Faculty of Science, Engineering and Technology

Introduction to Programming

Pass Task 2.1T: Debug

Overview

This program will allow you to demonstrate simple debugging of Ruby code.

Purpose:	Learn to correct programming errors.
Task:	Correct the provided code so it works according to the specification.
Time:	This task should be started in your second lab.
Resources:	**Pine, C 2013 *Learn to Program*, The Pragmatic Programmer. (available in the library).** **Learning Material for Week 1 and Week 2.**

Submission Details

You must submit the following files to Canvas:

- Corrected *debug.rb* source code
- Screenshot of the Terminal showing the execution of the debug program.

Make sure that your task has the following in your submission:

- The program must read the required details from the user, and respond correctly
- Code must follow the Ruby coding convention used in the unit (layout, and use of case).
- The code must run and the screenshot show it working.
- Your program must have a procedure for Main.
- Your program must have the indicated local variables, and use them appropriately.

Instructions

To explore this topic, we will correct a Terminal program that:

- asks the user to enter their name and their age.
- calculates and outputs the year the user was born

Note: these are tasks you have already done by now in other programs.

Look at the *debug.rb* program shown below and available in the Resources for the task:

```
# Fix up the following code that it works and produces the
# expected output in the task specification.

# Asks the user to enter their age and returns an integer age
def get_age()
  puts "Enter your age in years: "
  age = gets
  return age_in_years
end

# takes a prompt and displays it to the user then returns the
# entered string
def get_string()
  puts prompt
  s = gets
  return s
end

# Calculate the year born based on the parameter age and print
# that out along with the user's name.
def print_year_born(age)
  year_born = Date.today.year -  age
  puts "You were born in: " + year_born
end

def main()
  age = get_age()
  name = get_string()
  print_year_born(age)
end

main()
```

An example of the expected operation of the debug program is as follows:

```
Enter your age in years:
11
Enter your name:
Jasper
Jasper you were born in: 2008
```

1. You may need to add the following code elements to correct the program:

```
1. .to_s - converts a type to a string.
2. .to_i - converts an integer in string form to numeric form.
3. require 'date' - includes the date library.
4. .chomp - removes whitespace from a string (eg: newlines,
   tabs, spaces).
```

Example of how to fix the program:

We are looking for the **where** and the **what.**

The first error message you encounter looks as follows:

```
MacBook-Pro-6:Resources mmitchell$ ruby debug.rb
Enter your age in years:
11
debug.rb:10:in `get_age': undefined local variable or method `age_in_years' for main:Object (NameError)
        from debug.rb:29:in `main'
        from debug.rb:34:in `<main>'
MacBook-Pro-6:Resources mmitchell$
```

The part that says: `debug.rb:10:in `get_age':` indicates both the line and the function **where** the error occurred, in this case line 10 which is in the function get_age().

The part that says: `undefined local variable or method `age_in_years'` indicates the **what** – in this case the code refers to something called *age_in_years* which we have not defined as either a variable or a function – so Ruby does not know what it is.

So how do we fix this problem?

```
# Asks the user to enter their age and returns an integer age
def get_age()
  puts "Enter your age in years: "
  age = gets
  return age_in_years
end
```

This variable *age* should be called *age_in_years*

Without the change in name (see box above) this variable has not been declared.

So now you need to continue with the other errors that arise.

Now that the Task is complete you can submit it for assessment, which will help prepare it for your portfolio.

1. Use **Skitch** (or your preferred screenshot program) to take a screenshot of the Terminal, as this is one of the things you will need to submit.
2. Save the document and backup your work to multiple locations!
   - Once you get things working you **do not** want to lose them.
   - Work on your computer's storage device most of the time... but backup your work when you finish each task.
   - Use **Dropbox** or a similar online storage provider, as well as other locations.
   - Canvas is not a Backup of your work, so make sure you keep a copy!
   - A USB key and portable hard drives are good secondary backups... but can be lost/ damaged (do not rely upon them).
3. Login to Canvas, and locate Tutorial Task 2.1
4. Change the status of the task to **Ready To Mark**
5. Upload your completed code and the screenshot.
6. If you check back later Canvas will have prepared these as PDFs for your tutor to assess.

You now have another of your first portfolio pieces. This will help demonstrate your learning from the unit.

Note: This is one of the tasks you need to **submit to Canvas**. Tutors should give guidance and perhaps feedback in the tutorial class.
Check the assessment criteria for the important aspect your tutor MAY check when assessing your finished portfolio.