GOVERNMENT OF WESTERN AUSTRALIA

TAFE WA South Metropolitan

Pseudocode

(Session 12)

Overview

- Programming Concepts
- Variable Rules
- Calculations and Selections
- Pseudocode Rules

Some Programming Concepts

- Design
  - Often people like to *jump right* in and start tinkering, and this is fine, it shows you're curious and interested.
  - However, this "fail fast" method that allows you to try, fail, fix, and try again constantly doesn't work well once you get past simple programs.
  - Designing what you want to happen and the steps involved before you start coding is an important step in the process.
  - To do this, a common tool we use is **pseudocode**.

Some Programming Concepts

- Pseudocode
  - A way of reflecting structured programming
  - Can be difficult to follow if you are new to it
  - Acts as a high-level blueprint or description of an algorithm or program
  - Used to plan out a program prior to writing it, this is especially helpful when needing to create large and complex programs.

Pseudocode Rules

1. Like regular code – **write one statement per line**
2. Use CAPITALISATION for keywords
3. Indentation
   - This shows hierarchy and will get you used to programming
4. End multiline structures
5. Keep statements language independent
   - This is not a program, its **plain English**

Variables

- Used to store information to be referenced and used by programs
- Provide a means of labelling data with a descriptive name
- Think back to Rule 2, this is important for variables.
- Also have their own unique set of rules

Variable Rules

- Be consistent with how you name your variables:
  - **camelCase**: first word starts lowercase and the rest start with an uppercase
  - **PascalCase**: All words start with an uppercase letter.
  - **snake_case**: all lowercase with an underscore between each word.
- Cannot start with a number
- Contain no spaces
- Unique names within your code and no use of keywords
- Consistent use of names
  - E.G. myName, myAge, myGender, myAddress

Rule One: One Statement Per Line

- One of the hardest things about pseudocode is starting.
- A good process to use is to think about what your code needs to do in order, in a general sense.
- Then, write a task list (simple dot points) of the major things that should happen from the start to the end.
- This can then serve as a structure to start your pseudocode.

How to Show Data Access

- In pseudocode, we still need to show the access and transition of data. This can be achieved with the READ statement
- READ depicts us telling the computer to get a value from an input device and store it in a memory location.

Fields: Calculation and Selection

Calculations

+ Add

- Subtract

* Multiply
/ Divide
** or ^ Exponential
( ) grouping

Selection

> Greater than

< Less than

= Equal to

>= Greater than or equal to

<= Less than or equal to

<> Not equal to

Rule One: One Statement Per Line

- Express just one action for the computer on each line.
- An organised task list will correspond to a line of code

Task List

- **Read item name, item price, item sale amount**
- **Perform calculations**
  - **item gross = item price X item sale amount**
- **Write item name, item gross**

Pseudocode

```
READ item_name, item_price, item_sales

item_gross = item_price * item_sales

WRITE item_name, item_gross
```

Rule Two: Use CAPITALISATION for keywords

- Within Pseudocode you should remember to CAPITALISE your key words
  - Key words are reserved words where the system will perform some form of action
  - This enables the reader to identify the functions being used
- Some keyword examples include READ, WRITE, AND, OR, IF, ELSE IF, ELSE, ENDIF, WHILE, ENDWHLE, etc.
- Where necessary also use camelCasing/PascalCasing/snake_casing to identify the name of a variable.

Rule Two: Use CAPITALISATION for keywords

The following example provides evidence of CAPITALISATION and camelCasing

Pseudocode

READ name, hours_worked, pay_rate

gross = hours_worked * pay_rate

WRITE name, hours_worked, gross

Rule Three: Indentation

- Each design structure uses a particular indentation pattern
- Indentation will make your pseudocode easier to read
- It also prepares you for programming with Python
  - There's a lot of indentation in Python ☺

Rule Three: Indentation

The rules of indentation include:

- Sequence:

  Keep statements in sequence all starting in the same column

- Selection:
  - Indent statements that fall inside selection structure, but not the keywords that form the selection
- Loop:
  - Indent statements that fall inside the loop but not keywords that form the loop

Rule Three: Indentation

Example Indentation:

```
READ name, gross_pay, taxes
IF taxes > 0
    net = gross_pay – taxes
ELSE
    net = gross_pay
ENDIF
WRITE name, net
```

Rule Three: Indentation

Example Indentation and ending of multiline statements:

```
SET temp_rate to 0
WHILE temp_rate < 30
        temp_rate +1
        IF temp_rate < 10
                WRITE “It’s a bit chilly”
        ELSE
                WRITE “This is nice”
        ENDIF
ENDWHILE
WRITE “It’s toasty”
```

Rule Four: End Multiline Structures

- When we look back to our Indentation example, you can see that the multiline structure is started with IF, provides varying results based on the condition with ELSE and ends with ENDIF.
- This allows the reader to see what subprocesses happen in the structure.
- By showing where they end, it allows for better clarity.
- This should be placed in line with the key word that started the subprocess.
- The same rule applies for WHILE and ENDWHILE, FOR and ENDFOR etc…

Rule Five: Language Independence

- As pseudocode is intended to be **plain English** you must resist the urge to write in the language you are studying or comfortable with.
- **The main aim of pseudocode is to create a blueprint of code which could then be translated using any language.**
- The purpose of this is to enable you to describe a logic plan to develop a program, you are not actually programming it in python (or any other language).
- Quite literally think “what is this line going to do?” and write it using the rules we have discussed.

Questions?