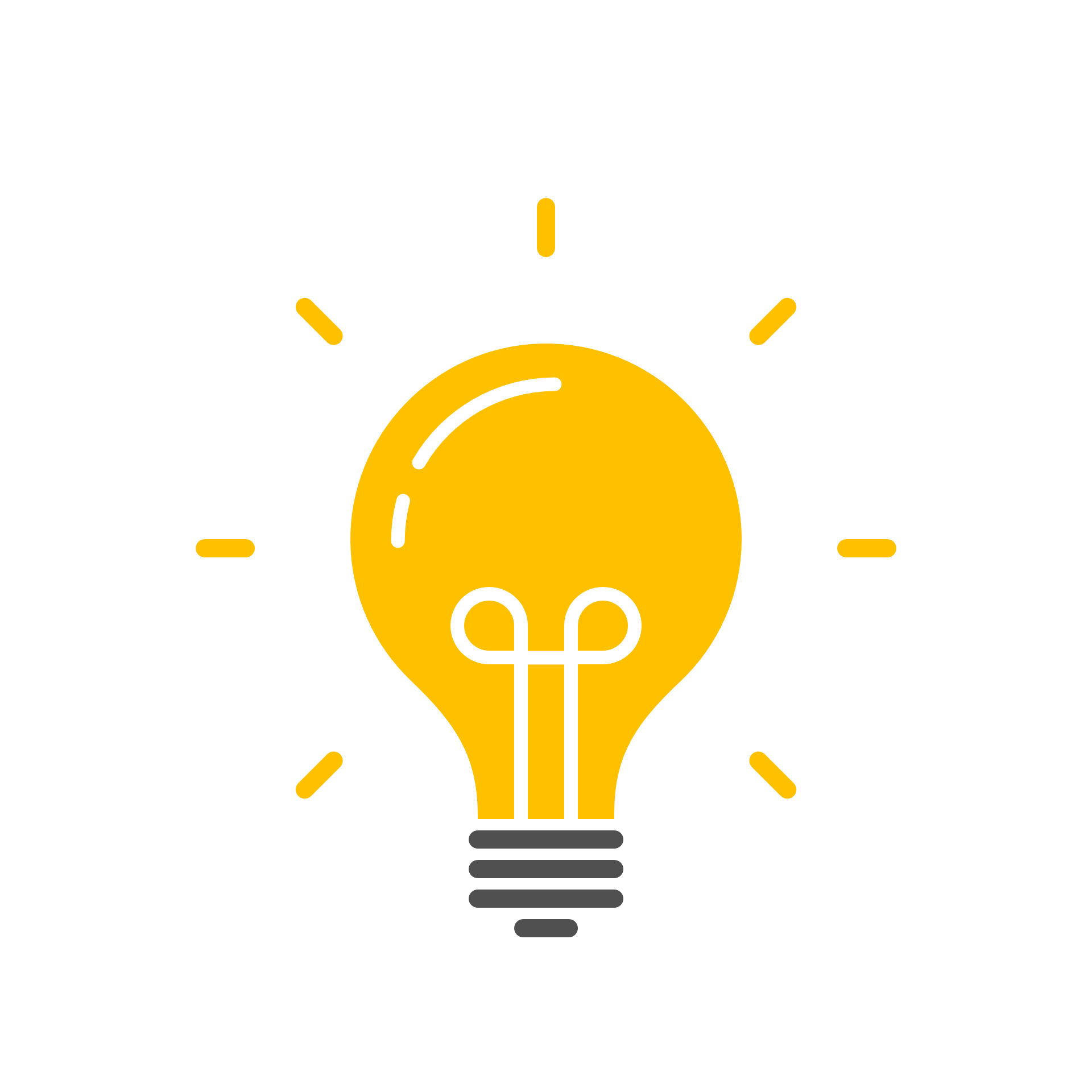
On this page:

* Create a “Mind map”, which is a brainstorm of all your ideas surrounding this task.
* Try and classify where possible your ideas in the mind-map into categories and sub-categories
* Stimulus analysis – use the "scrap book" genre and floating text boxes to point out anything interesting from the **task sheet** or **technical proposal** (stimulus) around your mind map.

Cognitive verbs evidenced here might include: **recognise**, **describe**, **explain**, **analyse**, **identify**, **determine, symbolise, communicate**.

**Contextual analysis**



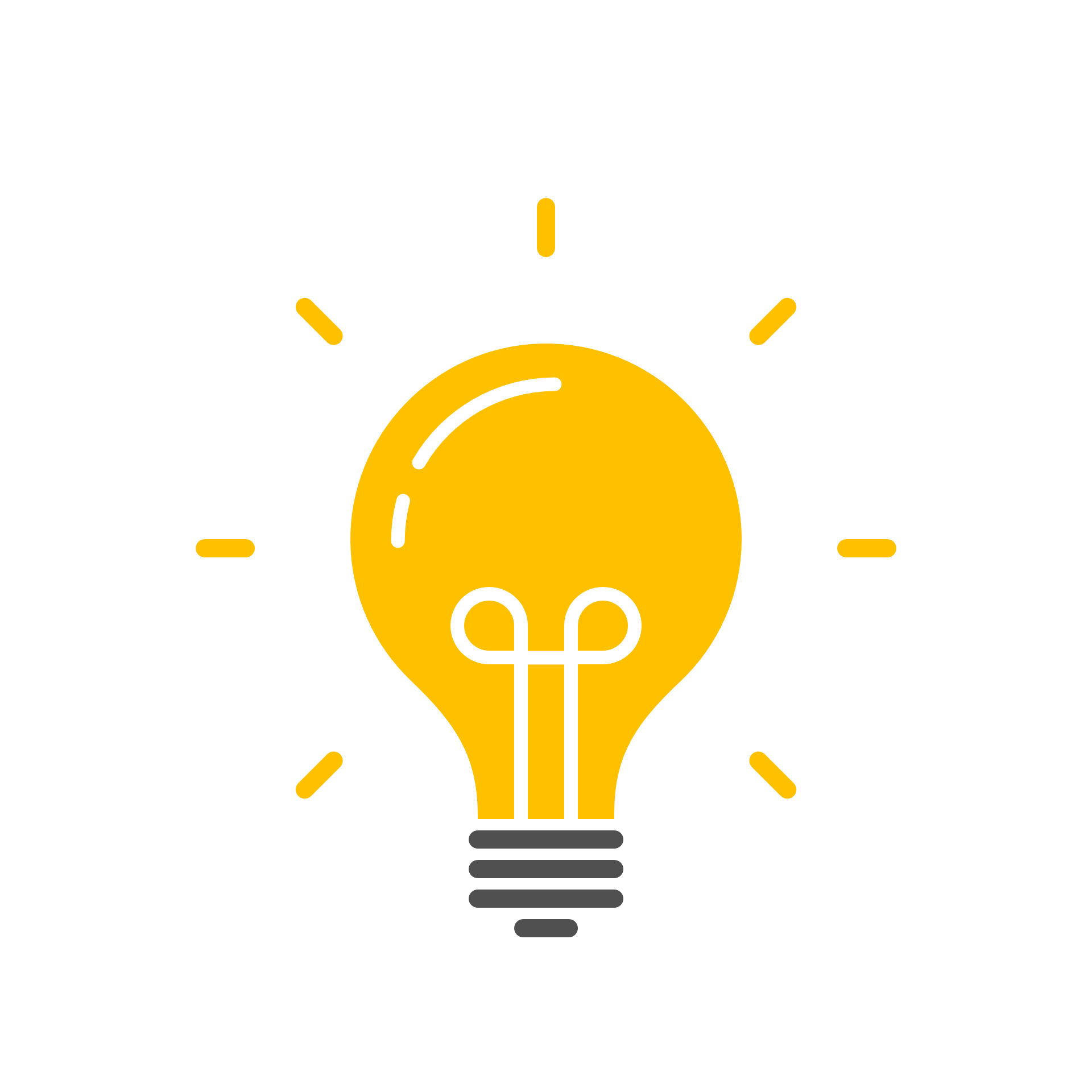
Mind Map – ideas for **categories** to use to break up your Digital Solution Web Application task might include:

* Task requirements
* Front end development
* Back end development
* Task analysis
* User analysis
* Contextual analysis
* Explore
* Develop
* Generate
* Evaluate and refine
* Problem analysis
* Features required
* Personal, social, ethical, or legal impacts
* Sustainability or economic impacts
* Prototype requirements
* Technical considerations
* Client-side development
* Server-side development
* Programming components
* Data structures
* Inter-relationships between UX and data
* Useability requirements – learnability, accessibility, utility, safety and effectiveness
* UX – principles of design or aesthetic
* Project methodology
* Risks, disclaimers, assumptions or limitations
* Milestones or benchmarks
* Scope and out of scope
* Submission requirements
* Core functionality
* Extended functionality
* Systems or success criteria

On this page:

* Determine a big list of “Prescribed Criteria”, which are **determined** from the requirements of the task sheet or technical proposal (stimulus).
* Determine a big list of “Self-determined Criteria”, which are **determined** from your own ability, research and experience building web applications.
* Draw a small tech stack diagram, that illustrates all the software, hardware and networking technologies you can think of that would be needed to power the live web application.
* If there is any analysis left over from the first page, feel free to continue that here.

**Prescribed and self-determined criteria**



Ideas you can frame your **self-determined criteria** around might include:

* **Reliability** – how will you determine whether the application is satisfactorily performing its intended task(s)
* **Accuracy** – look for logical threats to maintaining accurate records
* **Maintainability** – refers to the ease and speed at which a system can maintain operational status, especially after fault or repair.
* **Security** – such as delivering an application over a secure connection (HTTPS)
* **Sustainability** and **scalability** – what happens when the application “grows”?
* **Efficiency** – Efficiency seeks maximum productivity with the least / minimal consumption of system resources. Can you measure this somehow?
* **Modularity** – can your codebase be repurposed easily?
* **Standards** – such as meeting specifications for code design or system design
* **Optimisation**, either for navigation or searching
* Advanced application of *useability principles*, such as:
  + **Accessibility** – compatibility with assistive devices or semantic tags
  + **Safety** – error troubleshooting

Consider self-determined criteria as criteria that you (personally) know will make the application much better.

**You want to “show off” in your Self-determined criteria.**

**Criteria**: characteristics by which something is evaluated or appraised; the ~~teacher~~ **stimulus** or client (prescribed) or students (self-determined) develop criteria, e.g. specific needs, identified purpose, impacts, quality or effectiveness of solution.

Ideas you can frame your **prescribed criteria** around might include:

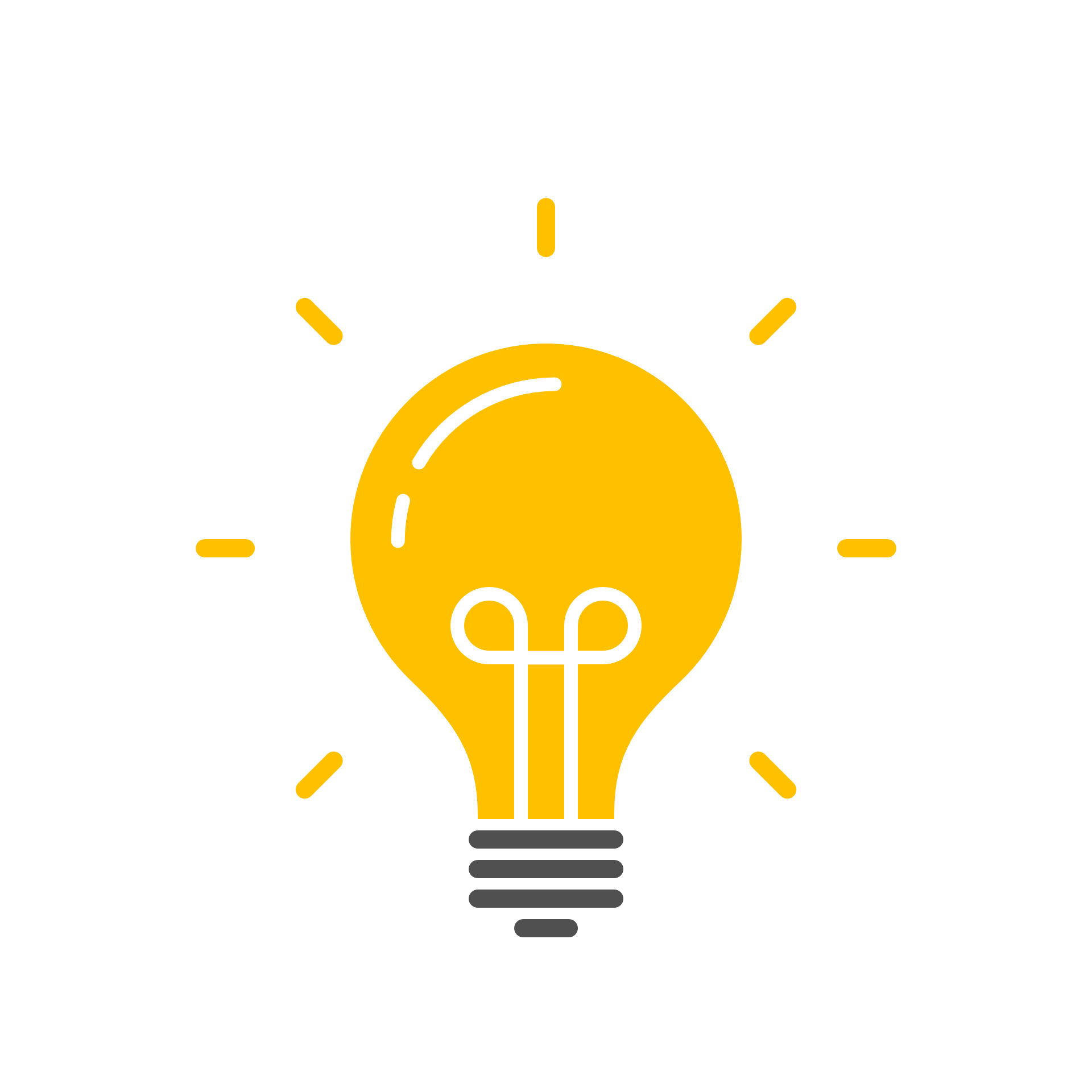
* Task requirements
* Documentation or video requirements
* Syllabus requirements (e.g. EDGE problem solving model)
* Base / basic / core aims of the application
* Non-negotiable useability principles (e.g. accessibility – meeting legal requirements)
* Minimum or mission critical utility requirements – what **must** the application be able to do?
* Simple measures of success – what are the primary aims of the solution?
* Standards or levels of acceptance – what is the lowest or baseline level of functionality that must be achieved?

Consider prescribed criteria as obvious criteria.

On this page:

* Synthesise your User Interface designs.
* Annotate them with useability principles.
* Justify elements and features of UI based on your previous analysis. Where possible, make links!

**User Interface**



Useability principles: principles used to improve the user experience, including:

**accessibility**: ability to be used by many different people, even people with disabilities

**effectiveness**: ability of users to use the system to do the work they need to do, includes reliability

**safety**: ability for users to make errors and recover from the mistake

**utility**: ability of the system to provide all the functionality that users need

**learnability**: how easy a system is to learn.

Source: Queensland Curriculum & Assessment Authority, "Digital Solutions 2019 v1.2 General Senior Syllabus". (2020). Retrieved 01/01/2021, from <https://www.qcaa.qld.edu.au/downloads/senior-qce/syllabuses/snr_digital_solutions_19_syll.pdf>

On this page:

* Continue to synthesise your User Interface designs, recognising and describing GUI components
* Keep annotating them with useability principles.
* Consider showing inter-relationship between UI / UX with algorithms / coded components. Where possible, make links!

User Interface

On this page:

* Brief amount of Cipher research
* Site map
* Variable table (a.k.a. “variable dictionary”)

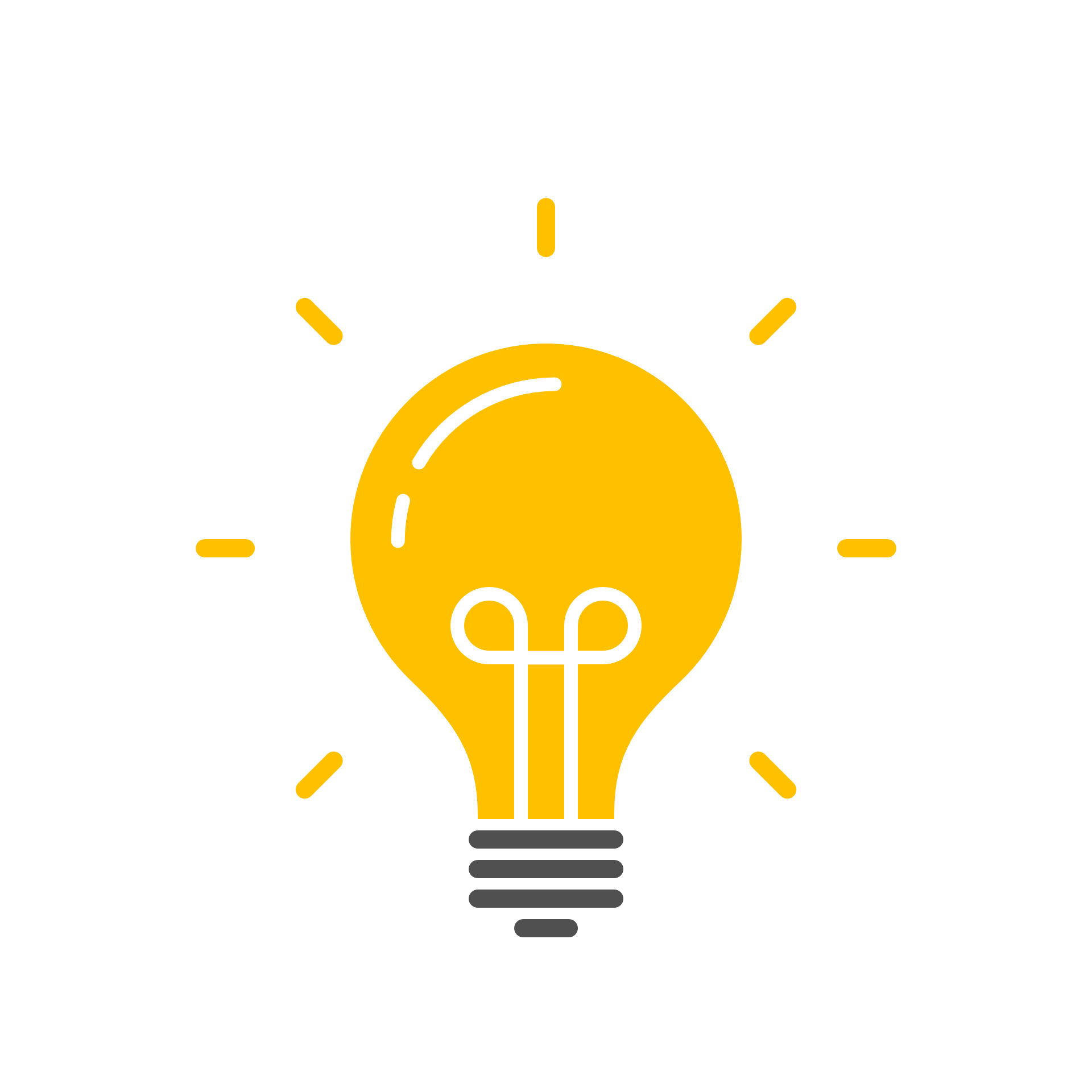
In a table of variables, specify all variables that will be required by:

* The cipher puzzles themselves
* The web application (e.g. username)

For each variable:

* Specify its data type
* Specify sample values that the variable will hold
* Specify the limits or range of the values that the variable will hold

Ciphers and variables

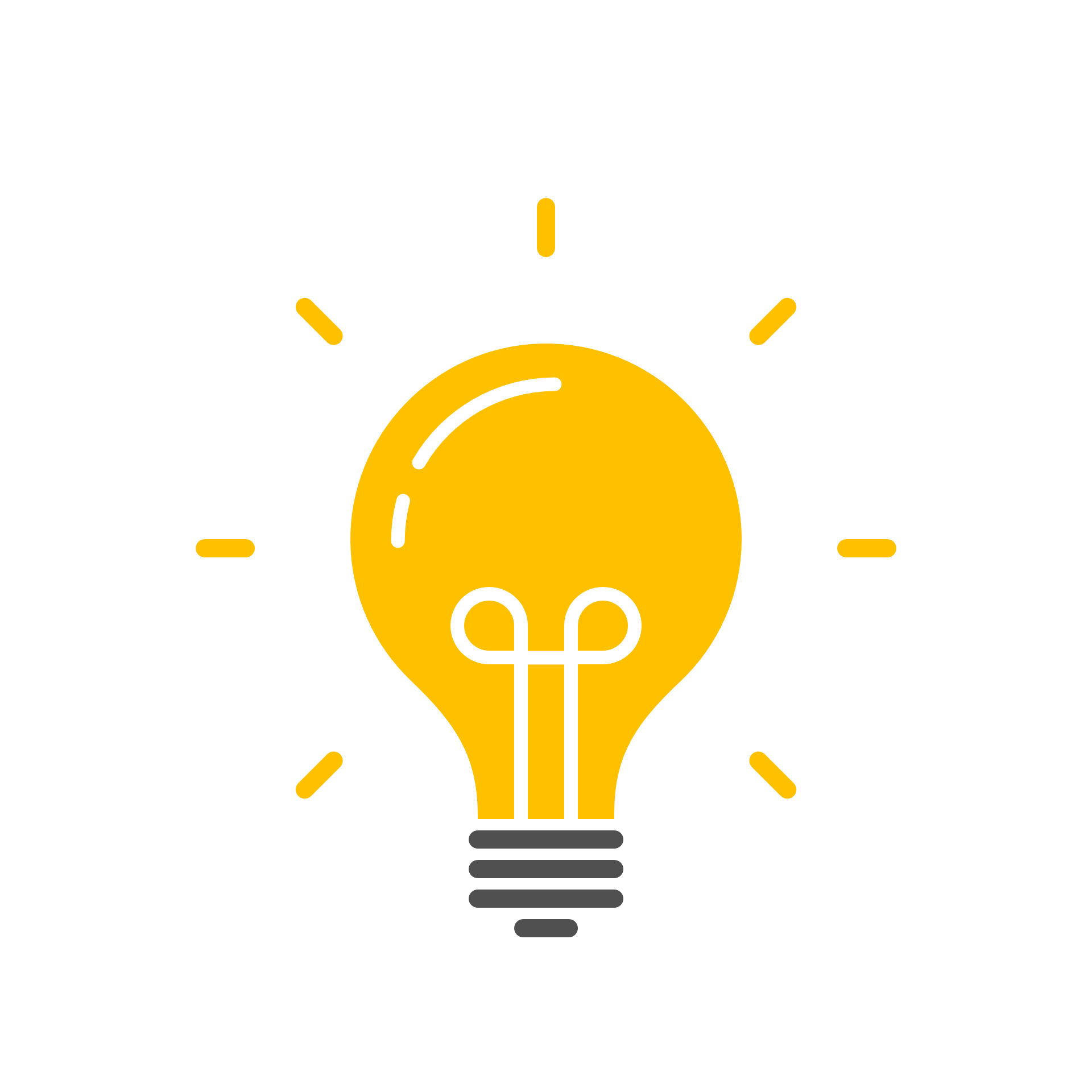


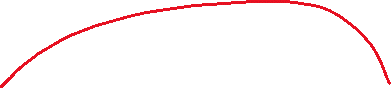
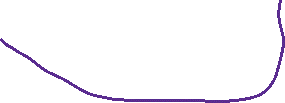
|  |  |  |
| --- | --- | --- |
| Variable Dictionary for Cipher Puzzles | | |
| Name | **Data Type** | **Sample usage, limits or notes** |
| Plain\_text | String | 15 characters maximum, sample value: "happy" |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

On this page:

* Write algorithms in pseudocode to power your site.
* Algorithms might include cipher algorithms, ranging in complexity (can take an extra page for this if necessary)
* Algorithms may also include web application algorithms for managing users, statistics, etc

Algorithms





The following information is a summary of this topic: <https://digisoln.com/computational_thought/pseudocode>

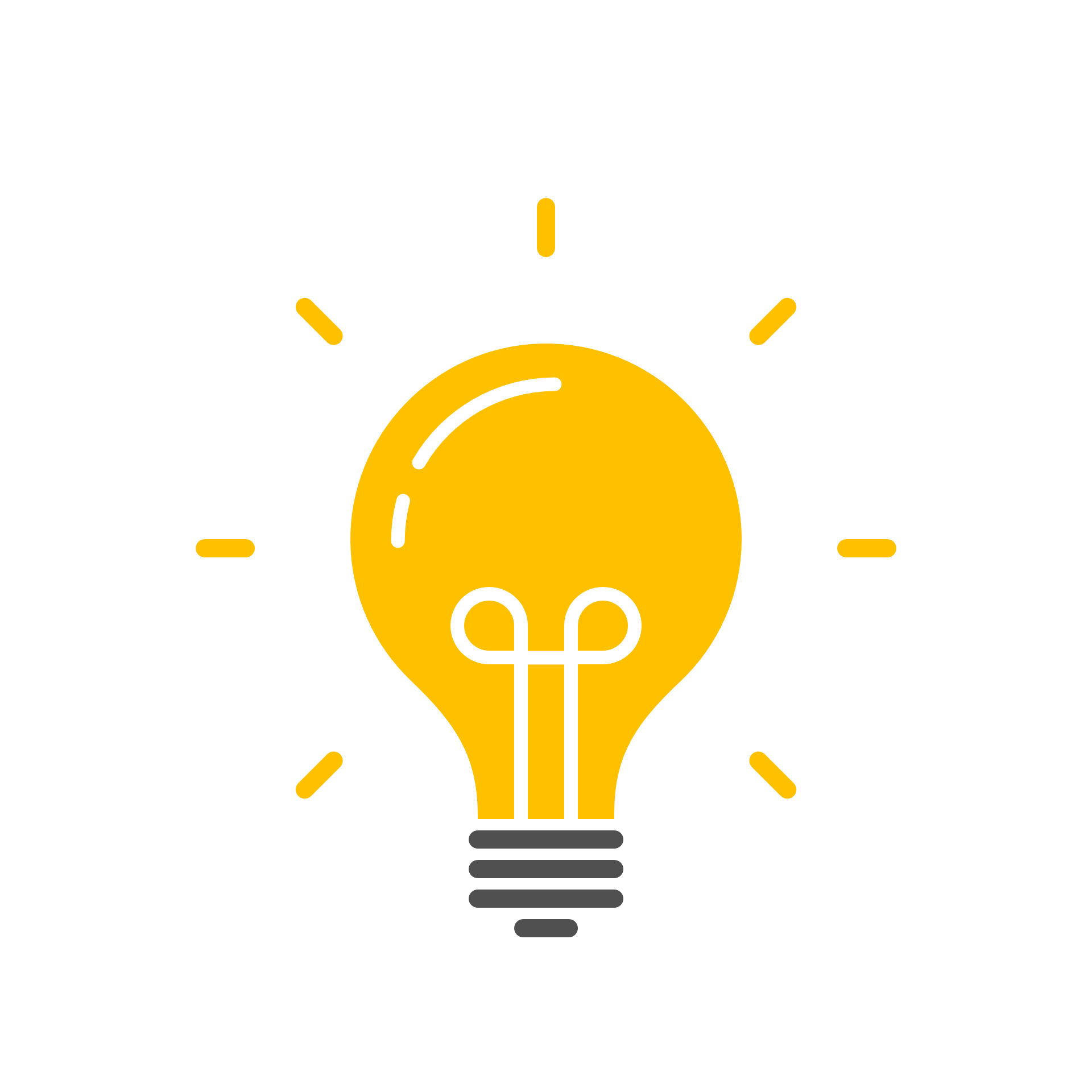
* Write **BEGIN** and **END** on every algorithm module.
  + Indent ...
  + ... consistently.
* When using **IF** remember to use **THEN** (an alternative branch is **ELSE**), and **ENDIF**
* **WHILE** remember to use **ENDWHILE**
* **IF** and **WHILE** only execute when **TRUE**
* **Advanced**: Modular algorithms can have parameters that can be swapped or passed in, E.g.: **BEGIN get\_area(radius)**
* **Advanced**: algorithms can have local or global variables:
  + Local variables can only be accessed in the current module (such as temp\_score)
  + Global variables can be accessed from all algorithm modules (such as privilege\_level)
* **Advanced**: Functions can return values to other function calls. E.g.: **area = get\_area(10)**
* If in doubt, just write Python code, and simplify it to ‘make’ it pseudocode.



On this page:

* Continue remaining algorithms in pseudocode to power your site.
* You may also like to use this space for screen shots of development, illustrating testing or problems, and refinements made

Algorithms



Sample algorithm written in pseudocode: **Cipher**

BEGIN Cipher

SET plain\_code TO ['Z','A','B']

SET cipher\_code TO empty array

FOR count = 0 TO length(plain\_code)

SET ordinal = ord( plain\_code[count] ) - 65

SET shift = (ordinal + 3) mod 26

SET cipher = shift + 65

APPEND chr(cipher) TO cipher\_code

NEXT count

PRINT cipher\_code

END Cipher

Code paste and annotate page 1.

* Avoid pasting any code you did not write. Borrowed code can be referenced in your reference list.
* Include both Python and Jinja2 code, and modularise your programming components (e.g. into application path routes for Python, or sections of content rendering for Jinja2).
* Annotate refinements made throughout: “evaluation of **impacts**, user experience (client-side) and coded components (server-side) and the digital solution against essential prescribed and self-determined criteria”

Code paste and annotate page 2 (if needed)

Code paste and annotate page 3 (if needed)

Code paste and annotate page 4 (if needed)

Code paste and annotate page 5 (if needed)

Code paste and annotate page 6 (if needed)

Any remaining code truncate at bottom of this page and refer to video

**FIA2 Evaluation – fill in the tables:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Prescribed Criteria** | **Evaluation of:**   * **impacts** (including Personal, Social, Economic, *Ethical, Legal and Sustainability impacts)* * **user experience** * **coded components** * **digital solution** | **Data for justification**   * **testing** * **qualitive / quantitive** * **screen shots??** | **Refinements made /**  **Recommendations for future** |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Self-Determined Criteria** | **Evaluation of:**   * **impacts** (including Personal, Social, Economic, *Ethical, Legal and Sustainability impacts)* * **user experience** * **coded components** * **digital solution** | **Data for justification** | **Refinements made /**  **Recommendations for future** |
|  |  |  |  |
| Web Application needs to be minimalistic, appealing and practical | **user experience:**  The web application was not appealing. It lacked a consistent or professional layout and colour scheme.  **Economic impacts / digital solution:** It was minimal because it had barely anything on it. This meant the website loaded really quickly, so users didn’t waste their time waiting. |  | **Refinement** |

**Reference List**

Queensland Curriculum & Assessment Authority, "Digital Solutions 2019 v1.2 General Senior Syllabus". (2020).   
Retrieved 01/01/2021, from <https://www.qcaa.qld.edu.au/downloads/senior-qce/syllabuses/snr_digital_solutions_19_syll.pdf>