COMP101

Introduction to Programming 2019-20

Assignment-02

Issue Date: **Monday 14th October 2019**

Submission Date: **Monday 21st October (12:00 noon)**

Summary:

Assignment-02 is worth 12% of the total marks for COMP101.
The assignment uses I/O control of strings and numbers, sequencing, selection and module/function handling

You must submit an attempt else a fail grade for the module will be awarded.

Guidance:

Assessment is based on design, clarity, accuracy, code use, testing and documentation.

Deadline and submission details:

Two files need to be submitted: One .py file and one .pdf file

a) Submit one .py file of your coded solution from IDLE with filename in format of:

familyName_givenName-CA02.py e.g. Smith_John-CA02.py

Within the code, the first 3 lines should be comment lines as follows:
#Your University id and filename
#Month and Year of coding
#Brief description of the problem solved

b) Submit one .pdf file containing your test table (use the test table template)

Your documents are to be submitted electronically via the department submission server at https://sam.csc.liv.ac.uk/COMP/Submissions.pl

Earlier submission is possible, but any submission after the deadline attracts the standard lateness penalties - see http://www.csc.liv.ac.uk/department/regulations/practical.html

Plagiarism and collusion guidelines will apply throughout the assignment submission

COMP101 2019-20 Assignment-02

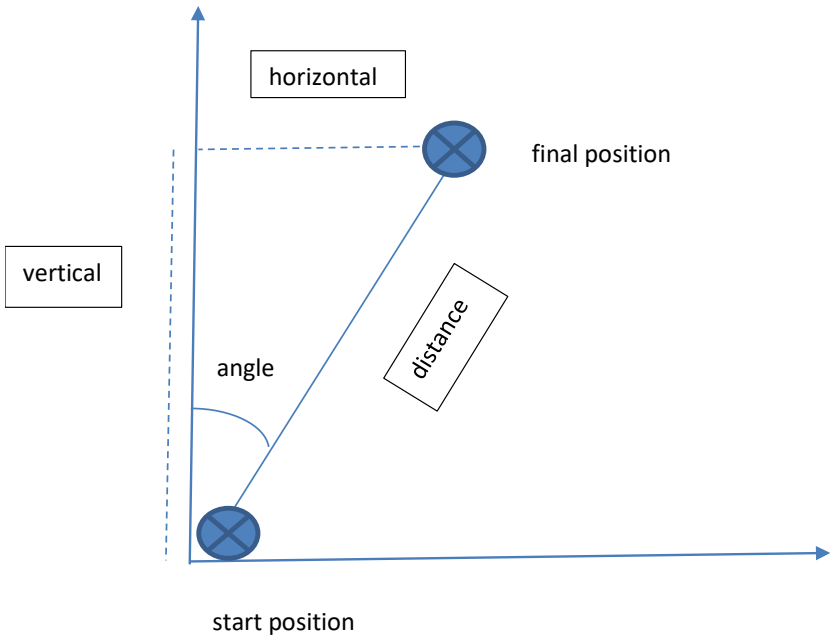
Assessment Information

Assignment Number	02 (of 07)
Weighting	12%
Assignment Circulated	See front page
Deadline	See front page
Submission Mode	e-submission
Learning outcome assessed	LO1: Identify principles and practice of using high level programming constructs to solve a problem LO3:Produce documentation in support of a programmed solution LO4: Use a suitable Integrated Development Environment to carry out implementation, interpretation/compilation, testing and execution LO7: Develop debugging skills to correct a program
Purpose of assessment Marking criteria	Using sequence and selection constructs to control I/O of strings and numbers in successful calculations for a given problem, using modules as required. Total marks over seven questions as a percentage
Submission necessary in order to satisfy module requirements?	Yes Assignments are not marked anonymously
Late Submission Penalty	Standard UoL Policy.

Problem Specification:
A robot rover on our moon/Mars etc moves on a flat horizontal plane at an angle between 0 to 90 degrees (inclusive) from North (the vertical).
It travels at a fixed speed of 1.5m/sec for a period of time.

We need to know how far the robot has moved overall, its horizontal distance from its start point, its vertical distance from its start point, the amount of battery used and the amount of battery remaining.

e.g.



The robot uses battery power to move. Battery usage is 2.7% per second. Assume the battery is fully charged when the robot is told to move from its original start position

The estimated battery usage spent when the robot moves is calculated as time spent moving multiplied by battery usage.

Input:
i) an angle in degrees between 0 and 90 inclusive
ii) a time of travel in seconds as a positive number

No validation is required and there is no need to test for values outside of the ranges specified and as designated in the specification

---

Calculate:

(i) The overall distance travelled by the robot
Where distance = speed * time

(ii) The horizontal distance from the starting point
Where horizontal distance = distance * (sin angle)

(iii) The vertical distance from the starting point
Where vertical distance = distance * (cos angle)

(iv) The estimated battery usage to reach its final position

(v) Under the robot's current technical specification (battery = 2.7% usage, speed = 1.5m/s etc) this obviously implies there is a limited range for the robot to travel before the battery expires.
After user input, if the battery is likely to expire before the robot reaches its final position, use a selection statement to inform the user the robot cannot travel the distance required as the battery will expire before it gets there. The program can stop at this point under this test case (simply because we haven't yet covered iteration to allow the user to re-enter)

Whenever the robot does reach its final position, consider:
Should battery usage exceed 50% but has not reached 100% inform the user the robot may not have enough battery to perform a return trip along the same path it just came from.

Should battery usage be below 50%, tell the user how much battery life remains for the return trip.
The user is not expected to take any action on this information and there is no need to program the robot to make the return trip. It can stay where it is. You can assume solar panels will re-charge the battery eventually, but don't consider this here

Horizontal and vertical distances use trigonometric ratios of sine (sin) and cosine (cos). Input by the user is in degrees. Python calculates in radians.

Output:
e.g.:
A robot moving at an angle of 30 degrees for 10 seconds:

Distance travelled =
Horizontal movement =
Vertical movement =
Estimated battery usage =
Message to user about battery usage

Mark scheme
Analysis and Design 20%

Implementation 60%

Testing 20%

The mark scheme looks for:
Sequence and selection constructs with module handling:
Iteration, def constructs etc are not part of the mark scheme, so must be ignored

Efficient use of variables to handle the input data and the use of variables to make the code clear and readable, thus aiding maintenance and debugging

Appropriate use of output techniques that are of benefit to the user in providing usable feedback concerning the status of the robot

Use in-line comments sparingly, but effectively

Appropriate testing to determine accuracy and/or problems (which may be documented in the test table comments column)

There is no need to consider obstacles, mazes, transmission lag-time between celestial bodies etc

---