Portfolio Tasks

About the portfolio tasks

Note: there are two (2) tests which ALL students must complete. Watch for announcements about those, they are not in this section.

NB: Some tasks are automatically marked, others are manually marked. The manually marked ones are indicated by . Some of these tasks your tutor will ask you to demonstrate in class.

NOTE: the checkpoints for feedback - you need to submit your tasks before these if you want feedback.

Two deadlines are hard deadlines. By Week 7 - you must have a draft in for the week 1 - 5 tasks and have met your tutor to discuss these by the end of Week 7. Also you must have a draft for 9.1.1 which you have shown to your tutor before the end of Week 11. Failure to meet these 2 deadlines will result in a maximum grade of 50 P.

If you need an extension on any task, you need to apply for Special Consideration within 3 days of the deadline. See the process here: https://www.swinburne.edu.au/media/swinburneeduau/documents/Foundations-and-Pathways-Special-Consideration-.pdf

This is what you need to have marked Complete to be awarded different grades:

- Pass Level
  - Everyone must complete **ALL** these tasks at least to the 50 P level (**especially the 2 tests and 9.1.1**).

- Credit Level
  - To get a Credit (65) grade **ALL** Pass tasks must be completed and **ALL** Credit tasks completed (including the 2 tests).
  - The higher grades within the Credit level are determined as follows:
    - If you complete **only** 9.1.2 then you may be eligible for a 60 C
    - If you complete 9.1.2 and all other credit tasks then you are eligible for 65
    - If you complete some of the D tasks then you **may be** eligible for a 67 C. This will depend on the assessment panel.

- Distinction Level
    - To get a Distinction of **over 70 D ALL** Pass tasks must be completed and **ALL** Credit tasks completed and **ALL** Distinction tasks (including the 2 tests). **(Proviso: you can get a 70 D if you do all the pass and credit tasks, and all distinction tasks other than the custom program)**
    - The distinction tasks include a Custom project . This needs to be discussed and approved by your tutor based on the design you prepared in Task 7.4 The requirements for the Custom Project are attached to the task description.

- High Distinction Band 1
    - To get a High Distinction from 80 - 90 **ALL** Pass tasks must be completed and **ALL** Credit tasks completed and **all** Distinction tasks**,** as well as: **9.4 HD, 11.4 HD and a Custom Code Program.**
    - You must attend an interview to present your custom program. This includes a requirement to prepare a 3 minute video explaining your program and your design. The requirements for the Custom Program are attached to the task description.

- High Distinction Band 2
    - To get a High Distinction from 90 - 100 **ALL** Pass tasks must be completed to a 53 P level, and **ALL** Credit tasks completed **all** Distinction tasks completed as well as a **Custom Code Program** and a **Custom Project.**
    - You must attend an interview to present your custom program and your custom project. This includes a requirement to prepare a 3 minute video explaining your program and your design and a 2 minute video explaining your Custom Project. The requirements for the Custom Code Program and Custom Project are attached to the relevant task descriptions.

Assessment Criteria

Fail
Does not meet Pass standard.

Portfolio not submitted,

Tests not signed off as Complete, and/or

Fails to demonstrate coverage of all unit learning outcomes to the required standards

Pass
Tests are complete, and all Pass Tasks are Complete.

Credit
All Pass and Credit Tasks are Complete.

Distinction
All Pass and Credit Tasks are Complete, and all Distinction Tasks are Complete. Attended an interview if Custom Program created.

High Distinction
Distinction met, and either Custom Program meets HD requirements, or adequate HD Report.

High Distinction criteria met with both a Custom Project at HD (85 minimum) standard and High Distinction project.

50 P Tests passed, major task passed, all pass tasks.		**55 P (think D)** Complete all Pass Tasks, Tests, major task, and all tutorial tasks.		Learning Summary Report + Tests +Pass Tasks + Tutorial/Lab Tasks
60 C All Tasks are complete to 55P, all Credit tasks, except major task.		**65 C (think C)** All Tasks to 55P, all credit tasks and major credit task.		+ Credit Tasks
70 D All Credit Tasks are Complete, and all D Tasks except for Custom Program.	**73 D (think B-)** Meets Distinction, but some issues with code, reflections, or learning summary.	**75 D (think B)** Meets Distinction, with generally good code, reflections, and learning summary.	**77 D (think B+)** Meets Distinction, with good code, reflections, and learning summary.	+ Distinction Tasks + Custom Program +15 min Interview
	83 HD Excellent outcomes, but some weaknesses.	**85 HD (think A)** Excellent outcomes, good reflections, code quality and learning summary.	**87 HD** All outcomes are excellent, with very high quality finish.	+ HD standard on custom program,
	93 HD All outcomes are excellent, some small issues with research report or analysis.	**95 HD (think A+)** All outcomes are excellent, good research report and analysis.	**97 HD** All outcomes are excellent, including research method, report and analysis.	**100 HD (think A++)** Something special! + HD standard on custom program (85 or above), and HD Project

1. Categorise based on grade first! Use self assessment, with sanity check. Should be obvious, based on work included.

2. Assume middle grade initially, then work up/down based on evidence.

3. Review 97's with unit panel to check for 100s

Introduction to Programming - Tutorial and Task Timeline

Commencement weeks for tasks						Task Descriptions						
Week/Topic	Tut. Tasks	Pass Tasks	Credit Tasks	D grade	HD (Band 1)	Tutorial Focus	Tutorial Tasks	Pass Tasks	Credit Tasks	Distinction Tasks	HD (Band 1)	HD (Band 2)
1	1.1T	1.4P				Sequence and basic data types	Hello world	Hello User (sequence)				
	1.2T						Desk Check - Bill Total					
	1.3T						Run Bill Total					
2	2.1T					Functions, parameters, return values. Structure charts.	Write your own functions					
	3.1T					Composite Data Types	Read Album from Terminal					
3	3.2T						Read Lines from Terminal					
4	4.1T					File Handling, Static and Dynamic Typing	Read lines from File					
	4.2T						Read Track from File					
5	5.1T	5.2P				Selection and iteration	Simple Menu Program	Static Type Checking				
6	6.1T	6.2P	6.3C	6.4D		Introduction to Arrays/ GUI Drawing	Hsnd Execution - Arrays I	Read Multiple Tacks	GOSU: Drawing Shapes	GOSU: Maze Creation		
7	Check point week - deadline for submissions for Tasks Week 1 -5						See your tutor to mark complete Tasks Week 1-5					
	7.1T	7.3P	7.4C			Arrays, Searching, Stack and Heap	Concept Map	Array Search	Custom Program Design			
	7.2T						Hand Execution - Arrays II					
	Test 1 during Week 8 - evening online					GUI and Game Programming	**Test1 (online)**					
8	8.1T	8.1.1P	8.2C	8.3D			Read Album with Tracks	Printing Genre Names	Gosu Major Cycle	Hover Button		
9	***9.1.1P***		***9.1.2C***	***9.1.3D***	9.4HD	Testing/Debugging Strategies		***Text Music Player***	***Text Player with Add***	***GOSU Music Player***	Maze Search	
			9.2	9.3					Recursive factorial	Gosu Shape moving		
10	Check point week - deadline for submissions for Tasks Week 6 - 8						See your tutor to mark complete Tasks Weeks 6 -8					
	Test 2 during Week 10 LAB						**Test2 (in labs classes)**					
	Test 2 in Lab	10.1P				Sorting, Complexity and Recursion		Fix It - Sort Records				
		10.2P						Hello World in C				
		10.3P						Hello World in Python				
11	Check point week - deadline for submissions for Tasks Weeks 9 - 10					Testing and Debugging	See your tutor to mark complete Tasks Weeks 9 - 10 (except custom code and project tasks)					
	11.1T	11.2P		11.3D	11.4HD		Learning Summary Draft	Python Program		PythonShapeMoving	C Program	
12	Task Sign Offs		12.1C			Programming in Other Languages	Task Sign Offs		Test Harness		Custom Code	Custom Project
	Check point week - deadline for submissions for Tasks Weeks 11 - 12						See your tutor to mark complete Tasks Weeks 11 -12 (except custom code and project tasks)				NB: Only Week 11, 12 tasks and Major tasks	
13	Custom Code/Custom Video/Custom Project (for HD band 2) Reflection and Custom Code/Project Completion						Reflection and Custom Code/Project Completion					

Requirements for Pass Grade:	All Pass Tasks Completed Satisfactorally
Requirements for Credit Grade:	All Pass Tasks and All Credit Completed Satisfactorally
Requirements for Distinction Grade:	All Pass Tasks and All Credit and All Distinction Tasks Completed Satisfactorally
Requirements for HD (Band 1) Grade:	All Pass Tasks and All Credit and All Distinction and all HD (Band 1) Tasks Completed Satisfactorally
Requirements for HD (Band 2) Grade:	All Pass Tasks and All Credit and All Distinction and all HD (Band 1) and all HD (Band 2) Tasks Completed Satisfactorally

NB: The Music Player is the Major Task at each grade level.

1.1T Hello World

Write a program in Ruby that outputs `Hello World.`

Click on 'Terminal' then type **ruby hello.rb** to run your code. When you are ready click 'Mark' to see if your program works as expected.

Video Guide.

Operators Quiz

Question 1 *Submitted Jul 30th 2024 at 12:23:33 pm*

Assigning a value to a variable is done using the `=` operator

- (●) True
- ( ) False

Question 2 *Submitted Jul 30th 2024 at 12:23:26 pm*

The expression `a + b` is **evaluated** to find the sum.

- (●) True
- ( ) False

Question 3 *Submitted Jul 30th 2024 at 12:23:20 pm*

The following statements can be executed in any order:

```
1   a = 3
2   b = 4
3   a = b + c
4   c = a
5   puts(" c has the value #{c}")
```

- ( ) True
- (●) False

Testing Programs

The following is an example of how to test a program - it shows the **test data** and the **expected result .**

We want to make sure that when we run our program the **actual output** matches the **expected result.**

The **pseudocode** is a description of what the program does in regular English (i.e no particular programming language):

The **required variables** describes where data is stored in the program, in this case the data being stored is **Integer** data (i.e whole numbers):

Program 1: Add 2 numbers *(Example of Desk Checking)*

Required Variables:

Integer: a, b, c.

Pseudocode:

Read the value of **a**

Read the value of **b**

Add **a** to **b** and assign the result to **c**

Print the value of **c** to the terminal.

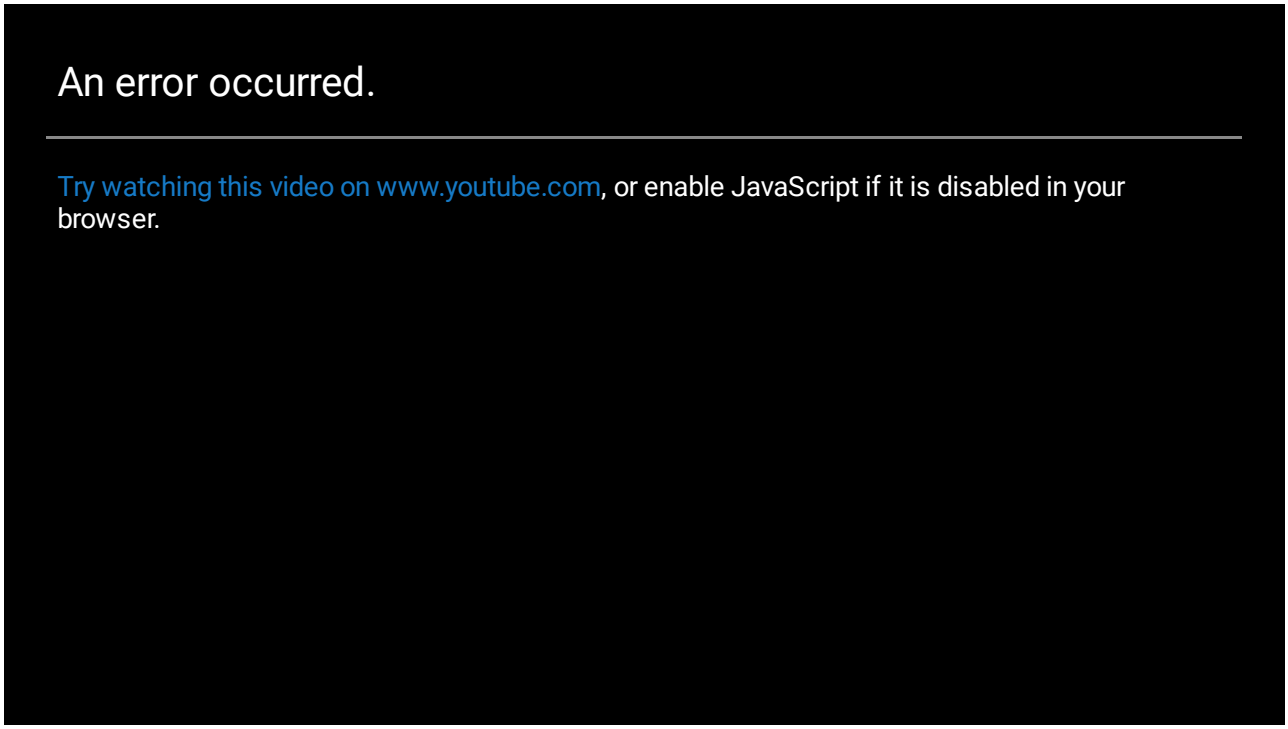
Test Data:

	First data set	*Second data set*
a	*10*	*3*
b	*5*	*4*

Expected Result:

	First data set	*Second data set*
Output:	*15*	*7*

Hand Execution



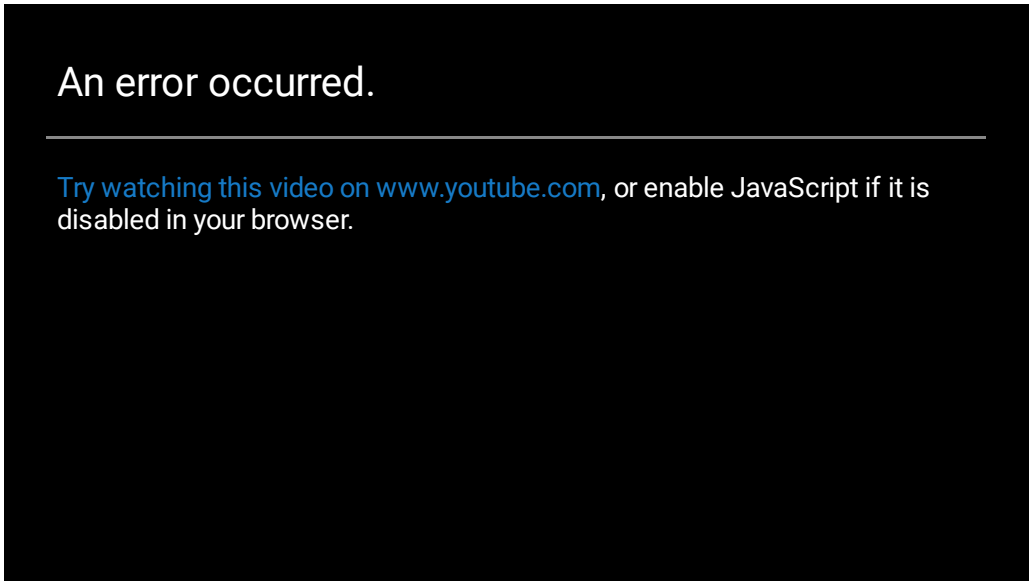
1.2T M □□ Desk Check the Bill Total program

Complete the attached document - filling in the fields based on the output of your program. Check it is working correctly, then answer the questions.

Upload your completed document (AS A PDF) to the workspace.

Tutorial Task 1.2 - Answers V2.docx

Watch the following video on how to do trace-tables using hand-execution:



Click the 'Mark' button to submit your task for manual marking by your tutor.

1.3T Run the Bill Total program

Complete and run the program Bill Total in the Terminal using the test data below.

Record the actual output of the program using this data and check it matches the expected result.

Test Data:

	First data set	*Second data set*
appetizer_price	*10.30*	*12.40*
main_price	*34.00*	*41.00*
dessert_price	*8.50*	*9.80*

Expected Result:

	First data set	*Second data set*
Output:	*$52.80*	*$63.20*

If your program is working as expected click the 'Mark' button and make sure your program passes the tests.

Video guide.

Reading and Writing Using Variables

Variables allow you to store values that can change in your program.

When you declare a **local variable** you are telling the computer to create a variable for use within the function or procedure.

You must give the variable a name and keep in mind the type of data it will store. The computer will then set aside space for you to store a value for that variable.

You can then write values to the variable and read values back.

```
# This is a variable called x we assign it the value 10
x = 10
# Print out the value of x:
puts("x is #{x}")
# read in another value for x:
puts("Enter an integer value other than 10: ")
x = gets().to_i()
# print the new value of x:
print("x is now #{x}")
```

1.4P Hello User

Change the code provided to:

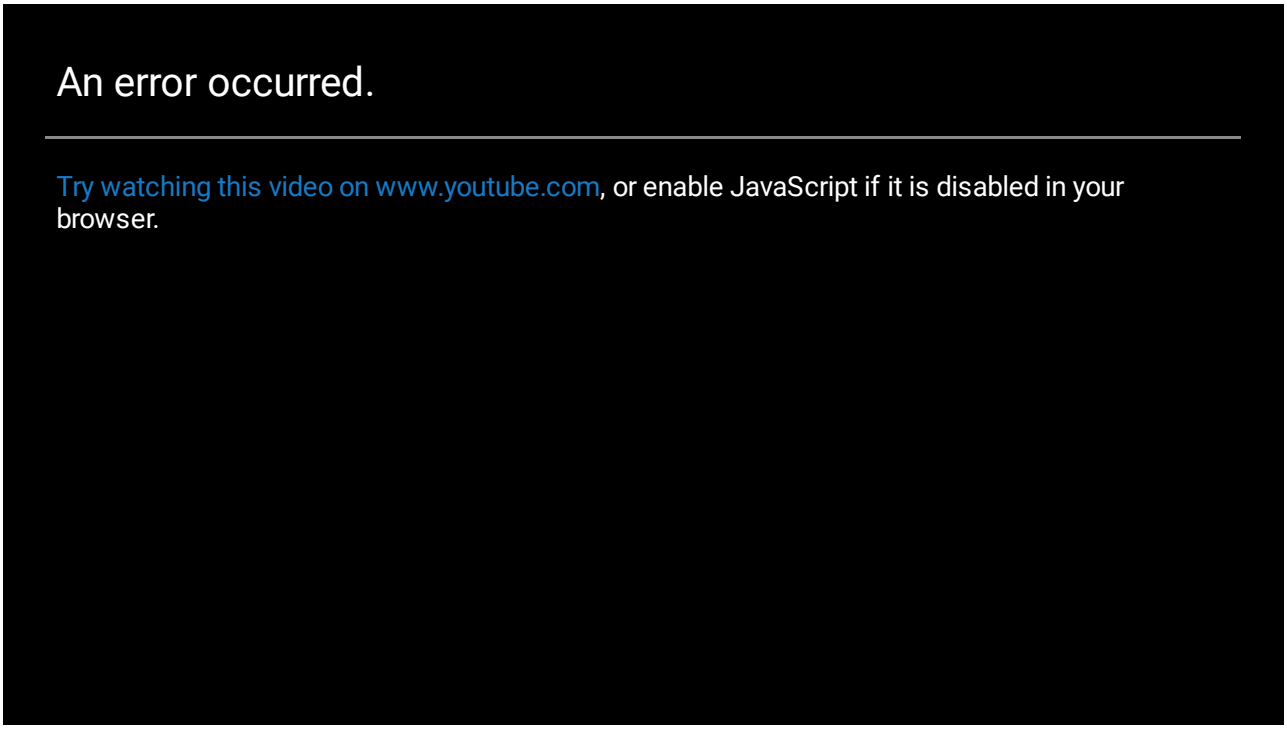
1. Read the user's name (a String, prompt with 'Please enter your name: ') and store it in the name variable.
2. Print out the user's name followed by an exclamation mark.
3. Read the user's family name, remove any white space and store it in the family_name variable.
4. Print out the user's family_name followed by an exclamation mark.
5. Read in the user's year of birth
6. Convert the year of birth to an integer then calculate the user's age and print it out.
7. Prompt for and read in the user's height in metres
8. Convert the height to inches and print out the result.

Video Guide.

Example:

```
What is your name?
Sam
Your name is Sam
!
What is your family name?
McClan
Your family name is: McClan!
What year were you born?
2012
So you are 12 years old
Enter your height in metres (i.e as a float):
1.3
Your height in inches is:
51.18113
Finished
[user@sahara ~]$
```

Naming artifacts in code (only watch to 3.35 minutes)



2.1 T M □ □ Write your own functions - Letter App

In your tutorial class design and write a program which reads in a name, address and message to be sent as a letter to the recipient.

1. Based on the description below write pseudocode for the program. Upload the pseudocode to the Ed workspace. ***NB Make sure your pseudocode is decomposed modularly.***
2. As a class group produce a structure chart for the program. Upload a photo or drawing of the structure chart to the Ed workspace.
3. Write the code and test it in Ed.
4. Annotate the code with comments explaining how to use each module (function/procedure) and what they do (not how they work).

The program should work as described below

Description of the program

Create a `main()` that calls three functions as follows:

1. Returns the label for the letter.

The label should include the title, first name, last name, and street address for the letter. It should be formatted to appear as it would on the front of a sent letter.

2. Returns the message to be send.

Your `main()` should also call a function which returns the contents of the letter. The contents should consist of simply a subject line and then after a blank line the content of the message. The message should have a line that starts: "RE:" then the subject (read from the user) then a blank line, followed by the message.

3. Prints the label and the message.

Call a procedure that takes the label and the message and prints it to the terminal.

All values except the postcode should be read as strings, the postcode should be an integer between 0 and 9999.

Example output:

```
Please enter your title: (Mr, Mrs, Ms, Miss, Dr)
Dr
Please enter your first name:
John
Please enter your last name:
Lennox
Please enter the house or unit number:
2/34
Please enter the street name:
Platinum St
Please enter the suburb:
Canberra
Please enter a postcode (0000 - 9999)
2601
Please enter your message subject line:
Philosophical Musings
Please enter your message content:
Recent discoveries using laser light have unveiled hidden properties of various gem stones. If we slice the stones very th
inly then shine cross-polarised light through them we find there are two types of jewels - anisotropic and isotropic. In p
ure light anisotropic jewels turn into all colours of the rainbow (regardless of their original colour). Isotropic jewels
(like diamonds, rubies, garnets) lose their colour and turn black. Fun fact: the 12 precious stones used to build the New
Jerusalem (Revelation 21:19) - "The wall of the city was built on foundation stones inlaid with twelve precious stones the
 first was jasper, the second sapphire, the third agate, the fourth emerald, the fifth onyx, the sixth carnelian, the seve
nth chrysolite, the eighth beryl, the ninth topaz, the tenth chrysoprase, the eleventh jacinth, the twelfth amethyst.". Al
l 12 are anisotropic. No diamonds or rubies.
Dr John Lennox
2/34 Platinum St
Canberra 2601
RE: Philosophical Musings

Recent discoveries using laser light have unveiled hidden properties of various gem stones. If we slice the stones very th
inly then shine cross-polarised light through them we find there are two types of jewels - anisotropic and isotropic. In p
ure light anisotropic jewels turn into all colours of the rainbow (regardless of their original colour). Isotropic jewels
(like diamonds, rubies, garnets) lose their colour and turn black. Fun fact: the 12 precious stones used to build the New
Jerusalem (Revelation 21:19) - "The wall of the city was built on foundation stones inlaid with twelve precious stones the
 first was jasper, the second sapphire, the third agate, the fourth emerald, the fifth onyx, the sixth carnelian, the seve
nth chrysolite, the eighth beryl, the ninth topaz, the tenth chrysoprase, the eleventh jacinth, the twelfth amethyst.". Al
l 12 are anisotropic. No diamonds or rubies.
```

NB: This task must be marked complete by your tutor - click "Mark" to check it passes the Ed tests then ask your tutor to mark it complete.

Types Quiz

Reorder the Data Types to match the Field Names below:

Question *Submitted Aug 1st 2024 at 2:33:01 pm*

1. Name
2. Family Name
3. Year Born
4. Date
5. Age
6. Height
7. Continue

NB: Some types have (1) or (2) after them - use these in that order.

String (1)

String (2)

Integer (1)

Date

Integer (2)

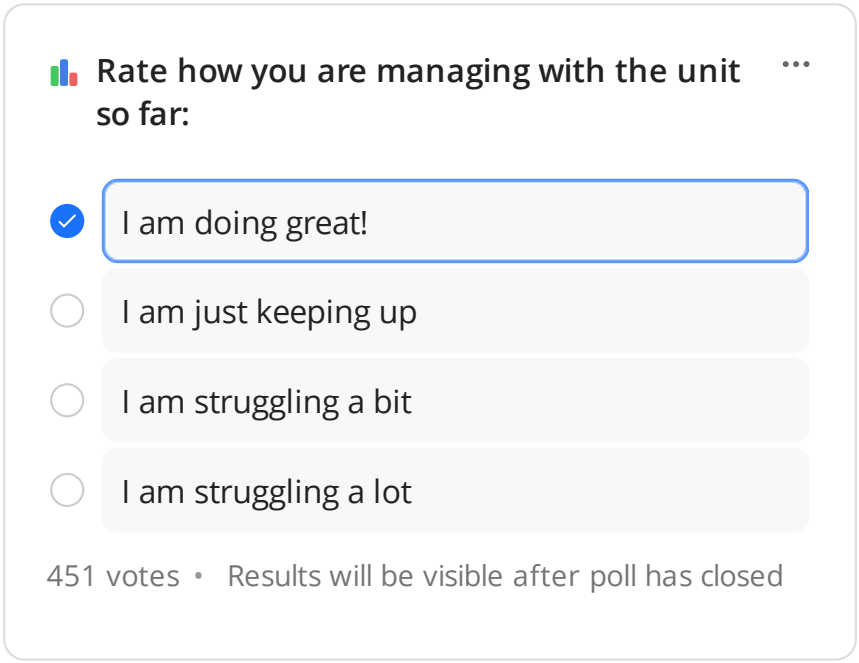
Float

Boolean

✔ Week 3 Attendance and Check-in

Do this in the tutorial this week. **You need to be at the tutorial in Week 3 to have this task marked complete.**

Answer the survey below, click the submit button in the workspace then make sure you ask the tutor to mark your submission complete in the tutorial (you must be at the tutorial for the tutor to mark it complete).



3.1T Read Album Without Tracks from Terminal

Use the code provided to get started.

You must enhance the code provided as follows:

1. Add the missing code to the function/method `read_album()`
2. Add the missing code to the procedure/method `print_album()`
3. Optional: Add an `initialize()` method to the Album class/record definition.

The code should work similar to the following when run:

```
Enter Album
Enter album name:
Greatest Hits
Enter artist name:
Don McClean
Enter Genre between 1 - 4:
2
Album information is:
Greatest Hits
Don McClean
Genre is 2
Classic
```

Once your program is running upload a screenshot to the workspace.

Video

3.2T Read Single Track from Terminal

Write code to read in a single track from the terminal.

You must:

1. Write a class/record with two attributes as follows:

- `name` - eg: "Crackling Rose"
- `location` - eg: "sounds/01-Cracklin-rose.wav"

Remember the form for creating a class/record in Ruby:

```
class Classname
    attr_accessor :attribute_a :attribute_b
    def initialize(attribute_a, attribute_b)
        # body here
    end
end
```

2. Write a function called read_track() which prompts for a track and a location then reads in the `name` and `location` and returns a new `Track` record.

3. Write a function called print_track() which takes as an argument a `track` record and then prints out "Track name:" then the track name and "Track location: " then the track location.

4. Write a `main()` to call `read_track()` then `print_track(track).`

The program should look as follows when run:

```
[user@sahara ~]$ ruby track_terminal_answer.rb
Enter track name:
Crackling Rose
Enter track location:
sounds/01-Cracklin-rose.wav
Track name: Crackling Rose
Track location: sounds/01-Cracklin-rose.wav
[user@sahara ~]$
```

4.1T M □ □ Read Multiple Lines from file

Start this task in Week 4 - but do the loop component in Week 5. Make sure in Week 4 you understand what is happening.

Files allow you to store data persistently. In this task you will write a simple file reading program to read multiple lines using a loop printing each line read to the terminal screen.

To explore this topic, we will modify a program that will, when complete:

- Open a file and write a number of lines to the file. Close the file.
- Open a file and loop according to the number of lines to read in each line.
- Print each line that is read (as they are read).

Use the code provided to get started, using this code complete the following:

1. Open and look at the code provided for reading and writing the lines from files. The functionality of this code is basically correct, but the code can be improved in design and implementation, these are the modifications you will make.

2. Make the following modifications the **basic_read_write.rb** program so that it:

• **in Week 4 do**: Improve the functional decomposition by removing as many lines of code from main as possible, yet retaining good structure.

NB: you must change `read_data_from_file()` and `write_data_to_file()` so that they take a text filename rather than a file descriptor.

Hint: you need to change what you are given so as to have the following procedures: `read_data_from_file(file_name)` and `write_data_to_file(file_name)` - note the parameter change from file descriptor to filename.

• **In Week 5 do:** Uses a loop in `read_data_from_file()`, with the loop controlled by the number at the start of the file.

The program output should look as follows:

```
MacBook-Pro-6:4.1T File Handling mmitchell$ ruby basic_read_write_answer.rb
Fred
Sam
Jill
Jenny
Zorro
MacBook-Pro-6:4.1T File Handling mmitchell$
```

4.2T Read Single Track From File

Write code to read in a single track from a **file** (make sure you use `track.txt` as provided).

> You can take your code from Task 3.2 (Read Single Track from Terminal) and modify it for this task.

You must:

1. Use (write) a class/record with two attributes as follows:

- `name` - eg: "Crackling Rose"
- `location` - eg: "sounds/01-Cracklin-rose.wav"

Remember the form for creating a class/record in Ruby:

```
class Classname
    attr_accessor :attribute_a :attribute_b
    def initialize(attribute_a, attribute_b)
        # body here
    end
end
```

2. Write a function called `read_track(a_file)` which takes as an argument an open file descriptor and reads in the `name` and `location` from the file and returns a new `Track` record.

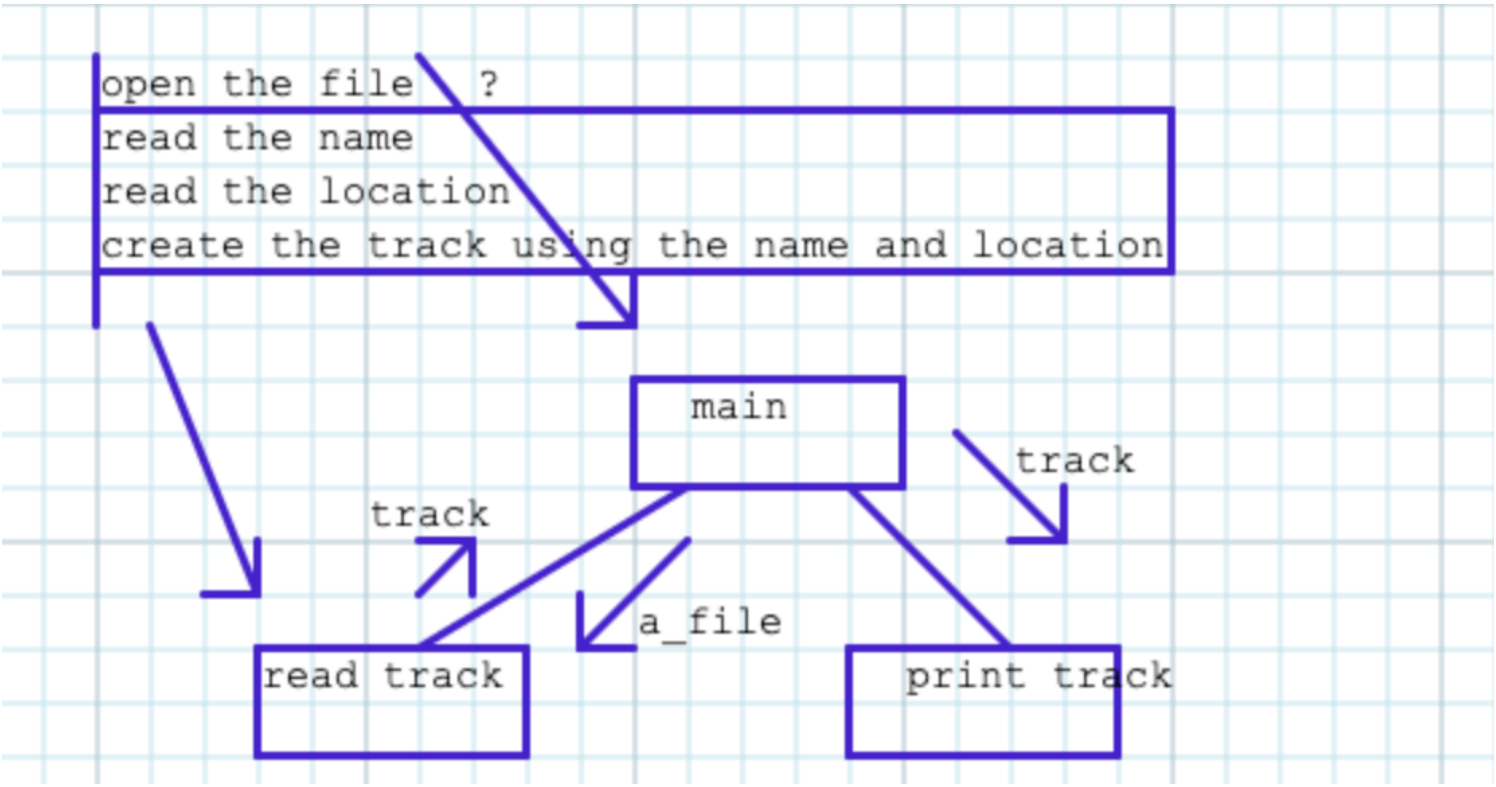
3. Write a function called `print_track()` which takes as an argument a `track` record and then prints out "Track name:" then the track name and "Track location: " then the track location.

4. Write a `main()` to open a file, call `read_track(a_file)` then `print_track(track)` to print the track to the terminal.

Your program should look as follows when run:

```
[user@sahara ~]$ ruby read_track_from_file_answer.rb
Track name: Crackling Rose
Track location: sounds/01-Cracklin-rose.wav
[user@sahara ~]$
```

The structure should look as follows:


open the file ?
read the name
read the location
create the track using the name and location
main
track
track
a_file
read track
print track

♦Using the Debugger to step through a Ruby program

Stepping through a Ruby program

In this case we will step through a Ruby program line-by-line.

> i To run this on your own machine type in the terminal or command line: `gem install byebug`. Then Save the code here to your machine (with Ruby installed) save it as `step.rb`. Run the debugger as: `byebug step.rb`

1. Run the program in the terminal. Run the debugger as:

`byebug step.rb`

Follow the prompt.

2. Now follow the instructions below to use the debugger:

In the Terminal type: (or here: https://replit.com/languages/ruby)

byebug step.rb

Type 'list' - what is shown?

Type 'step' - what is displayed? What line of the program is being indicated?

Type 'step' again - what line are we up to now?

Type 'step' once more - what has happened?

Keep typing 'step' until the pointer is at line 16, then type 'step' again before typing in your name,

Type 'step' again then type 'var local'

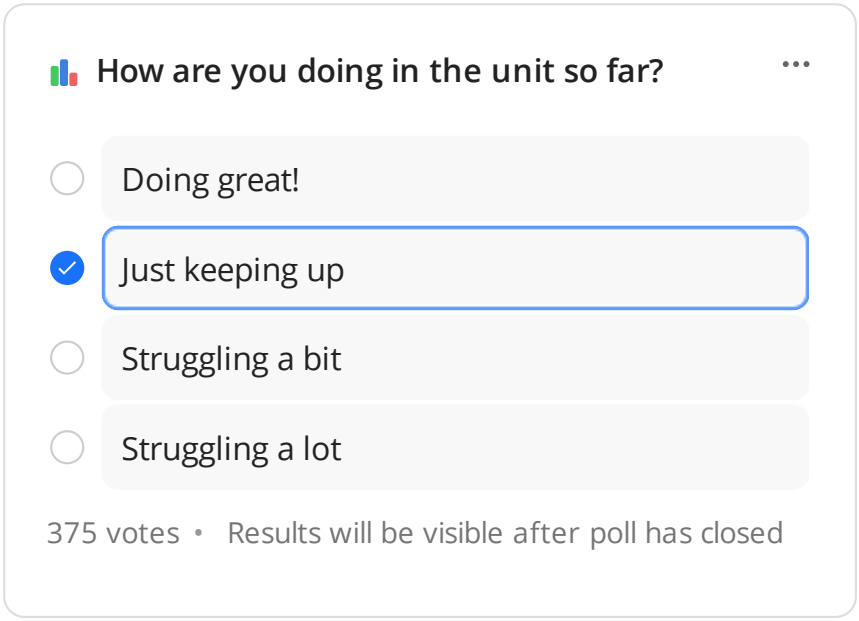
What is displayed?

3. Follow the instructions in *2*. again but this time enter "Blinky" as the name.

✔ Week 5 Attendance and Check-in

Do this in the tutorial this week. **You need to be at the tutorial in Week 5 to have this task marked complete.**

Answer the survey below, click the submit button in the workspace then make sure you ask the tutor to mark your submission complete in the tutorial (you must be at the tutorial for the tutor to mark it complete).



5.1T M □ □ Simple Menu Program

Modify a small program that will give the user the following options:
Display a menu that offers the user the following options:

> 1. Read in Albums
>
> 2. Display Albums
>
> 3. Exit the application

Option 1 should just call a **stub** - i.e a piece of code acting as a place holder until we write the actual content later. In this case, the stub code should just display the following message:

> "You have selected Read Albums'. Press enter to continue."

Option 2 should take you to another (sub) menu as follows:

> 1. Display all Albums
>
> 2. Display Albums by Genre
>
> 3. Return to Main Menu

Options 1 and 2 should again take you to a stub code.

The sample output below shows what your program should look like when run:

```
[user@sahara ~]$ ruby simple_menu_answer.rb
Main Menu:
1 Load Albums
2 Display Albums
3 Exit
Please enter your choice:
1
You selected Load Albums. Press enter to continue

Main Menu:
1 Load Albums
2 Display Albums
3 Exit
Please enter your choice:
2
Display Albums Menu:
```

```
1 Display All Albums
2 Display Albums by Genre
3 Return to Main Menu
Please enter your choice:
1
You selected Display All Albums. Press enter to continue

Display Albums Menu:
1 Display All Albums
2 Display Albums by Genre
3 Return to Main Menu
Please enter your choice:
2
You selected Display Albums By Genre. Press enter to continue

Display Albums Menu:
1 Display All Albums
2 Display Albums by Genre
3 Return to Main Menu
Please enter your choice:
3
Main Menu:
1 Load Albums
2 Display Albums
3 Exit
Please enter your choice:
3
[user@sahara ~]$
```

Video guide.

⚠ When you are ready click "Mark" to see if your program produces the correct output. Once it does ask your tutor to check your code and mark it Complete.

Using a menu in the Music Player

Once you have your menu working from 5.1T you can modify it for your music player (Task 9.1.1).

Copy in the code from 5.1T to 9.1.1 and adapt it to meet the specifications. At this stage just use 'stub' code for the options 1 and 3 in the Main Menu and for options 1 and 2 in the Display Album Menu.

We will gradually add in the full code for each menu item as the weeks progress.

```
Main Menu:
1 Read in Albums
2 Display Albums
3 Select an Album to play
5 Exit the application
Please enter your choice:
```

```
 Display Album Menu:
 1 Display all Albums
 2 Display Albums by Genre
 3 Back to Main Menu
 Please enter your choice:
```

5.2P Static Type Checking in Ruby

You can do these in Ed or the on-campus lab (or your own computer). For your own computer type: `gem install steep` then you can run steep.

To use steep in the standard way we need to create two folders.

1. **`lib`** For our code and its input files. In this case this means `track.rb` and `track.txt` (the track data our code reads)
2. **`sig`** For the `.rbs` file (in our case `track.rbs`) which was produced by `typeprof` and describes the types in our `track.rb` program.

Take a look at these two folders in the Ed workshop and make sure you understand what is in each.

Steep will know where to find our program and its RBS type description by looking at a file called `Steepfile`.

our Steepfile is in the parent directory of **`lib`** and **`sig.`** Find the Steepfile in the Ed workspace and check its contents - it should look as follows:

```
target :lib do
    check "./lib"
    signature "./sig"
end
```

The Steepfile file tells `steep` where to find the code it is checking (in **`./lib`**) and the type description (in . **`/sig`**).

Run the steep program in the parent directory in Ed as follows:

```
[user@sahara ~]$ steep check
```

You should see the following output:

```
[user@sahara ~]$ steep check
# Type checking files:

.....................................................................F

lib/track.rb:18:2: [error] The method cannot return a value of type `::Track` because dec
lared as type `nil`
|   ::Track <: nil
|
| Diagnostic ID: Ruby::ReturnTypeMismatch
|
└   return track
    ~~~~~~~~~~~~

lib/track.rb:32:14: [error] Cannot pass a value of type `nil` as an argument of type `::T
rack`
|   nil <: ::Track
|
| Diagnostic ID: Ruby::ArgumentTypeMismatch
|
└   print_track(track)
                ~~~~~

Detected 2 problems from 1 file
[user@sahara ~]$
```

Reset

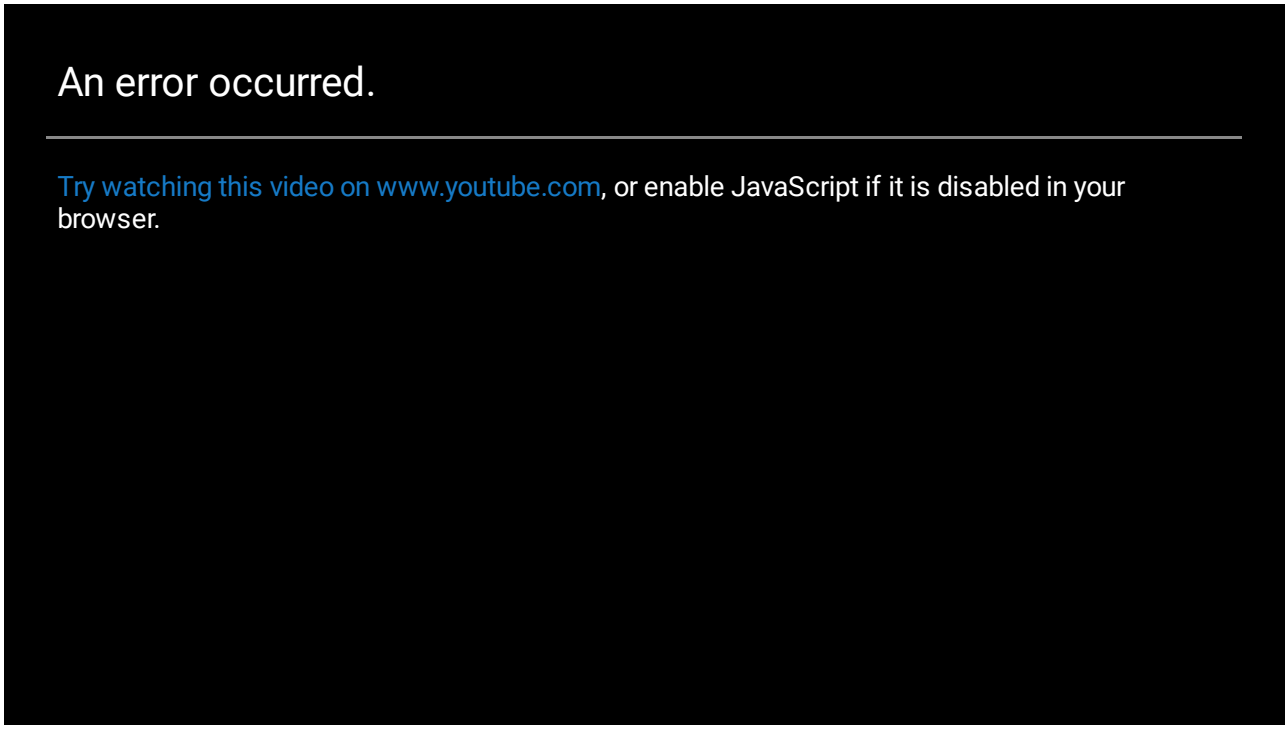
YOUR TASK: Fix the .rbs file so that you produce the following output:

```
# Type checking files:

.....................................................................

No type error detected. 🍵
```

Tip: the problem is on line 6 of the track.rbs file - check the track.rb code to see what this line should look like.

Ask your tutor to mark this once you are done.

Nesting Code (just watch until 3.20)

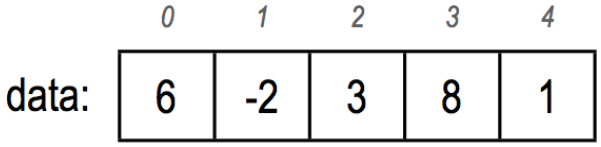


6.1T M □□ Hand Execution - Arrays - Part 1

Arrays are incredibly useful, and really important to understand. Hand execution using trace tables can help you understand these more fully.

To represent an array draw a box divided into sections for each value. This represent the array's location in memory, providing you with access to the array as a whole and to the individual elements.

The following is an example of how to represent an array for your hand execution using a trace table:



Tip: It is useful to include the indexes above the data. This will help make it easy to lookup the values in the array.

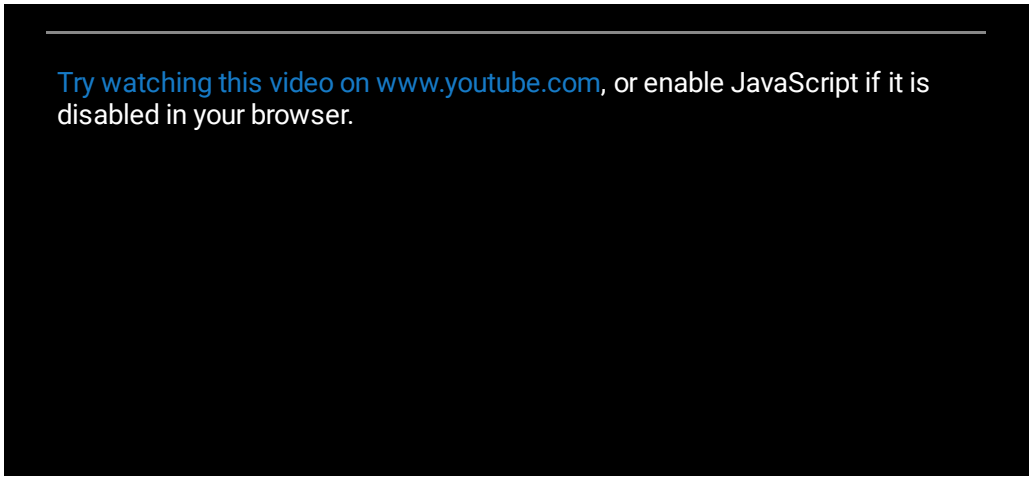
```
def whatdoesthisfunctiondo?(data)
    result := 0;
    i = 0
    while i < data.length
        result := result + data[ i ];
        i = i + 1
    end
    return result
end
```

Complete a trace table (using the answer sheet) for the following two arrays:

data	*Result*
[6, -3, 3, 8, 1]	
[2, 6, -2, 3]	

To remind you of how to do trace tables please watch the following:

An error occurred.



Use this answer sheet

Tutorial Task - Hand execution - Arrays.docx

Upload to the Ed workspace your answer sheets (or a photo of them if needed) showing:

1. a trace table showing the workings of your hand execution

(you may want to do this using an online drawing tool eg: https://print-graph-paper.com/virtual-graph-paper)

2. the answer for the data provided

3. the name you chose for the function in the code above.

6.2T Read Multiple Tracks (from file)

Use the code provided to get started.

You must enhance the code provided as follows:

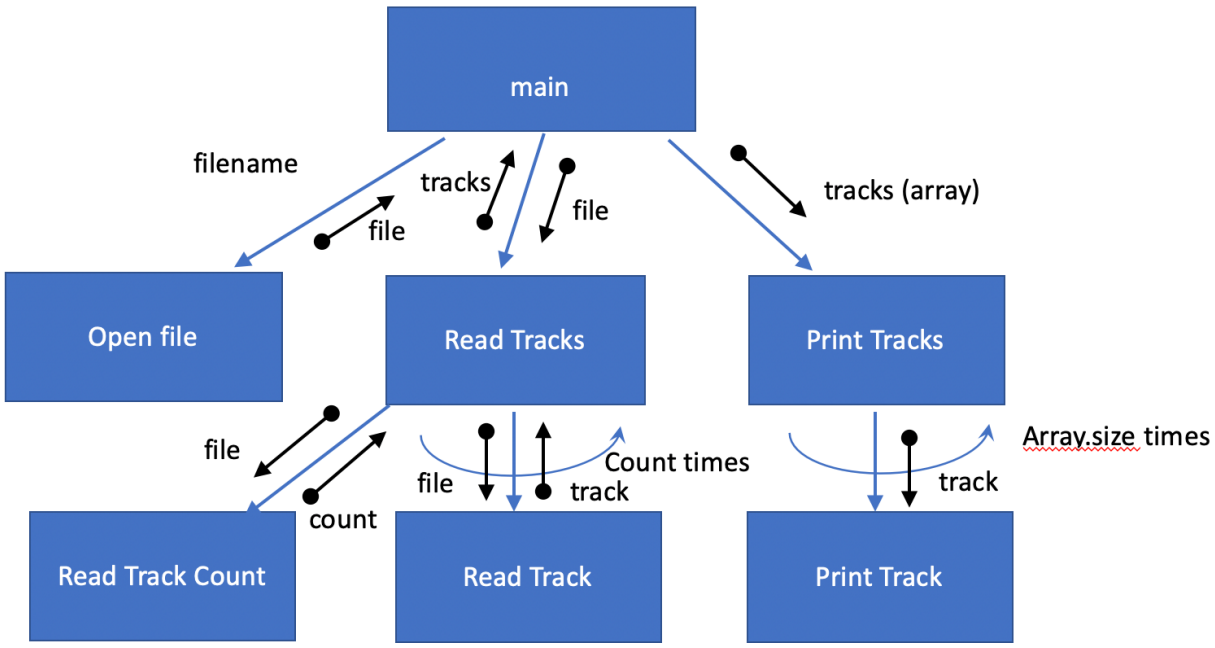
- Complete the provided code so that it reads in an array of tracks then prints them out.

The program output (using the file input.txt) should look as follows:

```
MacBook-Pro-6:TuteTasks mmitchell$ ruby music_tracks_only.rb
Crackling Rose
sounds/01-Cracklin-rose.wav
Soolaimon
sounds/06-Soolaimon.wav
Sweet Caroline
sounds/20-Sweet_Caroline.wav
MacBook-Pro-6:TuteTasks mmitchell$
```

Once your program is running upload a screenshot to the workspace.

Below is a structure chart for this task:



See this video for help with arrays:

https://commons.swinburne.edu.au/items/59365216-7a95-4020-b70a-c2c5350f0c05/1/?search=/searching.do&index=5&available=146

Listing tracks in the Music Player

Once you have completed Task 6.2T you can start to work on the *Play Album* option in the Music Player (Menu Option 3 of Task 9.1.1).

Copy in your code and implement Menu Option 1 (`load_albums()` - but at this stage just load the tracks for one album - remember to copy the tracks input file from T6.2). Once you have loaded the tracks, pass them into `play_album(tracks)` then list the tracks as follows:

```
1  Track name: Crackling Rose
2  Track name: Soolaimon
3  Track name: Sweet Caroline
Select a Track to play:
1
```

Then use *read_integer_in_range()* from the input_functions() to allow the user to select one track from the list. Then display a message that the track is playing:

```
1  Track name: Crackling Rose
2  Track name: Soolaimon
3  Track name: Sweet Caroline
Select a Track to play:
1
Playing track Crackling Rose from album Greatest Hits
```

Later on we can change this code to read in all the albums and first display the albums, and select an album before listing the tracks for that albums.

GOSU Resources

See this guide:

IntroProg-Week04i-supplement-1.pdf

Sobkowicz, M 2015 *Learn game programming with Ruby : bring your ideas to life with Gosu*, The Pragmatic Bookshelf (See chapter 7 for help the grid aspect of the Maze Task)

https://www.rubydoc.info/gems/gosu/Gosu/

Gosu site

Gosu game video tutorial

6.3C M □ □ Drawing Shapes using GOSU

In this task you use Gosu to create a program that draws a picture.

GOSU is a development environment that makes it easy to create programs that use graphics, sounds, animations and other aspects relevant to creating small interactive games.

Follow these **3** steps:

1. Copy the code provided to your IDE (both `gosu_shapes.rb` and `circle.rb`) and use the demonstration shapes to create a picture of your own design. Your picture should include at least 3 different types of shapes (eg: a triangle, a rectangle and a circle)

> Use the following site to select colours for the circle (which uses RGB values):
> https://www.rapidtables.com/web/color/RGB_Color.html
> Or Use the Gosu colour constants:
> https://www.rubydoc.info/gems/gosu/Gosu/Color

> eg: a Red circle with a radius of 50 pixels would be produced by the two statements:

```
img = Gosu::Image.new(Circle.new(50))
img.draw(200, 200, ZOrder::TOP, 0.5, 1.0, Gosu::Color::RED)
```

> Or you could use the HEX values:

```
img.draw(300, 50, ZOrder::TOP, 1.0, 1.0, 0xff_ff0000)
```

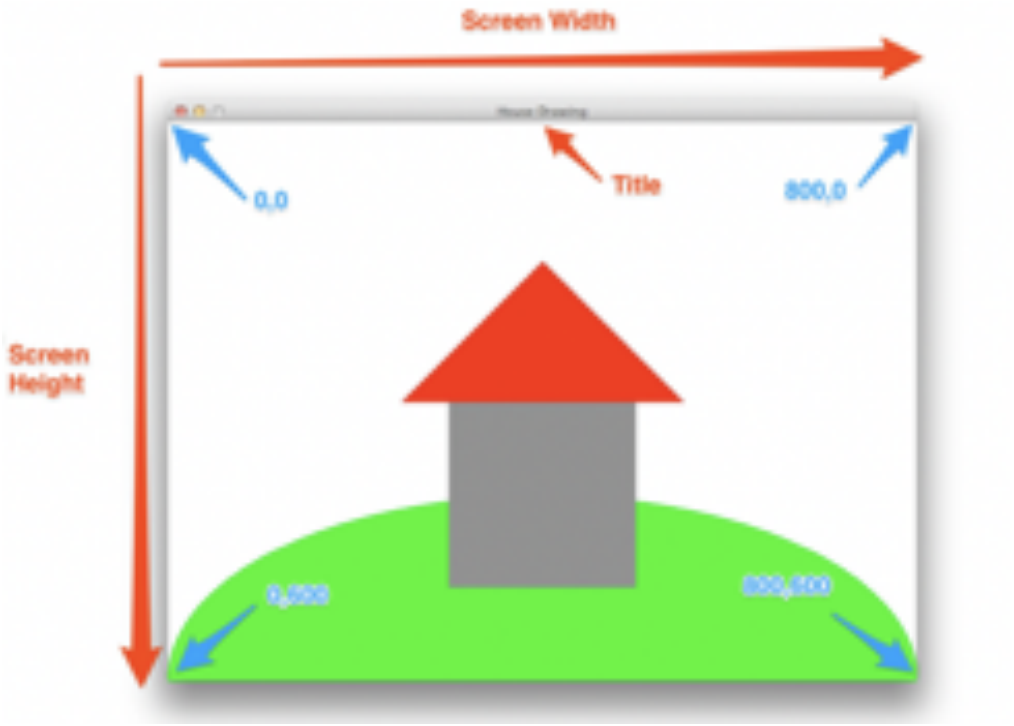
> See here to work out HEX values:
> https://www.binaryhexconverter.com/decimal-to-hex-converter

2. Run the code provided and study it to understand what is happening (you may need to run the command `gem install rubygems` to use the `circle.rb` code).

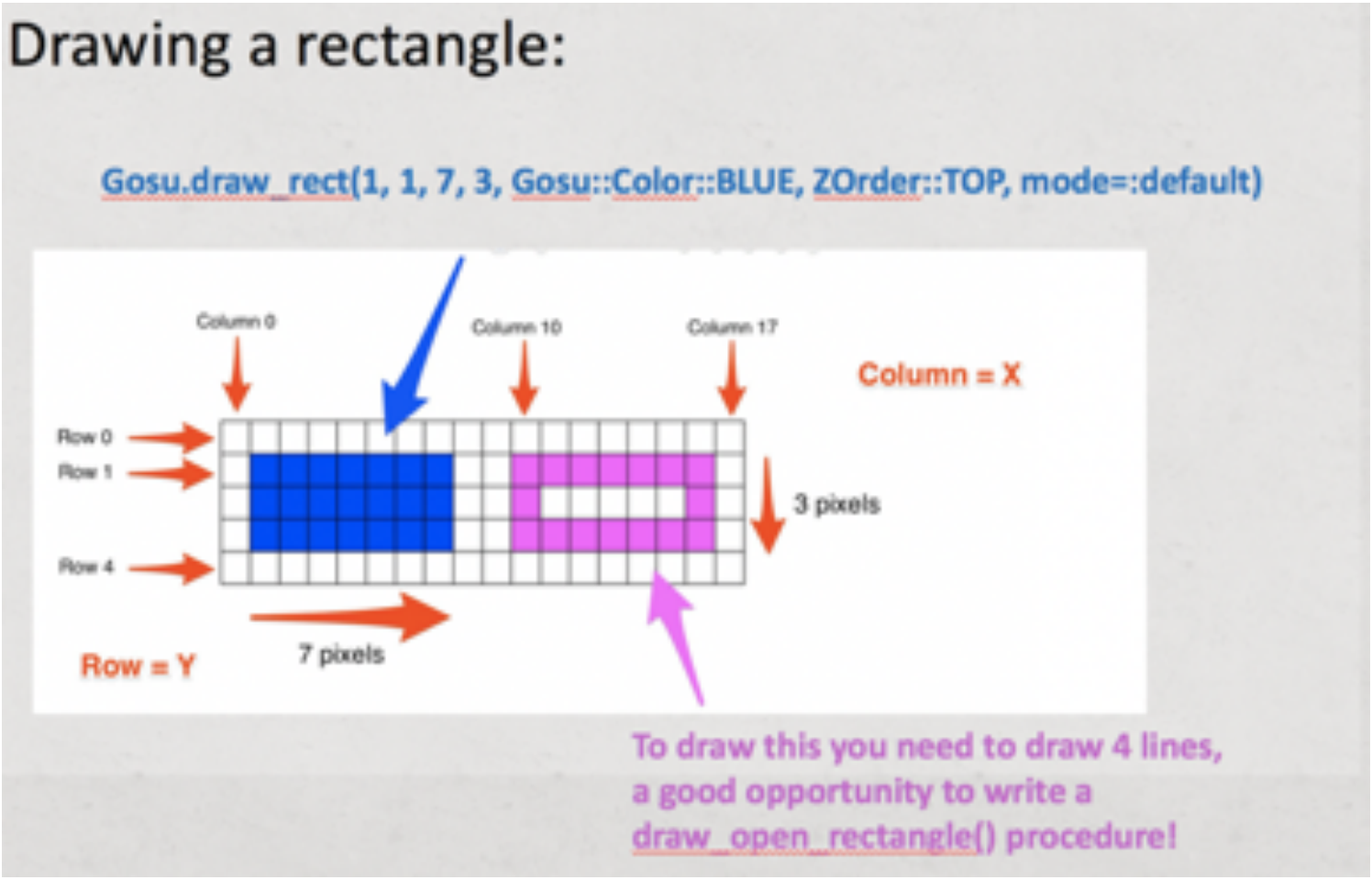
3. Using an IDE like Visual Studio Code, change the code to draw your own unique picture. You might want to draw it on paper first (perhaps graph paper or an electronic equivalent like this).

Video Guide to the following.

> The co-ordinate system works as follows



> Example:



You might want to first draw your shape on graph paper:

https://print-graph-paper.com/virtual-graph-paper

Once your code is complete submit a screen shot.

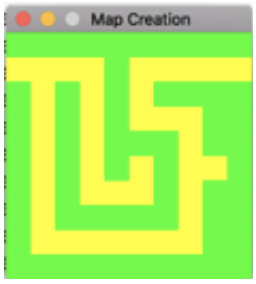
See also this guide:

IntroProg-Week04i-supplement-1.pdf

6.4D M Maze Creation

You must complete the code so that it works as follows:

1. The code in the `initialize()` procedure of the `Gosu` window should be completed to set up cells that are connected to each other with variables joining each cell to its neighbours (using references).
2. The user should be able to left click on cells on the screen to create mazes (and later in the Maze Search task we will use recursion to find a path through the maze).
3. Each cell clicked on should turn yellow.



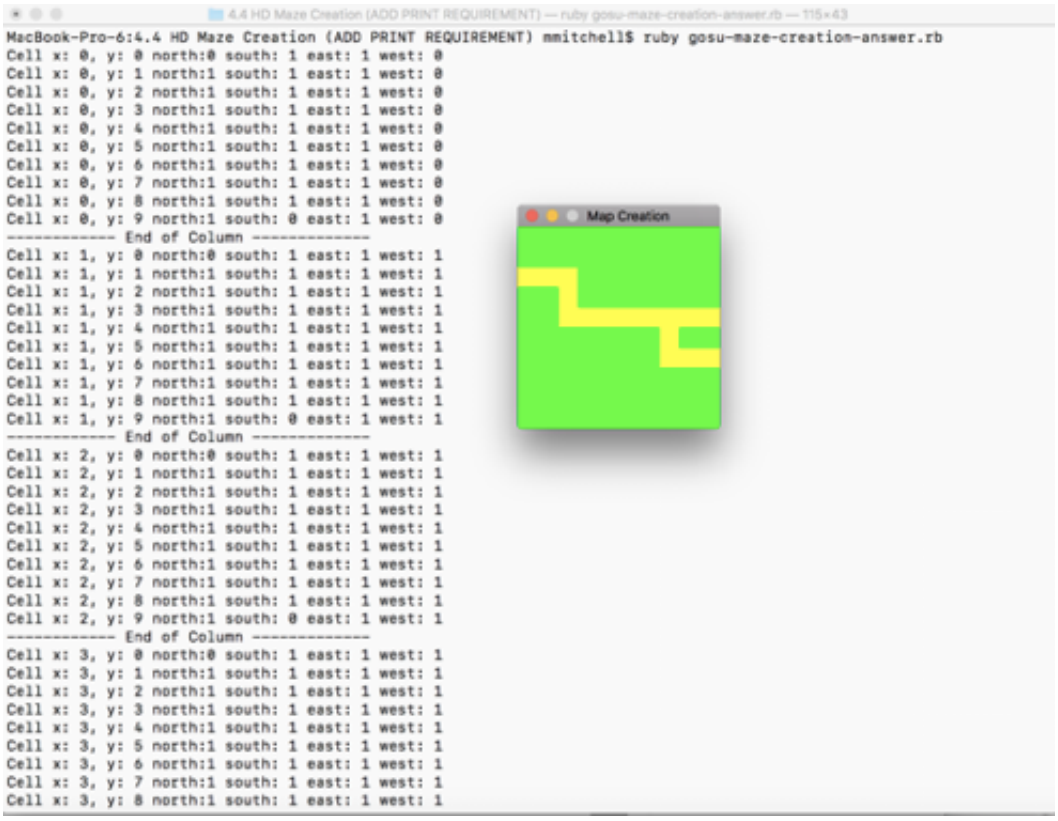
4. Once this is working (or perhaps before) add code to print out each cell and indicate whether the reference to the neighbour on each side is nil or not, as per the following:

```
Cell x: 0, y: 0 north:0 south: 1 east: 1 west: 0
```

In this case there is no neighbour to the north or the west.

(NB: whether the east or the west neighbour is nil will depend on your perspective – i.e is your perspective looking into the screen, or out of the screen)

Your submitted screenshot should look something like the following:



See the video demonstration here.

Once you have the program working as required submit a screenshot to the workspace.

Sobkowicz, M 2015 *Learn game programming with Ruby : bring your ideas to life with Gosu*, The Pragmatic Bookshelf (See chapter 7 for help the grid aspect of the Maze Task)

Gosu Ruby Documentation

Gosu site

Gosu game video tutorial

Sobkowicz, M 2015 *Learn game programming with Ruby : bring your ideas to life with Gosu*, The Pragmatic Bookshelf (See chapter 7 for help the grid aspect of the Maze Task)

Gosu Ruby Documentation

Gosu site

Gosu game video tutorial

Check Point: Feedback on Tasks up to Topic 6

By the end of Topic 7 you should have received feedback on the manually marked tasks up to and including 5.3P.

If you needed to REDO or FIX and RESUBMIT any of these tasks, you should look at the tutors feedback in the task submission and make any corrections. **If you have not submitted these tasks by now you will not get feedback and risk failing that task.**

Once your corrections are complete you MUST notify your tutor and ask that the task be re-marked.

✔ Week 7 Attendance and Check-in

Do this in the tutorial this week. **You need to be at the tutorial in Week 7 to have this task marked complete.**

Answer the survey below, click the submit button in the workspace then make sure you ask the tutor to mark your submission complete in the tutorial (you must be at the tutorial for the tutor to mark it complete).

How you are managing in the unit so far?

- Doing great!
- Just keeping up
- Struggling a bit
- Struggling a lot

7 votes

7.1T M □ □ Concept Map

In your tutorial/lab groups produce a concept map in preparation for the test next week. You should cover the concepts from the lessons and tasks. First make a list of terms, then draw a diagram that organises those terms into a comprehensive overview that relates the concepts and terms to each other.

You could use a tool like Creately to create your concept map.

Here are some of the things you should cover:

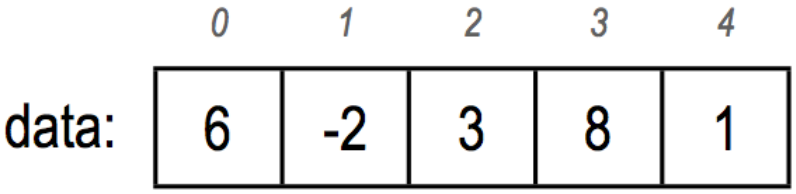
Concepts	Artefacts	Action	Terminology
Sequence	Program	Assignment	Statement
Selection	Procedure	Function Call	Expression
Repetition	Function	Procedure Call	Identifier
Type	Variable	If	Parameter
Value	Constant	Case	Global Variable
	Record	Repeat	Local Variable
	Enumeration	While	Condition
	Pointer		
	Variable		

Submit your completed concept map to the workspace.

7.2T M □ □ Hand Execution - Arrays - Part 2

In this task we again hand execute a program using trace tables. This task also involves arrays.

The following is an example of how to represent an array for your hand execution using a trace table:



Tip: It is useful to include the indexes above the data. This will help make it easy to lookup the values in the array.

```
def whatshouldthisfunctionbecalled?(data, val)
    i = 0
    result = false
    while i < data.length
        if data[i] == val
            result = true
            return result
        end
        i = i + 1
    end
    return result
end
```

Complete a trace table (using the answer sheet) for the following two arrays and parameter values:

data	*val*	*Result*
[2, 6, -4, 3, 7]	3	
[-2, 8, 2, -5, 9]	6	

Use this answer sheet:

Tutorial Task - Hand execution - Array Search.docx

Upload to the Ed workspace a document (or photo) of your answer sheets showing:

1. a trace table showing the workings of your hand execution

2. the answer for the data provided

3. the name you chose for the function in the code above.

4. Suggestion as to how the design of the code could be improved in line with structured principles?

7.3P Array Search

In task 6.2 you did the following:

Read in a number of tracks **from a file**.

In THIS task you must: ***(NB. Build on your code for Task 6.2)***

·Once the track information is loaded, prompt the user to enter a search string (use the `read_string()` function from the `input_functions` library.

·The search function should have the following prototype: `search_for_track_name(tracks, search_name)` – where `tracks` is an array of tracks and `search_name` is the string being searched for. The function should return -1 if the track name is not found, otherwise it should return the array index of the track with that name.

·Display a message indicating if the track has been found or not (this is provided in the sample code in Resources)

·Make sure you test for a track that does exist and one that does not.

An example of the expected output of your program is provided below:

```
Crackling Rose
sounds/01-Cracklin-rose.wav
Soolaimon
sounds/06-Soolaimon.wav
Sweet Caroline
sounds/20-Sweet_Caroline.wav
Enter the track name you wish to find:
Soolaimon
Found Soolaimon
 at 1
MacBook-Pro-6:6.2 T Array search mmitchell$
```

7.4C M □□ Custom Program Design

In this task you will provide a plan and overview of the structure of a custom program (something you would be interested in creating).

Credit Task 6.3 - Custom Program Design.docx

Specifically it should:

1. Demonstrate the use of functional decomposition - implement the program with a number of functions and procedures. (Maybe even modular decomposition with separate units if you can identify some reusable artefacts - optional but nice)

2. Demonstrate the use of arrays and records

3. Demonstrate the use of structured programming (sequence, selection, and repetition)

Here are some steps to get you started:

1. Download the Design Report template attached above.

2. Provide a summary of your program — What does it do? What are some of the key features etc.

3. Describe the main data types: records and enumerations.

4. Describe the main functions and procedures. Get some detail down now for your tutor to check, but there is no need to spend ages on this task. Have enough that you can start to see how the program will continue to develop as you proceed.

5. Show your plans to your tutor, lecturer, help desk staffers, and/or friends to get some feedback.

Note: Your program should be different from the food hunter program and the lecture demonstration programs. You want to demonstrate that you have learnt from these tasks and can apply what you have learnt to some other program design.

If you are aiming for a High Distinction, review the related High Distinction Project document for details on how you can ensure this program meets the HD requirements.

Note: Your program should be different from the food hunter program and the lecture demonstration programs. You want to demonstrate that you have learnt from these tasks and can apply what you have learnt to some other program design. If you are aiming for a High Distinction, review the related rubric (attached below) for details on how you can ensure this program meets the HD requirements.

MarkingRubricSummary(4)-1.pdf

Once your report is finished upload it to the workspace.

□ Test 1 □ Week 8 - online outside class times

In Week 8 you will sit a timed test.

This test will be available in a lesson (NOT in these portfolio tasks).

Your tutor will mark your test and give you feedback here on whether you have:

1) Passed

2) Need to fix and resubmit the test

3) Need to REDO the test in another sitting.

8.1T M □ □ Read Album with Tracks (from file)

The program must read in a **single** album and a number of tracks for the album as well as a genre for the album **from a file**. You can have as many genres as you like, but these must be defined using an enumeration. **For each track you must read in a track name and a track filename so re-use the code from task 5.1 T** (also see 5.2T for the album/music record).

Your application must read the album and track information **from the provided album.txt** file. Use the provided code `album_file_handling.rb` as a basis for your program.

The output of your program should look as follows:

Once your program is running upload a screenshot to the workspace.

```
MacBook-Pro-8:6.1T Album File Handling (updated pdf) mmitchell$ ruby music_record_answers.rb
Neil Diamond
Greatest Hits
Genre is 1
Pop
Crackling Rose
sounds/01-Cracklin-rose.wav
Soolaimon
sounds/06-Soolaimon.wav
Sweet Caroline
sounds/20-Sweet_Caroline.wav
MacBook-Pro-8:6.1T Album File Handling (updated pdf) mmitchell$
```

8.1.1P Printing the Genre Names

Look at the lecture notes for Week 7 (you may need to watch the video) and copy in (below) the code for printing the genre names using the array. Just place your code in `main().`

The output should look as follows:

```
1 Pop
2 Classic
3 Jazz
4 Rock
```

Make sure your output will change if you add an element (eg "Blues") to the array.

Listing the albums for genre in the Music Player

Once you have completed Task 7.3P you can start thinking about menu option 2 of the Music Player task (9.1.1).

Menu option 2 is "Display Albums" - it has two additional options "*Display All Albums*" and "*Display Albums for Genre*":

```
Display Album Menu:
1 Display all Albums
2 Display Albums by Genre
3 Back to Main Menu
Please enter your choice:
2
1. Pop
2. Classic
3. Jazz
4. Rock
Pick a Genre:
2
```

Once you can read in all the albums (you need to finish Task 8.1T first) then you can use the code from 7.3P Array Search to select which albums to display for the "*Display by Genre*" option.

For this to work you will need to bring together the following:

- 8.1T Read Albums with Tracks (from file) - but you will need to add a loop first to read in multiple albums - this will give you the code for the "*Load Albums*" option in the Music Player.
- Pass the albums into Display Albums
- Allow the user to select "Display by Genre"
- List the genres (see the Task 8.1.1P - Printing Genre names)
- Allow the user to select a Genre using the `read_integer_in_range()` from the `input_functions.rb.`
- Then modify the code from 7.3P to display only the albums that match the selected genre as follows:

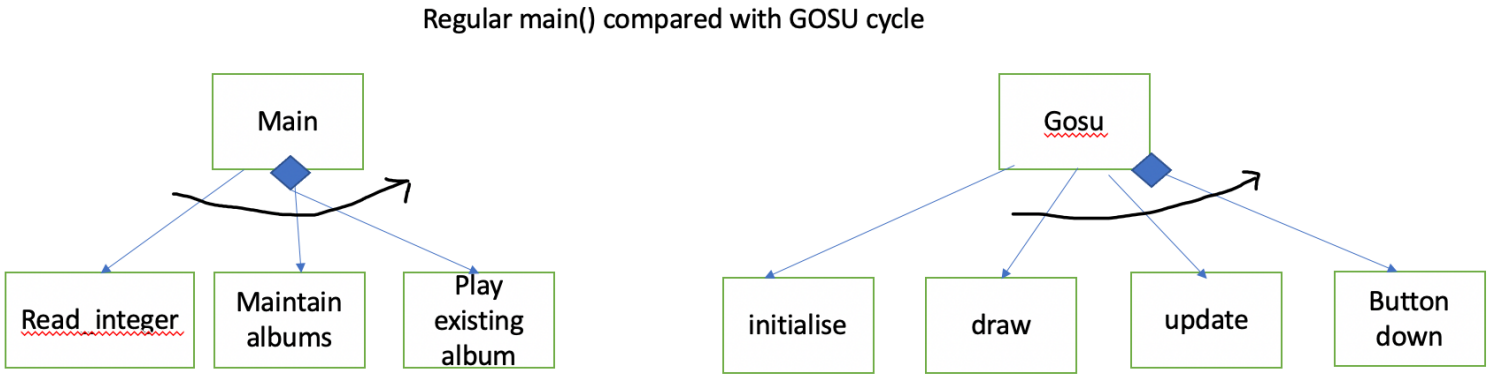
```
Display Album Menu:
1 Display all Albums
2 Display Albums by Genre
3 Back to Main Menu
Please enter your choice:
2
1. Pop
2. Classic
3. Jazz
4. Rock
Pick a Genre:
2

 Title: Greatest Hits Artist: Neil Diamond Label: Universal Genre: Classic
 Title: American Pie Artist: Don McClean Label: The Record Plant Genre: Classic
 Title: No Secrets Artist: Carly Simon Label: Elektra Records Genre: Classic
```

8.2C M □ □ Gosu Major Cycle

Modify a simple Ruby program to move a shape across the screen by modifying the tasks in the `Gosu` cycle `update()` and `draw()` methods.

Enhance the code provided as follows:

- □ Add a variable in the `initialize()` method called `shape_x` with the initial value of zero.
- □ Add code into the `update()` method that will add 10 to `shape_x`.
- □ Add code into the `draw()` method that will draw a shape (square or circle of any visible colour) at the *y* coordinate of 30 and *x* coordinate of `shape_x`.



The GOSU cycle can also be represented as:

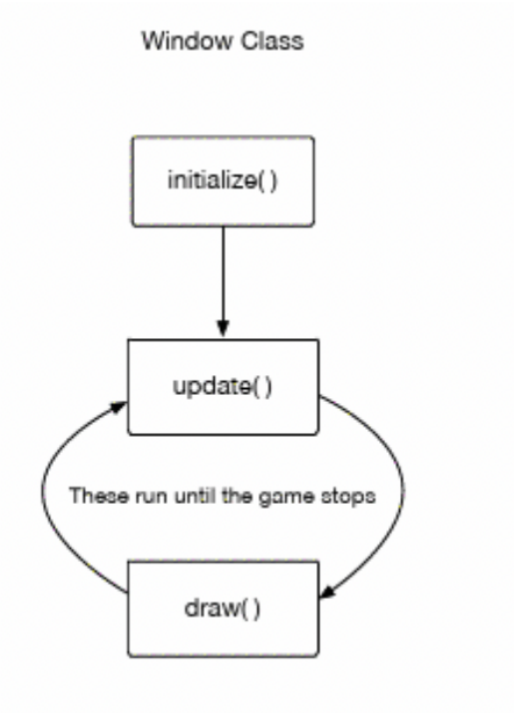


Image above taken from "Sobkowicz, M 2015 *Learn game programming with Ruby : bring your ideas to life with Gosu*" pg 24.

Use the code provided to get started. You also need to download the **media** folder with its contents.

Upload a screenshot of your code running to the workspace.

8.3D M □ □ Hover Button

Make the following corrections to the program hover_button_test.rb:

- The button should have a black border around it when the mouse is moved over it to highlight it.
- At the bottom of the screen there should be a display of the mouse x and y locations at all times (not just when the mouse is clicked)
- When the button is clicked the background should change to yellow, if the window area outside the button is clicked the background should change to white.
- Make sure the button works as per the test data below:

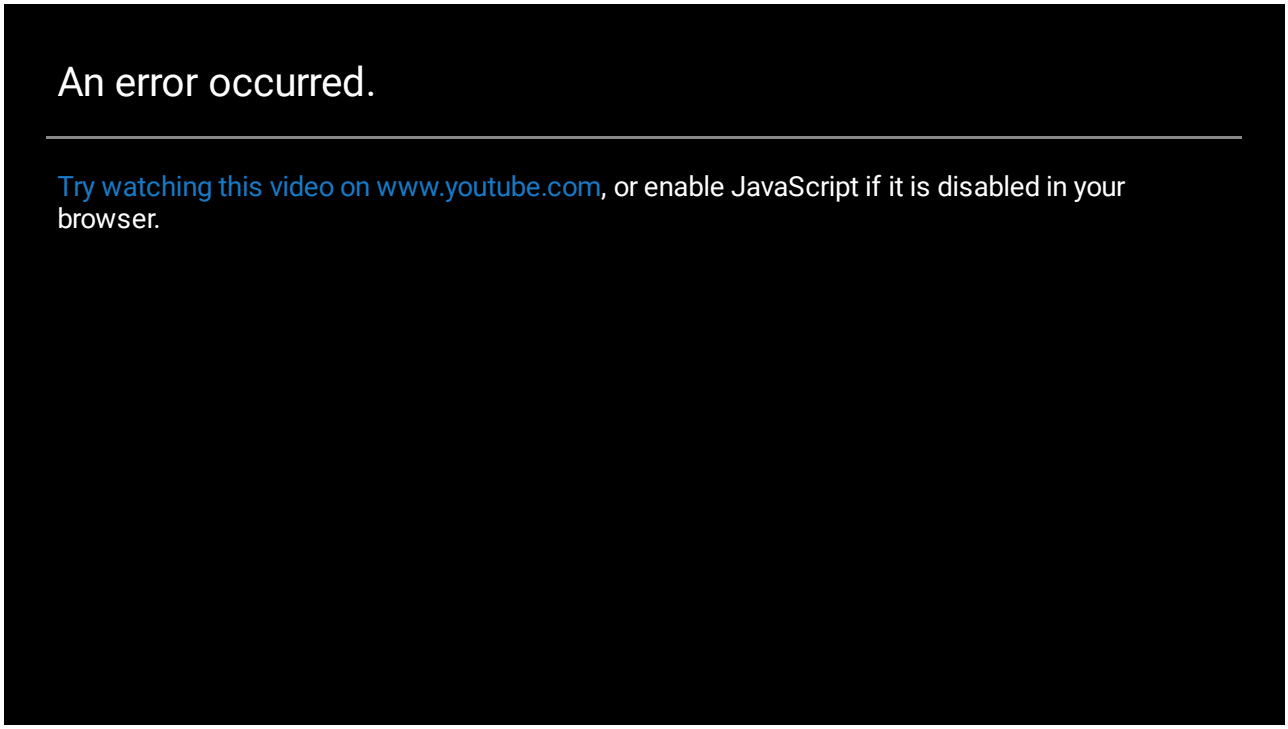
	btnX	btnY	btnWidth	btnHeight		MouseX	MouseY	Answer
Test 1	50	50	100	50		75	75	TRUE
Test 2	50	50	100	50		75	10	FALSE
Test 3	50	50	100	50		75	200	FALSE
Test 4	50	50	100	50		10	75	FALSE
Test 5	50	50	100	50		200	75	FALSE

The screen should look as follows when the mouse is NOT over the button:

The screen should look as follows when the mouse IS over the button:

Once your program is running upload a screenshot to the workspace.

Demonstration of the Text Music Player (Pass and Credit Levels)



✔ Week 9 Attendance and Check-in

Do this in the tutorial this week. **You need to be at the tutorial in Week 9 to have this task marked complete.**

Answer the survey below, click the submit button in the workspace then make sure you ask the tutor to mark your submission complete in the tutorial (you must be at the tutorial for the tutor to mark it complete).

How are you managing in the unit at this stage?

- ○ Doing great
- ○ Just keeping up
- ○ Struggling a bit
- ○ Struggling a lot

2 votes

9.1.1P Major Task Pass Level - M □ □ Text Music Player

⚠ **Do NOT load your song files to Ed.**

BEFORE YOU DO THIS MAKE SURE YOU HAVE DONE THE Read Album with Tracks TASK

⚠ In this task you will extend the implementation of your Text Based Music Player. Use the code and records from your previous tasks.

ℹ **You will need to update the Track to include an extra field: `duration` (String)**

ℹ The grade achievable for this task (within the Pass range) is:

- **Basic Pass Level - 55 (if you complete all Tutorial and Pass tasks to required standards)**

To see how to create the file with the albums (and design the code) see the following:

https://echo360.org.au/media/befb248b-a61b-4965-bb37-cc717e8906d1/public

Resources:

- Frieder, O. Frieder, G. & Grossman, D. 2013 Computer Science Programming Basics in Ruby, O'Reilly Media (Chapter 6)
- Flanagan, D. & Matsumoto, Y. 2008 The Ruby Programming Language, O'Reilly.
- Pine, C 2014, *Learn to Program (2nd Ed), Chapter 11,* The Pragamatic Programmer (library version – follow the link)

Pass Level Requirements

Your Text Based Music Application must have the following functionality:

⚠ **You must use while loops for this task - do not use `for` loops or other loops.**

Display a menu that offers the user the following options:

```
1. Read in Albums
2. Display Albums
3. Select an Album to play
5. Exit the application
```

Menu option 1 should prompt the user to enter a filename of a file that contains the following information:

·The number of albums

·The first artist name

·The first album name

·The record label

·The genre of the album

·The number of tracks

·The name and file location (path) of each track.

·The album information for the remaining albums.

Menu option 2 should allow the user to either display all albums or all albums for a particular genre. The albums should be listed with a unique album number which can be used in Option 3 to select an album to play. The album number should serve the role of a 'primary key' for locating an album. But it is allocated internally by your program, not by the user. If the user chooses list by genre - list the available genres.

Menu option 3 should prompt the user to enter the primary key (or album number) for an album as listed using Menu option 2.If the album is found the program should list all the tracks for the album, along with track numbers. The user should then be prompted to enter a track number. If the track number exists, then the system should display the message "Playing track " then the track name, " from album " then the album name. You may or may not call an external program to play the track, but if not the system should delay for several seconds before returning to the main menu.

NB: IF you are aiming for a CREDIT or higher in the unit, then you should move on to doing the Credit level Music Player.

Submit your final code.

9.1.2C Major Task Credit Level - M □ □ Text Music Player with Update

Your Text Based Music Application must have the following functionality:

⚠ **You must use while loops for this task - do not use `for` loops or other loops.**

The same functionality as in the Pass level Music Player task, but you must add a fourth option "Add an Album" menu option to the main menu as follows:

```
1. Read in Albums
2. Display Albums
3. Select an Album to play
4. Add an Album
5. Exit the application
```

Menu option 4. Your code should allow the user to add another album and multiple tracks to the album.

Your added album **MUST** appear when you choose `2. Display Albums` after adding the album but you **do not** need to update the `albums.txt` file.

The code should work as follows:

```
Main Menu:
1 Read in Albums
2 Display Albums
3 Select an Album to play
4 Add an Album
5 Exit the application
Please enter your choice:
4
Enter title:
new album
Enter artist:
new artist
Enter label:
new label
1. Pop
2. Classic
3. Jazz
4. Rock
Enter number of genre:
4
Enter number of tracks
2
Enter a name for the new track:
new track 1
Enter a location for the new track:
c:
Enter a name for the new track:
new track 2
Enter a location for the new track:
d:
Album added: new album. Press enter to continue.
```

NB: IF you are aiming for a Distinction or higher in the unit, then you should move on to doing the Distinction level Music Player.

Submit your final code.

9.1.3D Major Task Distinction Level - M □ □ GUI Music Player

Do NOT load your song files to Ed. Your tutor will ask you to demonstrate your code in class.

In this task you will build on the skills developed in your other Pass, Credit and Tutorial tasks.

There is a minimum requirement for naming and design before your code can be accepted – regardless of how well the code is functioning.

The grade achievable for this task (within the Distinction range) are:

- **Distinction Level – 70 - This task.**

Note: See the Gosu audio API at https://www.rubydoc.info/gems/gosu/. You will need to use the following: `Gosu::Song.new(filelocations)` – create a new song; and: `song.play(false)` – play the song.

Distinction Level Requirements - 70

Before doing this level complete the tasks 7.1 and 7.2

At this level your program must read in (from a file) at least four albums and up to 15 tracks for each album.

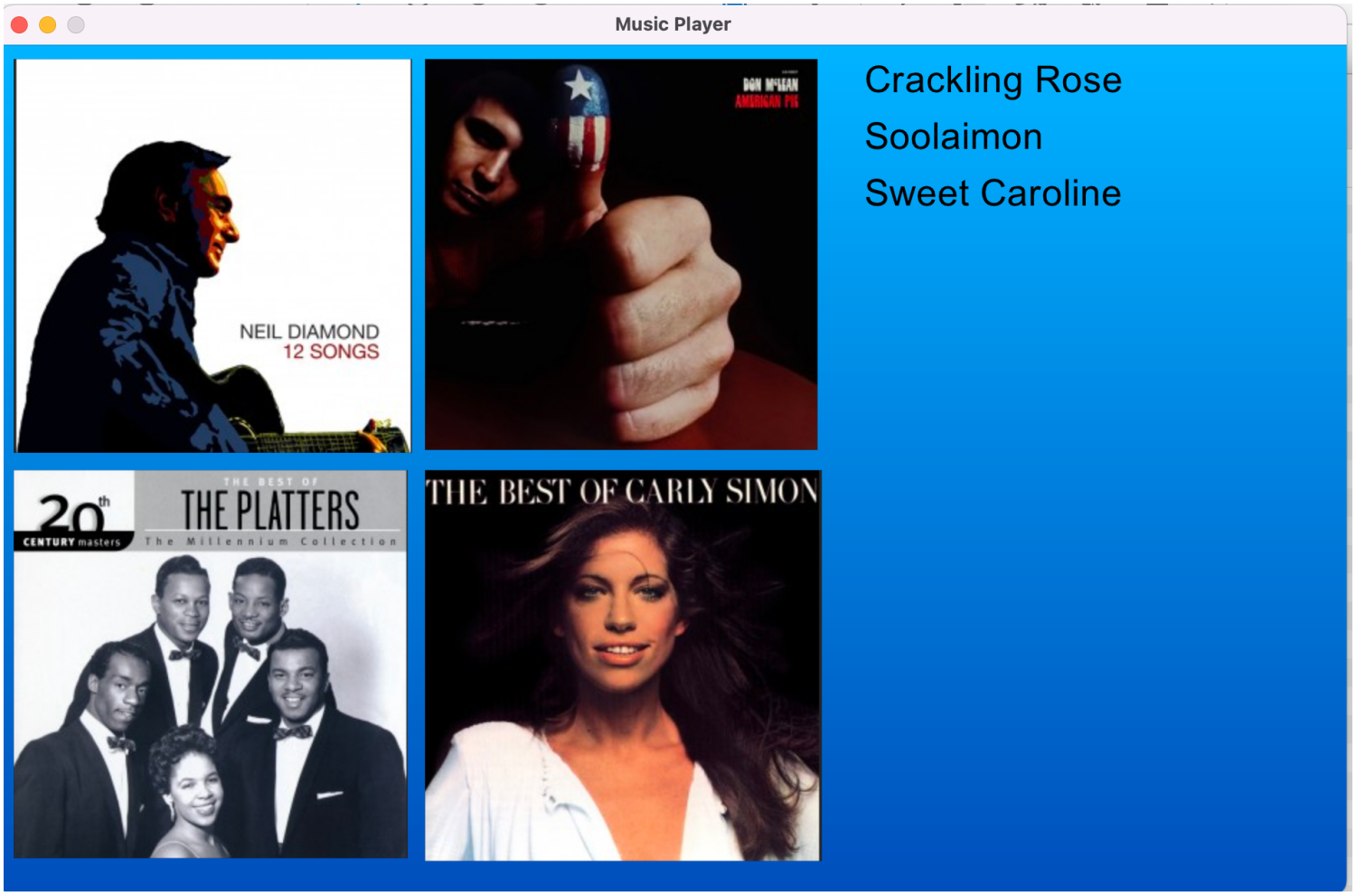
The information read from the file should include:

- Number of Albums
- Album title
- Artist
- Artwork file name (place your artwork in an /images folder under the main folder where you run the program)
- The number of tracks
- The title of each track
- The file location of each track

At this level user interaction must be entirely through a GUI. Your GUI interface should show all the albums using either a text description, artwork or both. Users should be able to click on any Album information (i.e the artwork) and the tracks will be listed. **The user should then be able to click on a track to play that track. The currently playing track must be indicated somehow (e.g the track could be highlighted or display a simple text message 'Now playing ..'). If the user clicks on another track (for the current album or another album) then any currently**

playing track should be stopped and the most recently selected track start playing.

You must use the `Gosu` audio API for this component.



Submit your completed code to Ed.

Custom Program Extensions (for higher Distinction Grades or for High Distinction Custom Program)

You may wish to extend on the requirements above for a higher distinction grade or for your custom program. You would need to discuss this with your tutor as to what is required for different grade levels. Some possible extensions are:

Possible custom program Distinction level Extensions:

- Allow users to page through multiple pages of albums (**required for all D & HD extensions**).
- Automatically discover and load albums from the file system.
- Allow sorting of albums based on year recorded, genre ,etc.

Possible Custom program High Distinction extensions:

- Allow users to select tracks from various albums to create playlists. Users can then select from the playlists to play a pre-selected sequence of tracks.
- Some combination of all the above extensions.

Submit your final code and a screenshot to the workspace.

9.2C Recursive Factorial

Building on the provided code in this week's Lesson Topic (look at the Lisp code for calculating factorials), write a recursive function that calculates a factorial using recursion.

The program will take a number on the command line and calculate the factorial for that number.

Once you have it working correctly, you also need to put in a check in `main()` in case an incorrect argument is passed in. You will need to include the following line of code:

```
puts("Incorrect argument - need a single argument with a value of 0 or more.\n")
```

You need also to add error checking into main. - for this see the lesson "Starting Ruby - Code Practice" - the final slide.

The output should look as follows:

```
MacBook-Pro-6:10.2 P Recursive Factorial (undergrad only) mmitchell$ ruby recursive_factorial_answer.rb
Incorrect argument - need a single argument with a value of 0 or more.
MacBook-Pro-6:10.2 P Recursive Factorial (undergrad only) mmitchell$ ruby recursive_factorial_answer.rb -1
Incorrect argument - need a single argument with a value of 0 or more.
MacBook-Pro-6:10.2 P Recursive Factorial (undergrad only) mmitchell$ ruby recursive_factorial_answer.rb 5
120
MacBook-Pro-6:10.2 P Recursive Factorial (undergrad only) mmitchell$
```

9.3D M □ □ Gosu Shape Moving

Modify a simple Ruby program to move a shape across the screen by changing the provided code in the Gosu cycle `update()` method.

Use the code provided to get started.

You must enhance the code provided as follows:

1. The shape also can be moved up and down
2. The shape does not move out of the window area

Upload a screenshot of your program running to the workspace.

Video

9.4 HD M □□ Maze Search

Use the code you completed in Task 4.4D to get started.

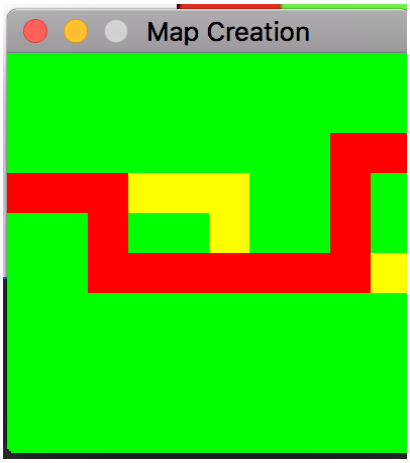
You must complete the code so that it works as follows:

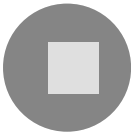
1.Once the user has used the Left Mouse Key to create at least one path from one side of the screen to the other, the user should then be able to Right Click the mouse on a yellow cell on the left most side of the screen.

2.The program should then search using the recursive *search()* function to find a path to the right side of the screen following one of the user's created paths.

3.Each cell on the path should then be displayed in red.

A found path through the maze may look something like:





Once you have the program working as required submit your code and a screenshot to the workspace.

Sobkowicz, M 2015 *Learn game programming with Ruby : bring your ideas to life with Gosu*, The Pragmatic Bookshelf (See chapter 7 for help the grid aspect of the Maze Task)

Gosu Ruby Documentation

Gosu site

Gosu game video tutorial

Week 10 Checkpoint: Feedback on tasks for Topics 6 - 8

By the this point you should have received feedback on the manually marked tasks up to and including 8.3D.

If you needed to REDO any of these tasks, you should look at the tutors feedback in the task submission and make any corrections.

If you have not submitted these tasks by now you will not receive feedback and risk failing those tasks.

Once your corrections are complete you MUST notify your tutor and ask that the task be re-marked.

□ Test 2 □ - in Week 10 labs

In Week 10 you will sit a timed test in the lab time.

Covering code and quiz tests from Week 1 to 6.

This test will be available in a lesson (NOT in these portfolio tasks).

Your tutor will mark your test and give you feedback here on whether you have:

1) Passed

2) Need to RE-SIT the test in another sitting.

10.1P Fix it - Sort Records

In this task you will build on the debugging skills developed in your other Pass and Tutorial tasks.

TIP: There are 3 errors in the code two are typing (syntax) errors, one is a logic error.

Your program should produce the following output:

```
[user@sahara ~]$ ruby fix_it.rb
first line: 6
Line read: Fred
Line read: Jill
Line read: Johnny
Line read: Sam
Line read: Jenny
Line read: Bill
Printing list: number of elements: 6
Name: Bill
 Age: 17
Name: Sam
 Age: 18
Name: Jill
 Age: 19
Name: Fred
 Age: 20
Name: Johnny
 Age: 22
Name: Jenny
 Age: 23
[user@sahara ~]$
```

10.2P 'Hello World' in C

In this task you will compile and run a “Hello World” terminal program in C.

- List the files in this directory using the ls command
- Print the working directory using the **pwd** command
- Compile your program using **gcc hello_world.c**
- List the files in this directory using the **ls** command to see the files created by the compile process
- Run your program using **./a.out**

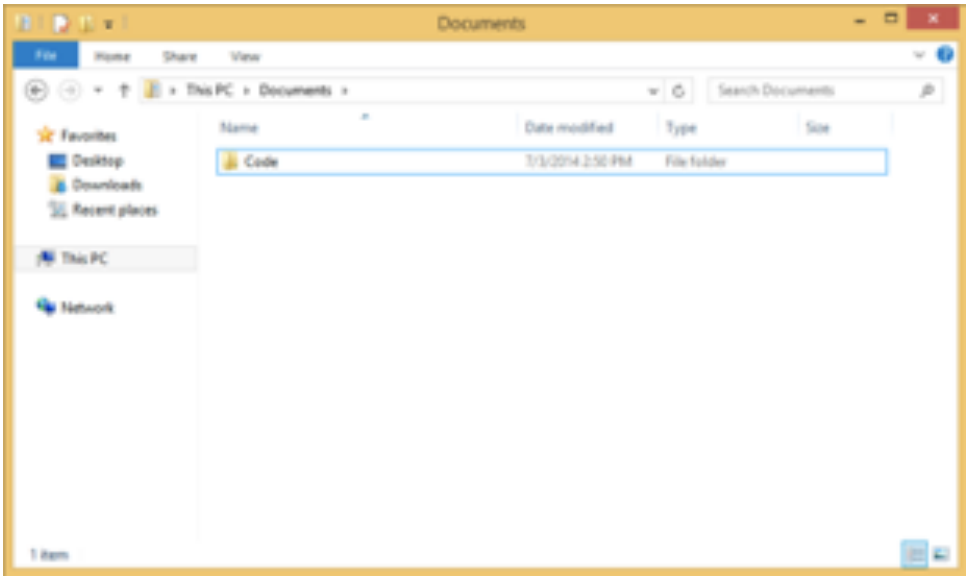
> **Note**: When you want to run a program in Terminal use `./` before the name of the program (no spaces). The . refers to the current directory. So `./` means look for the program in the current directory.

Use the sample code provided in the Resources for this task and modify that code to meet the requirements as described above.

DO THE FOLLOWING ONLY IF RUNNING ON YOUR OWN MACHINE

To compile your program you will need to use **mingw (**in Windows) or the **Terminal** (Mac OS):

1. If you don’t already have one, make a directory (i.e., a ‘folder’) to store your code (e.g., *Documents/Code/Lab1*). On a Swinburne computer you may wish to use a directory on your student drive or a USB storage device.
    1. • Navigate to your *Documents* directory in Finder or File Explorer
    2. • Right click in the *Documents* directory and select **New Folder**, name it **Code**
    3.



2. Open a **Terminal** (MSYS or MinGW shell in Windows), then perform the following commands:

- Change into the directory containing your code using the **cd** command.

```
        cd /c/Users/your_user/Documents/Code on Windows or

        cd ~/Documents/Code on MacOS or Linux
```

10.3P Hello User in Python

Write and run a program to read in from the terminal the user's name and age using Python.

Your program should then print out a hello message and the user's age (see the example below).

Hint: the code looks exactly the same as if you were printing a string in Ruby without a newline.

■ Compile and run your Python program:

Run and compile your python program by typing: `python3 hello_world.py`

Reference: Speight, A 2020 *Bite Sized Python*. Wiley. Available online from Swinburne Library.

See also: https://www.python.org/dev/peps/pep-0008/#prescriptive-naming-conventions

Use the following example code to guide you:

```
num = input('Number: ')
print(f'Number is {num}')
```

Your program should work as follows:

```
Enter your name: Fred
Hello, Fred!
Enter your age: 23
You are 23 years old
```

Week 11 Checkpoint: Feedback on tasks for Topics 9 - 10

By the end of Week 11 you should have received feedback on at least your first submission of the manually marked tasks up to and including 10.1P **and especially your major tasks** (the Music Player at whatever levels you are aiming for)

If you needed to REDO or FIX and RESUBMIT any of these tasks, you should look at the tutors feedback in the task submission and make any corrections.

If you have not submitted these tasks by now you will not get any feedback and risk failing the task.

Once your corrections are complete you MUST notify your tutor and ask that the task be re-marked.

NB: the remaining manually marked tasks (for Week 11 and Week 12) will be assessed when portfolios are assessed in the exam period. But you can seek feedback from tutors in tutorials and help desk sessions.

✔ Week 11 Attendance and Check-in

Do this in the tutorial this week. **You need to be at the tutorial in Week 11 to have this task marked complete.**

Answer the survey below, click the submit button in the workspace then make sure you ask the tutor to mark your submission complete in the tutorial (you must be at the tutorial for the tutor to mark it complete).

Select the best answer for your situation:

- ○ I am on track to pass
- ○ I am little behind
- ○ I am a lot behind
- ○ I am in trouble!

1 vote

11.1T M □ □ Learning Summary

The Learning Summary Report is your chance to outline how the work you have completed demonstrates that you have met all of the unit's learning outcomes. In this document you will indicate the grade you are applying for, and provide reasons why you should be awarded this grade based on the unit's assessment criteria.

COS00002-COS10009-COS60006 Portfolio Process and Assessment Criteria.pdf

This task will be assessed when we mark your portfolio after week 12.

Complete the following form and upload as a pdf file.

LearningSummaryReportTemplate.docx

11.2P Python Program - Silly Name Tester

In this task you will complete a simple program using the Python programming language.

Modify the provided code in `silly_name.py` to produce the following output:

```
192-168-1-101:PythonCode mmitchell$ python3 silly-name.py
What is your name? jock
jock is a
silly silly silly silly silly silly silly silly silly silly silly silly silly silly silly silly silly silly sil
ly silly silly silly silly silly silly silly silly silly silly silly silly silly silly silly silly silly silly
silly silly silly silly silly silly silly silly silly silly silly silly silly silly silly silly silly silly sil
ly silly name!
192-168-1-101:PythonCode mmitchell$
```

The pseudocode for this is as follows:

```
Make i equal 0
Print (staying on the same line) name, ' is a'
While i is less than 60
do
  Print (on the same line) 'silly '
  Increment i (make i equal i + 1)
end
Output 'name!' (moving to a new line)
```

There is also a Ruby version of the code (in the file `silly_name.rb`) that you can look at to translate from.

Video Guide.

11.3D M □ □ Python Shape Moving

You must enhance the code provided as follows:

1.The shape also can be moved up and down

2.The shape does not move out of the window area.

> You will need to install python3 and pygame on your machine for this task. See below taken from: https://www.pygame.org/wiki/GettingStarted

Pygame Installation

Pygame requires Python; if you don't already have it, you can download it from python.org. **Use python 3.7.7** or greater, because it is much friendlier to newbies, and additionally runs faster.

The best way to install pygame is with the pip tool (which is what python uses to install packages). Note, this comes with python in recent versions. We use the --user flag to tell it to install into the home directory, rather than globally.

```
python3 -m pip install -U pygame --user
```

(or use `pip3 install pygame`)

To see if it works, run one of the included examples:

```
python3 -m pygame.examples.aliens
```

If it works, you are ready to go! If not there are more detailed, platform-specific instructions here.

11.4HD C Name Tester

In this task you will redo the Silly Name program from 11.2 and write it in the C programming language.

You program should:

■Have a *main()* which prompts the user for a name

■But **you need to move some code out of `main()` into a procedure (void function)** that checks the name and prints the appropriate message.

The pseudocode for this procedure follows:

```
Procedure: print_silly_name
--------------------------------
Parameter:
 name (the silly name String)
Local Variables:
 i (an integer, used to count the number of loops)
--------------------------------
Steps:
 1: Make i equal 0
 2: Print (staying on the same line) name, ' is a'
 3: While i is less than 60
 4:     Print (on the same line) ' silly'
 5:     Increment i (make i equal i + 1)
 6: Output ' name!' (moving to a new line)
```

Compile your program using the following Terminal command:

```
$ gcc name_tester.c terminal_user_input.c
```

run your code by typing: `./a.out`

You can compile and run the code as is, then make your changes and check it still works.

Now `main()` should be CHANGED TO LOOK AS FOLLOWS:

```c
int main()
{
  my_string name;

  name = read_string("What is your name? ");

  printf("\nYour name");

  if (strcmp(name.str, "YOUR_TUTOR_NAME") == 0)
  {
    printf(" is an AWESOME name!");
  } else {
    print_silly_name(name)
  }

  return 0;
}
```

Your output should look as follows:

```
MacBook-Pro-5:Sample Code - C Name Tester mmitchell$ gcc name_tester_answer.c terminal_user_input.c
MacBook-Pro-5:Sample Code - C Name Tester mmitchell$ ./a.out
What is your name? YOUR_TUTOR_NAME

Your name is an AWESOME name!
MacBook-Pro-5:Sample Code - C Name Tester mmitchell$ ./a.out
What is your name? Fred

Your name Fred is a  silly  silly  silly  silly  silly  silly  silly  silly  silly  silly  silly  sill
y  silly  silly  silly  silly  silly  silly  silly  silly  silly  silly  silly  silly  silly  silly  s
illy  silly  silly  silly  silly  silly  silly  silly  silly  silly  silly  silly  silly  silly  silly
  silly  silly  silly  silly  silly  silly  silly  silly  silly  silly  silly  silly  silly  silly  si
lly  silly  silly  silly  silly name!
MacBook-Pro-5:Sample Code - C Name Tester mmitchell$
```

Submit your code and a screen shot of it running to Doubtfire.

12.1C M □ □ Minitest Test Harness

Using the Ruby `minitest` gem, write and run tests for the code provided for this task.

Complete the tests required (as indicted by the comments) in `tester_demo.rb.`

Submit your code and a screenshot - **NOTE: can be completed after WEEK 12 (i.e does not need to be marked complete)**

□ Portfolio Completion Quiz

Question

Select which of the following are true in relation to completing the requirements for different grade levels:

- [ ] All pass and tutorial tasks must be completed along with both tests, in order for a student to pass the unit.
- [ ] All the tasks for higher levels must be completed to achieve that level.
- [ ] For a distinction above 70 or high distinction a custom program is required.
- [ ] It is ok just to attempt tasks, they do not actually need to be marked complete.
- [ ] Learning Summary Submitted in Week 13

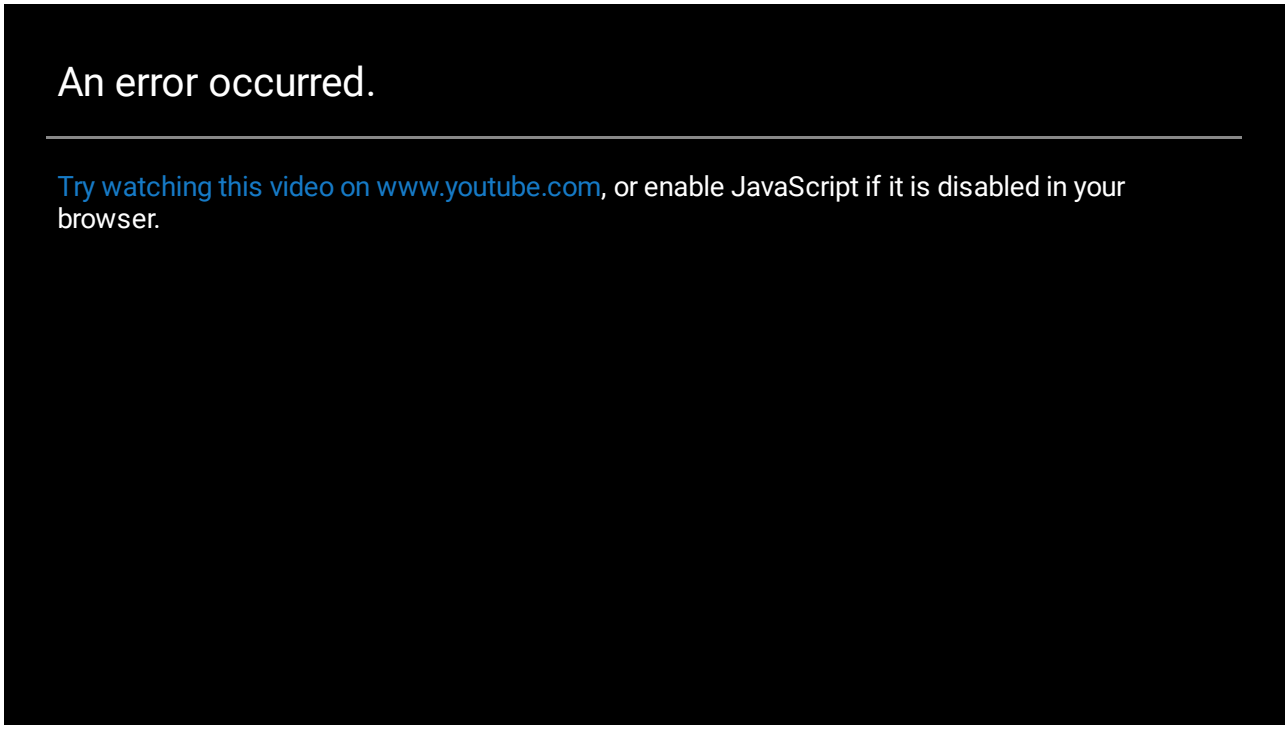
Checkpoint: All tasks submitted

Well done! If you have completed all the tasks, then the following will be looked at when your portfolio is marked:

- 11.2 Learning Summary
- Custom Program (if you are aiming for a HD)
- Video link for Custom Program.
- Custom Project (if you are aiming for HD band 2)

This week you also need to have Task 11.3 marked complete if you are aiming for a HD.

Demo Custom Program - Breakout by Dylan Glenister



A demonstration custom program from a previous semester.

Custom Code □ □ □

Note: You have a choice to do the Distinction level Extended Music Player or another Custom Program - approved by your tutor. This explains the Custom Program option.

You are now close to completing tasks related to all of the unit learning outcomes, and can work toward demonstrating these in your own program. If you are aiming for a Distinction or higher grade you should start working on this program now. Aim to create something of at least the complexity of the original Food Hunter program for the lower distinction grade or more complex for higher grades. Specifically it should:

MarkingRubricSummary(4)-1.pdf

1. Demonstrate the use of functional decomposition - implement the program with a number of functions and procedures. (Maybe even modular decomposition with separate units if you can identify some reusable artefacts - optional but nice)
2. Demonstrate the use of arrays and records
3. Demonstrate the use of structured programming (sequence, selection, and repetition)
4. Demonstrate appropriate use coding conventions - case, indentation
5. It must not use global variables, or goto.
6. Make sure you can explain your code in an interview!
7. Use the checklist on the next page to make sure you have everything you need to submit!

Note: Your program should be different from the food hunter program and the lecture demonstration programs. You want to demonstrate that you have learnt from these tasks and can apply what you have learnt to some other program design.

If you are aiming for a High Distinction, review the related High Distinction Project document and check the marking rubrics for details on how you can ensure this program meets the HD requirements.

Here are some steps to get you started:

1. Think about what you want the program to do. Maybe write up a paragraph or two to explain it to others. Drawing a picture of what you want it to look like is also a great idea.
2. Show your plans to your tutor, lecturer, help desk staffers, and/or friends to get some feedback.
3. Start thinking about the data - what records and enumerations will you need?

Tip: Start small, you can easily add to records at a later stage. Try to identify what records you will need, then add just the basic data - enough to get something working. Once that first part is working, add additional fields as they are needed.

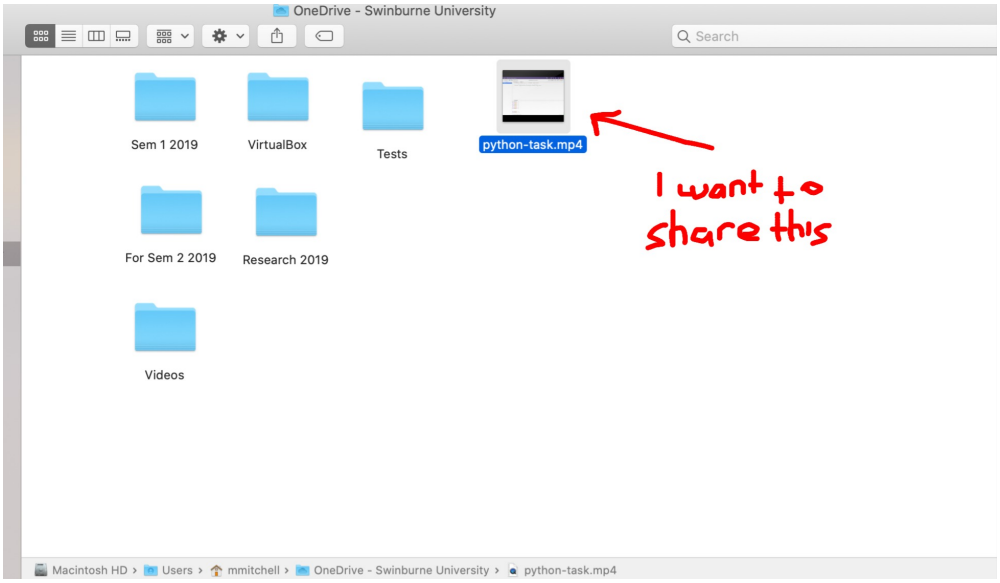
Submit your code and a screenshot of it to the workspace.

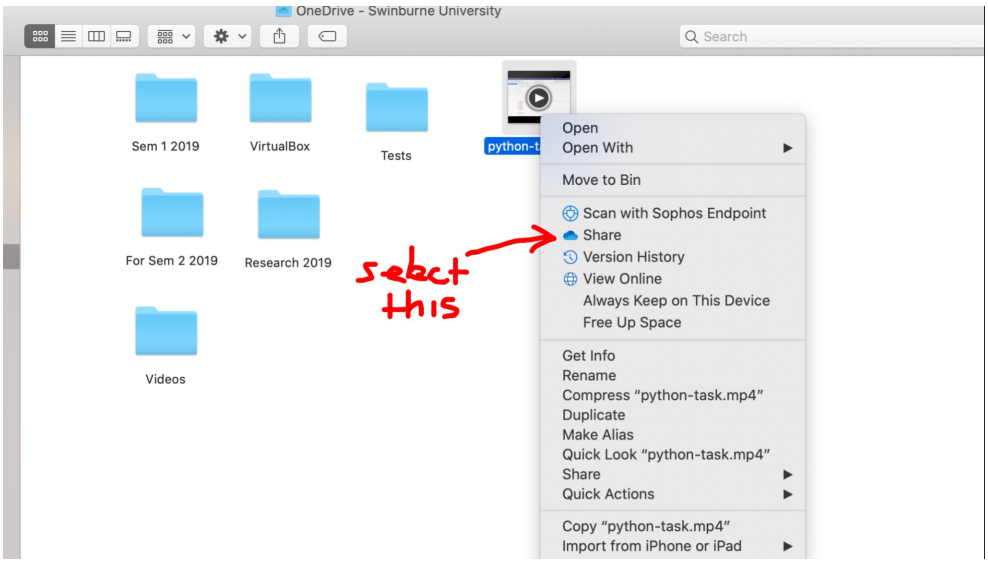
Custom Code Video □□ □

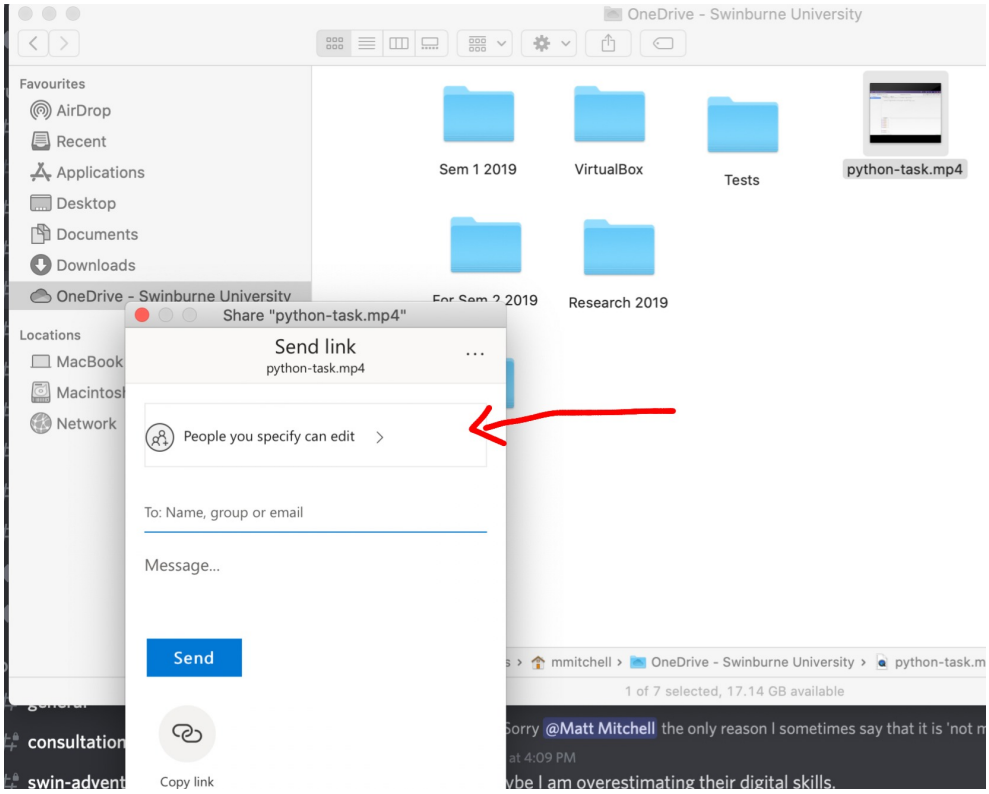
Just upload here a document with a link to your custom code video.

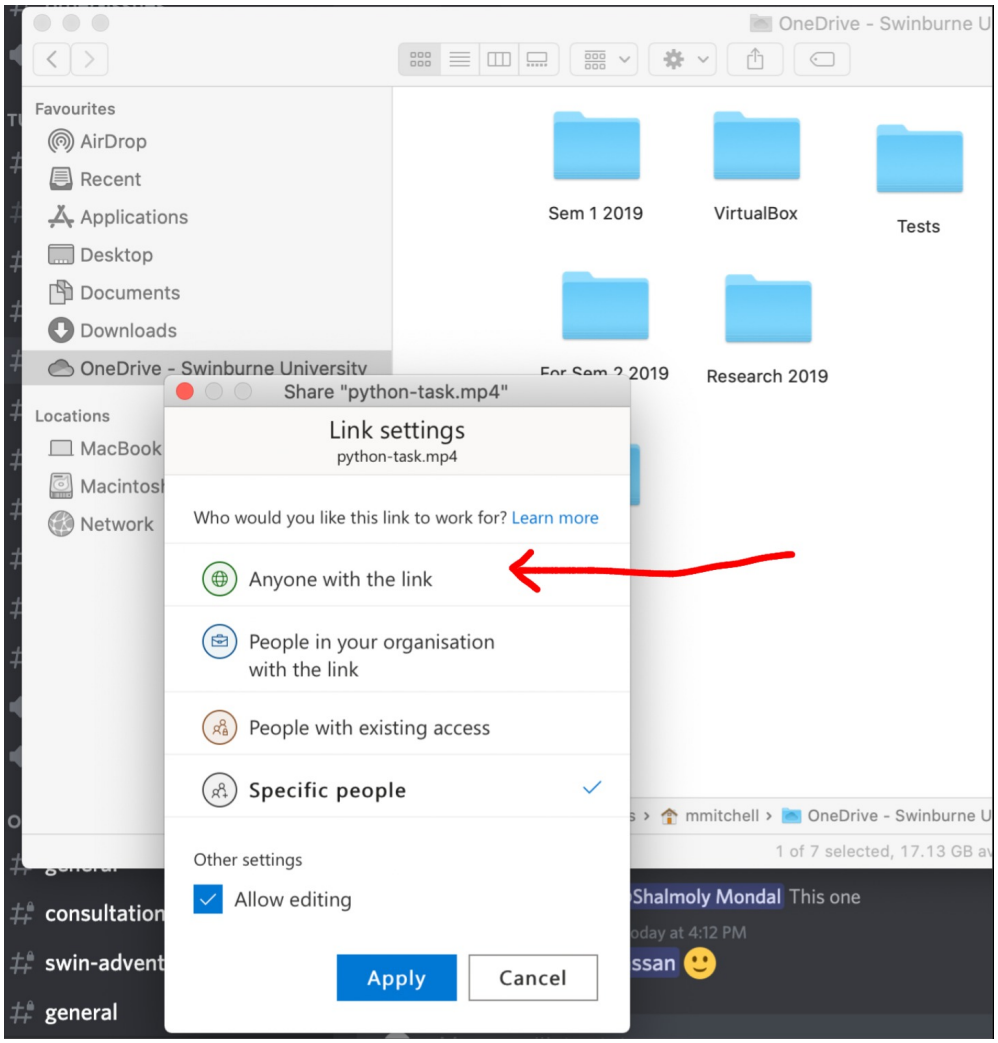
You can do it using One Drive if you want:

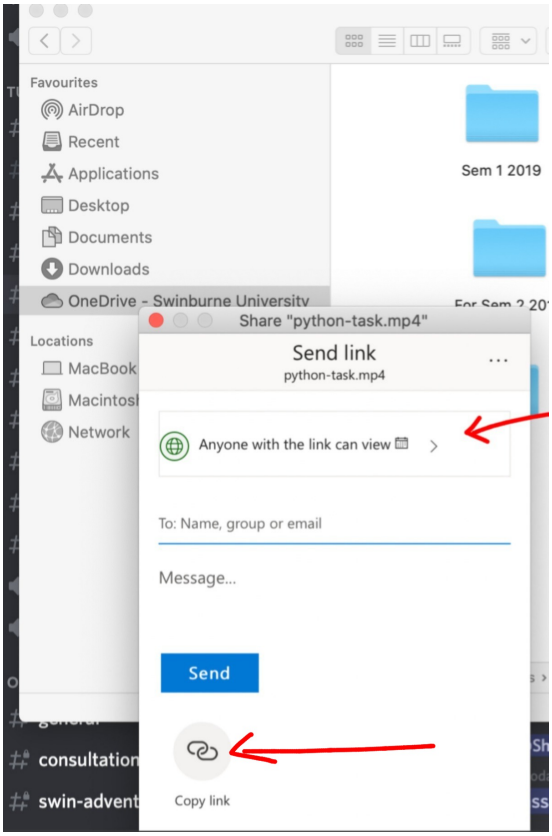
Copy your video to Swinburne OneDrive then:

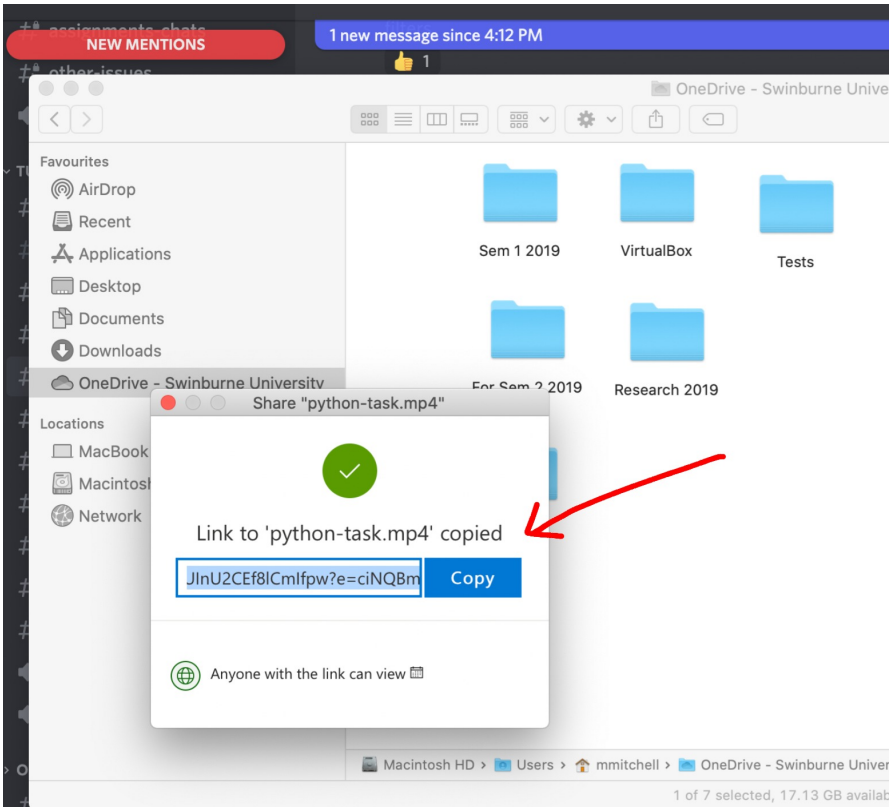





OneDrive - Swinburne University
Favourites
AirDrop
Recent
Applications
Desktop
Documents
Downloads
OneDrive - Swinburne University
Locations
MacBook
Network
Sem 1 2019
VirtualBox
Tests
python-task.mp4
Research 2019
Share "python-task.mp4"
Send link
python-task.mp4
People you specify can edit
To: Name, group or email
Message...
Send
Copy link
1 of 7 selected, 17.14 GB available
consultation
swin-advent


OneDrive - Swinburne U
Favourites
AirDrop
Recent
Applications
Desktop
Documents
Downloads
OneDrive - Swinburne University
Locations
MacBook
Network
Sem 1 2019
VirtualBox
Tests
Research 2019
Share "python-task.mp4"
Link settings
python-task.mp4
Who would you like this link to work for? Learn more
Anyone with the link
People in your organisation with the link
People with existing access
Specific people
Other settings
Allow editing
Apply
Cancel
1 of 7 selected, 17.13 GB
consultation
swin-advent
general


Favourites
AirDrop
Recent
Applications
Desktop
Documents
Downloads
OneDrive - Swinburne University
Locations
MacBook
Macintosh
Network
Sem 1 2019
Share "python-task.mp4"
Send link
python-task.mp4
Anyone with the link can view
To: Name, group or email
Message...
Send
Copy link
consultation
swin-advent


NEW MENTIONS
1 new message since 4:12 PM
1
OneDrive - Swinburne Unive
Favourites
AirDrop
Recent
Applications
Desktop
Documents
Downloads
OneDrive - Swinburne University
Locations
MacBook
Macintosh
Network
Sem 1 2019
VirtualBox
Tests
Research 2019
Share "python-task.mp4"
Link to 'python-task.mp4' copied
JInU2CEf8lCmlfpw?e=ciNQBm
Copy
Anyone with the link can view
Macintosh HD › Users › mmitchell › OneDrive - Swinburne Univer
1 of 7 selected, 17.13 GB availab

Custom Project □ □

The aim of this task is to demonstrate significant depth of understanding related to the unit's topics and concepts. Several ideas are presented, but you are free to do this in any way you see fit.

MarkingRubricSummary(4)-1.pdf

This task works in conjuncture with the HD standards on the Custom Program. With the custom program you are demonstrating that you can apply the concepts learnt, whereas in this task you are demonstrating that you can explain and discuss the concepts with similar depth of understanding.

There are a number of things you can do to help demonstrate your ability to explain and discuss programming concepts at depth. The following types of suggestions indicate appropriate projects in ascending order of grade level for this task:

- **Basic (90 - 93):** Provide a tutorial on the use of Gosu, Tcl/Tk or other Ruby gems to accomplish a task.
    - This could be a walkthrough to write a game, or use some more advanced features (net-working, web, sprites, physics, etc).
    - For higher grade levels you should relate use of the libraries/gems to coding with good program design and abstraction.
    - You must have primary source references.
- **Intermediate (93-95):** Explain a concept. The concept must be challenging, and you should do BOTH the following:
    - Write a research report on the concept explaining it and drawing on the literature.
    - Suitable concepts could be programming pattern or design principle.
    - Produce a video. Explain your concept. Maybe show example code to help with this.
    - You must have primary source references.
- **Advanced (95+):** Conduct a small research project aiming to answer a question related to programming or program design, or propose a new design pattern or practice.
    - Create a plan to outline the question and method for your research project. The **research question** is the question you aim to investigate in the project. The research method describes how you will approach answering the question.
    - Collect evidence to address your research question.
    - You must have primary source references.

Submit your project files to the workspace.

Booking the D and HD Interviews

If you are aiming for a mark greater than 70 D, then you need to complete and submit a custom program.

Interviews of the custom program are held over the exam period - watch the unit announcements for instructions on how to book.

Quiz