2807/7001ICT Programming Principles (I), Trimester 3, 2019 Workshop 9

School of Information and Communication Technology
Griffith University

November 6, 2019

Module	3
When	Day 9
Goals	This workshop's exercise has a longer problems using a `dict`.
Marks	4
Due	Pre-workshop questions at the start of the workshop; problems by the *beginning* of the next workshop.

1 Preparation

Before your workshop class:

- Read all of this document.
- Review the lecture notes sections 1 to 24.
- Bring some paper (a print-out of this document is best) and writing implements.
- Bring a storage device, such as a portable hard drive and cable, or a USB drive.

2 Pre-workshop questions (1 mark)

Complete these questions in writing *before* the start of the workshop. They will be marked early in the workshop.

1. Give two differences between a set and a dict.

2. Give an important difference the keys and the values in a `dict`.

3. Complete this table, if the following statements have already been executed. Try to work it out yourself, before confirming with the REPL.

```
i = 3
s = {1, 5, 2, 10}
t = {2, 12, 15}
d = {'a': 111, 'b': 222, 'd': 333}
```

expression	*type*	*value*
`i`	`int`	`3`
`s`		
`len(s)`		
`s | t`		
`s & t`		
`s - t`		
`t - s`		
`t <= s`		
`2 in s`		
`i not in s`		
`d`		
`len(d)`		
`d['b']`		
`d.get('b')`		
`d.get('c', 0)`		
`sorted(d.keys())`		

3 Workshop activities

3.1 Marking previous workshops' problems

If you have problems that still need marking from the previous workshops 7 and 8, get them marked at the *start* of this one.

3.2 Problem 1 (2 marks)

Problem: The people of ancient land of Pacific Baza had a simple mathematical system that knew only natural numbers and addition. The genius Bazan scholar, Gringo el Possum, built a computer from wood and various animal parts. Archeologists have recovered ancient scrolls with enough scraps of programs to reconstruct the programming language he named, *Adder*.

The Adder language has only a few simple statements:

`quit`	Exit the REPL or terminate a program.
`input` *`var`*	Prompt for and allow the user to enter a value for the variable named *`var`*.
`print` *`val`*	Print the value *`val`*.
`var` `gets` *`val`*	variable *`var`* is assigned the value *`val`*.
`var` `adds` *`val`*	variable *`var`* has the value *`val`* `added to it`.

Where:

- *var* is always a variable name that contains only letters; and
- *`val`* can be either:
  - a variable name that contains only letters; or
  - a natural number that contains only digits.

The Adder REPL allowed the user to enter commands interactively. For example:

```
Welcome to the Adder REPL.
??? a gets 1
??? input b
Enter a value for b: 2
??? c gets a
??? c adds b
??? print a
a equals 1
??? print b
b equals 2
??? print c
c equals 3
??? print z
z is undefined.
??? print 32
32
??? blerg
Syntax error.
??? 23 gets z
Syntax error.
??? quit
Goodbye.
```

Write the Adder REPL.

Hints: Make good use of string methods. Can you divide your program up into smaller pieces by defining functions?

3.3 Problem 2 (1 mark)

Problem: Write an Adder interpreter, that prompts for and executes an Adder script. For example if the file `Children.ad` contains:

```
input sons
input daughters
children gets sons
children adds daughters
print children
quit
```

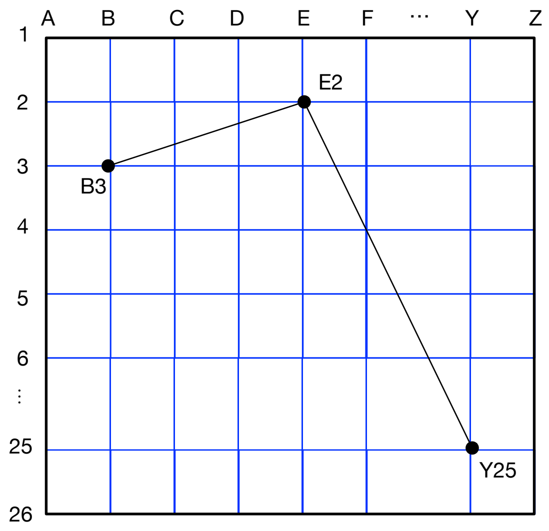
The interpreter would run like this

```
Script name: children.ad
Enter a value for sons: 3
Enter a value for daughters: 4
children equals 7
```

Hint: This should involve a few small modifications to your REPL.

3.4 Problem 3 (1 mark)

Problem: A road map defines locations as map references like `B3`, where `B` is the x-coordinate value and `3` is the y-coordinate.



The grid lines are 0.5 km apart.

Write a program that allows the user to enter a trip as a sequence of any number of map references on one line, and reports the total length of the trip, assuming they can travel in straight lines. For example:

```
Enter trip map references: C2 B5 Y25
Total distance = 16.8 km
```

For badly formatted map references, your program should exit, reporting the first bad map reference.

```
Enter trip map references: E6 E4 D7 d43 F5
Bad reference:  d43
```

Hints: you need to *split* the input line into separate references; each reference starts with one character which must be an upper case letter, and the rest must be only digits; and Pythagoras will help. The function `exit()` can abort the program if you detect an error in the input.

4 After the workshop

- You have created programs that might be useful to refer back to in future workshops. Make sure that you will have that work in the future. One copy is not enough for at IT professional. You should have at least 2 copies:
    1. on your Griffith network storage drive; and
    2. on your portable storage device.