
Enter and execute the following code into the REPL. Observe what happens and fill in the results.

(a) `not 6 > 3`	
(b) `not 6 > 3 and 7 < 8`	
(c) `not 6 > 3 or 7 < 8`	
(d) `6 > 3 and not 7 < 8`	
(e) `6 > 3 or not 7 < 8`	

§6.2.4 Operator Precedence

You may refer to here for a full list of operator precedence:
https://docs.python.org/3/reference/expressions.html#operator-precedence

Exercise 6
Enter and execute the following code into the REPL. Observe what happens and fill in the results.

(a) `not True and False`	
(b) `not (True and False)`	
(c) `True or not True and False`	

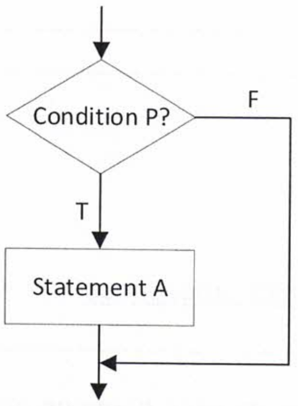
Note that for (c), it is the same as the statement as `True or (not True and False)`, since `and` should be evaluated before `or`.

Forming statements that apply the correct order of logical operators may be challenging when multiple logical operators are required. To circumvent this problem, brackets may be used to determine the order of operations. This follows Mathematical principles when statements in brackets are evaluated first. Statements enclosed in the innermost pair of brackets will be evaluated first when there are nested pairs of brackets.

By now, you should have some theoretical knowledge of how the selection control structure works ***(Recall: Problem Solving and Algorithm Design 2)***. In Python, selection control structures can be implemented using the **`if` conditional**, **`if-else` conditional** and **`if-elif-else` conditional**.

§6.3 The `if` Conditional

The `if` conditional makes use of the following selection control structure and has the following syntax structure (note the use of colon after the `<condition>` and the 4 spaces indentation of the `<action>`.



```
if <condition>:
    <action>
```

Exercise 7

The following code demonstrates the use of the `if` conditional. Type the code in your Python file editor and save it as **P6Ex7.py**.

```
# Example of if conditional
a = 3
b = 2
if a > b:  # condition
    print("a is larger than b")  # action if condition is satisfied
```

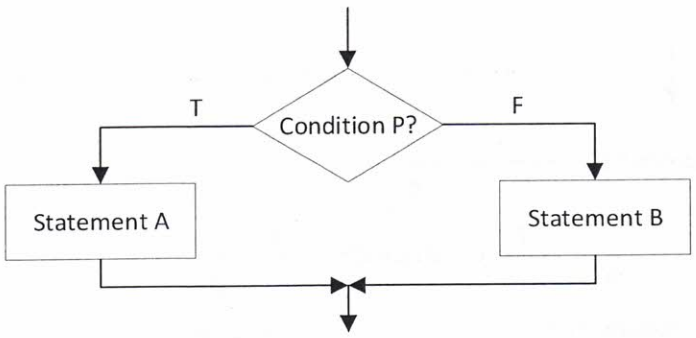
Exercise 8

Write a program to print the statement "`The length is valid`" if the user's input has exactly 8 characters.
(Similar to Example 6 of *Problem Solving and Algorithm Design* 2)
Name the file as **P6Ex8.py** and save it in your computer.

Sample Input	Sample Output
`88888888`	`The length is valid`
`7777777`	

§6.4 The `if-else` Conditional

The `if-else` conditional makes use of the following selection control structure and has the following syntax structure (note the use of colon after the `if <condition>` and the `else`; note also the 4 spaces indentation of the `<action>` and `<alternative action>`).



```
if <condition>:
    <action>
else:
     <alternative action>
```

Exercise 9

The following code demonstrates the use of the `if-else` conditional.
Type the code in your Python file editor and save it as **P6Ex9.py**.

```
# Example of if-else conditional
a = 3
b = 2
if a > b:
  print("a is larger than b")
else:  # alternative condition of a <= b
    print("a is smaller than or equal to b")
```

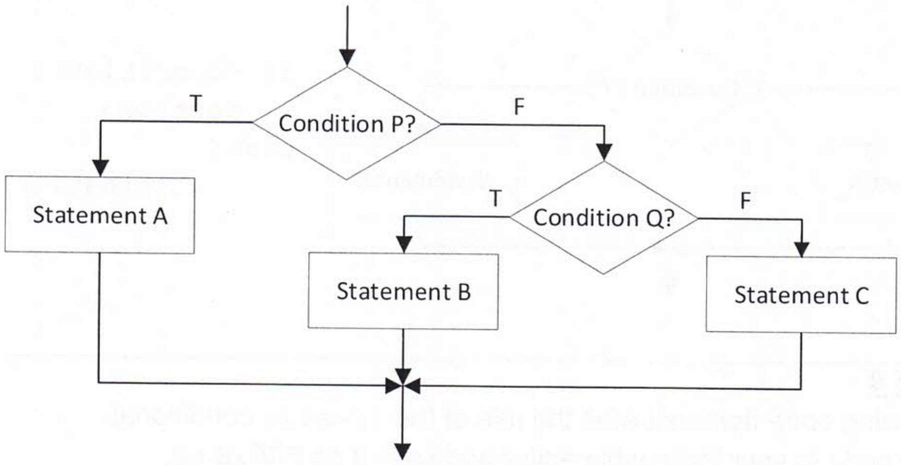
Exercise 10

Write a program to determine whether an integer keyed in by the user is even or odd.
Name the file as **P6Ex10.py** and save it in your computer.
Hint: Use the modulo operator `%` to get the remainder of the number when divided by 2.

Sample Input	Sample Output
98	98 is even
89	89 is odd

§6.5 The `if-elif-else` Conditional

The `if-elif-else` conditional makes use of the following selection control structure and has the following syntax structure (note the use of colon after the `if <condition>`, the `elif <condition>` and the `else`; note also the 4 spaces indentation of the `<action>` and `<alternative action>`).



```
if <condition 1>:
   <action 1>
elif <condition 2>:
    <action 2>
else:
    <alternative action>
```

Exercise 11

The following code demonstrates the use of the `if-elif-else` conditional.
Type the code in your Python file editor and save it as **P6Ex11.py**.

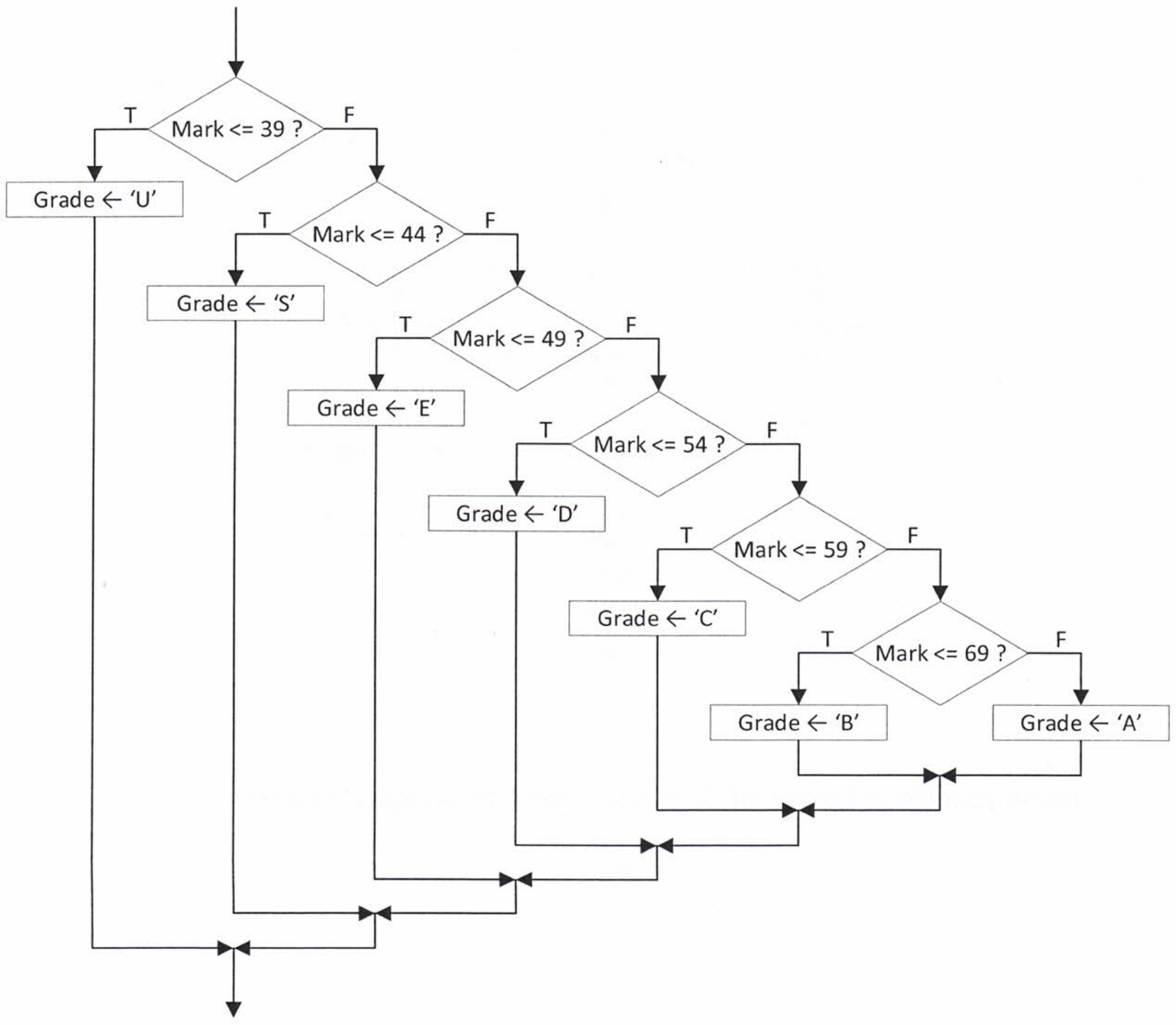
```
# Example of if-elif-else conditional
a = int(input("Please enter a: "))
b = int(input("Please enter b: "))

if a > b:  # condition 1
    print("a is larger than b")
elif a < b:  # condition 2
    # alternative action based on satisfying condition 2
    print("a is smaller than b")
else:
    # alternative action if condition 1 and 2 are both not satisfied
    print("a is equal to b")
```

The `elif` statement is Python's version of an `else if` statement and forms part of a nested selection control structure.

In fact, multiple nesting of conditions can be done with more than one `elif` statement using the following syntax structure:

```
if <condition 1>:
   <action 1>
elif <condition 2>:
   <action 2>
elif <condition 3>:
   <action 3>
          :
          :
else:
    <alternative action>
```

An example of the nested selection control structure is the assignment of A Level grades which you have seen previously *(Example 7 of Problem Solving and Algorithm Design 2)*.



Exercise 12

Write a program that is able to print the corresponding A Level grade based on the score provided by the user.

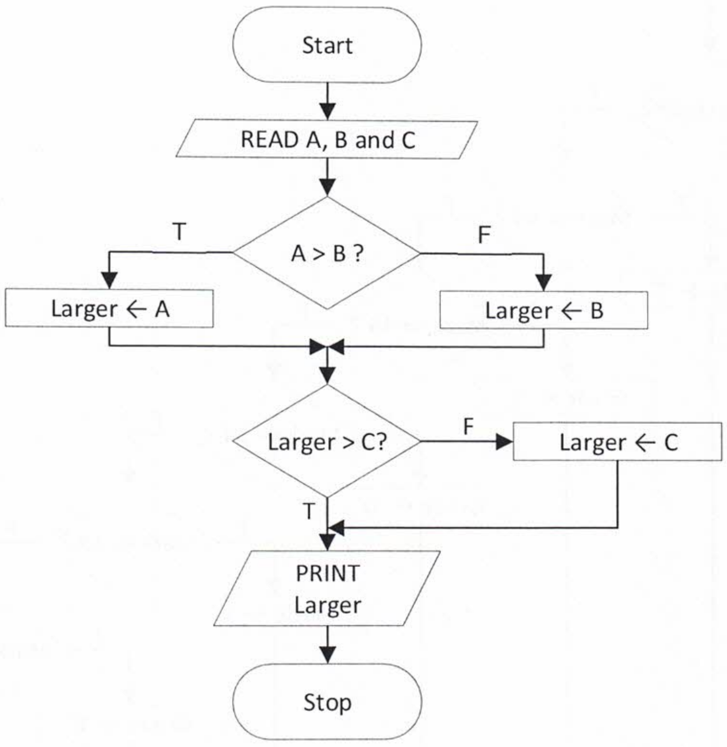
Name the file as **P6Ex12.py** and save it in your computer.

Sample Input	Sample Output
`27`	`Your grade is U`
`50`	`Your grade is D`
`59`	`Your grade is C`
`78`	`Your grade is A`

Tutorial (Submit on Google Classroom)

Please remember to use meaningful identifiers and identifier conventions adhering to PEP 8. Include the use proper commenting in your program.

1) **Which is the larger number?** (*Example 8 of Problem Solving and Algorithm Design 2*)
Write a program to implement the algorithm as seen from the flowchart below:



Name your file as **larger_of_3.py** and submit on Google Classroom.

2) **Human Year to Dog Year**

The age of a dog can be computed as follows:

- For the first two human years, a dog year is 10.5 times that of a human year.
- Thereafter, for each additional human year, a dog year is 4 times that of a human year.

Write a program that takes in the user input for the number of human years and then converts it to dog years as output.

Sample Input for Human Years	Sample Output
`2`	`Dog Years: 21.0`
`-2`	`Error`
`4.5`	`Dog Years: 31.0`

Name your file as **dog_year.py** and submit on Google Classroom.

3) **Home Loan Deposit** (*Tutorial 2.3, Q2 of Problem Solving and Algorithm Design 2*)

A home mortgage authority requires a deposit on a home loan according to the following schedule:

Loan $	**Deposit**
less than $25 000	5% of loan value
$25 000 – $49 999	$1250 + 10% of loan over $25 000
$50 000 – $100 000	$5000 + 25% of loan over $50 000

Loans in excess of $100 000 are not allowed.

Write a program that reads a loan amount, compute and print the required deposit. Name your file as **home_loan_deposit.py** and submit on Google Classroom.

Practice on Your Own

4) **Leap Years**

The algorithm to determine whether a particular year is a leap year is given as follows:

Step 1: If the year is divisible by 4, go to Step 2. Otherwise, go to Step 5.
Step 2: If the year is divisible by 100, go to Step 3. Otherwise go to Step 4.
Step 3: If the year is divisible by 400, go to Step 4. Otherwise go to Step 5.
Step 4: The year is a leap year (i.e. it has 366 days).
Step 5: The year is not a leap year (i.e. it has 365 days).

Write a program that takes in the user input for the year and then output whether the year is a leap year.

Sample Input for Year	Sample Output
`1997`	`1997 is not a leap year`
`1800`	`1800 is not a leap year`
`2020`	`2020 is a leap year`