[CS1] Intro Course Requirements

UD CISC - Version 0.0.4

*This document is in-progress and not official, it has not been approved by any departmental governing body!*

This document attempts to summarize the core course requirements of any UD CISC class that seeks to satisfy the “CS1” prerequisite of CISC181 and CISC210. Specifically, the document provides Learning Objectives that specify the necessary skills and abilities of learners who meet the requirements of passing CS1.

The goal of this document is to guide instructors building a CS1 course at UD, designing assessments/placement exams, and faculty deciding whether a given course or students’ prior experience should fulfill the CS1 requirement.

**Table of Contents**

[1. Structure](#_rlfw0fegz3k8)

[1.1. Naming Things](#_lzz6pf291jj0)

[1.2. Learning Objective Structure](#_l0eh88jht3yb)

[1.3. Assessment Requirements](#_px25gagaavcr)

[1.4. External Standards](#_98vbf4lhyl6s)

[1.5. Example Assessment Questions](#_lp48ngsa7whg)

[1.6. Performance Objectives](#_w24uwb88uxl0)

[2. Learning Objectives](#_9jnq83b0oogz)

[[Primitives] Variables and primitive data types](#_pnxio7m8x68b)

[[Strings] Strings and string processing](#_tl106yg6sj1m)

[[Expressions] Expressions](#_l6nx2nynya4j)

[[State] State](#_g6y6usw4xj2j)

[[Lists] Homogeneous Data Structures](#_4o4x2jwfeg2z)

[[Records] Heterogeneous Data Structures](#_kzqtyojwvci1)

[[NestedData] Nested Data](#_vxrwgwco6v7t)

[[Functions] Functions](#_xphn7gbtxi4s)

[[Selection] Selection](#_fkrt53uhwgx4)

[[Iteration] Iteration](#_gbkxeiry7t9t)

[[Recursion] Recursion](#_nj9bdhirtann)

[[Console] Simple Console I/O](#_y3s0blnejroy)

[[Files] File I/O](#_5cmyhtvuccva)

[[ControlFlow] Synthesis of Control Structures](#_pco94kfjwhj9)

[[Style] Style](#_p40mn5r0kvd3)

[[Documentation] Documentation](#_ue69tw1g6qii)

[[Testing] Testing](#_vddb7zh7ad3f)

[[Design] Design Ideas](#_hqzytzbmuefe)

[[Algorithms] Specific Algorithms](#_idg82e3amxm)

[[TimeComplexity] Algorithmic Time Complexity](#_yrxoy4gto291)

[[Plotting] Plotting](#_vsdtzluai94r)

[3. Changelog](#_ax7ljs7hq029)

# **1. Structure**

This document is organized into categories of learning objectives. Each category and learning objective (LO) can be identified by its bracketed code name (e.g., “[Primitives]”) and its friendly name (“Variables and primitive data types”).

Enduring, useful notes about the LO can be added below as indented bullets.

Temporary comments (e.g., suggestions, questions) can be made using the Google docs Comment feature.

Example assessment questions should be added as a full-width table below the LO.

## 1.1. Naming Things

Specific learning objectives can be fully referenced as:

*Course.Category.CodeName*

For example:

[CS1.Primitives.DescribePrimitiveTypes](#tjxx3svhh78d)

Of course, where it makes sense, you can use either just the category and code name, just the code name, or even just the friendly name.

Objectives are NOT numbered or organized that way, since they are unlikely to stay fixed over time.

## 1.2. Learning Objective Structure

If you are not familiar with learning objectives, there are specific processes to follow when writing them.

1. You must use observable, measurable, concrete action verbs.
2. The learning objective is focused on what the learner can do after instruction, not how you will teach.
3. Try to be specific, but be careful about over-combining separable objectives.

More information is available here and through Google: <https://www.cmu.edu/teaching/designteach/design/learningobjectives.html>

## 1.3. Assessment Requirements

If an objective is listed here, it is critical and should be on a secured exam (as opposed to a final project, in-class quiz, or take-home exam). However, all objectives must be covered by some kind of individual assessment, preferably an exam or project.

## 1.4. External Standards

After each objective, there may be one or more bracketed links to external standards that the objective connects with. Ideally, this should help with accreditation and deciding what to put into our program.

The most common standard referenced is the [CS2013 Curriculum Guidelines for Undergraduate Programs in Computer Science](https://www.acm.org/education/curricula-recommendations).

## 1.5. Example Assessment Questions

Each learning objective should be followed by one or more example assessment questions. Ideally, from a range of courses and professors.

## 1.6. Performance Objectives

We distinguish our Learning Objectives from *Performance Objectives*, which establish more specific conditions and criterion for success. The Learning Objectives only establish the desired behavior from the learner. Assessment Questions are a good way of specifying that behavior more clearly, but additional information is often necessary to specify criteria and conditions. For example, you might decide that demonstrating competency requires the student to “complete at least 1 out of 3 of these specific problems within 30 minutes with only this reference page of notes”.

To guide instructors, authors should provide specific criteria as notes and assessment questions as examples to better contextualize our departments’ threshold for competency.

In order to pass the CS1 requirement, students must meet some level of competency across *all* these learning objectives. In other words, they earn a D or F grade if they can’t meet every objective in some capacity. Think carefully about what criteria you want to establish, and how to connect your grades to those levels.

# **2. Learning Objectives**

After completing either 106 or 108, all students must be able to...

## **[****Primitives] Variables and primitive data types**

*(e.g., numbers, characters, Booleans)*

* [DescribePrimitiveTypes] Describe specific primitive data types. [[SDF2o1](https://www.inf.usi.ch/faculty/hauswirth/cc13/unit/SDF2#SDF2o2)]

|  |
| --- |
| For each of the given Python literal values, identify their type:  5  "37"  True  None  0.0  "Hello world" |

* [WritePrimitiveTypes] Write programs that use primitive data types. [[SDF2o2](https://www.inf.usi.ch/faculty/hauswirth/cc13/unit/SDF2#SDF2o3)]

|  |
| --- |
| Print out the following literal values: the integer 7, the float 9, the string representation of your name, the boolean value true, and the special None value. |

## **[****Strings] Strings and string processing**

* [IndexStrings] Write programs that index string values [[SDF3o3](https://www.inf.usi.ch/faculty/hauswirth/cc13/unit/SDF3#SDF3o3)]

|  |
| --- |
| Define a new function is\_surrounded that consumes a sentence (a string) and produces a boolean indicating whether the sentence begins and ends with parentheses. For example, "(This produces True)" but "This (produces False)". |
| Assume the following code is executed: name = "Hermione" What value is produced by each of the following expressions? name[0], name[-1], name[2] |

* [ManipulateStrings] Write programs that manipulate string values [[SDF3o3](https://www.inf.usi.ch/faculty/hauswirth/cc13/unit/SDF3#SDF3o3)]

|  |
| --- |
| Users ask too many questions. Create a function named calm\_down that takes in user input to get a sentence ending in '?', and then prints out the same sentence with a period ('.') by changing only the last character. Note that this function should take no parameters and returns nothing. |

## **[****Expressions] Expressions**

* [PredictMathOperators] Predict behavior of basic math operators (PEMDAS).

|  |
| --- |
| Predict the output of the following operations:   * 5+3 * 10-4 * 6/3 * 2\*4 * 10 % 3 * 1+(2\*3) |

* [PredictComparisonOperators] Predict behavior of basic comparison operators (Equality, Inequality, and Order).

|  |
| --- |
| Predict the output of the following operations:   * 5 == 5 * 4 != 4.0 * "7" == 7 * 1 < 10 * "A" < 1 |

* [PredictLogicOperators] Predict behavior of basic logic operators (And, Or, Not).

|  |
| --- |
| Assume the following code is run: is\_raining = True  is\_cold = False  is\_hot = False  Predict the value of the following operations:   * is\_raining and is\_cold, is\_raining or is\_cold * is\_cold and is\_hot * not is\_cold and not is\_hot |

* [PredictExpressions] Predict the behavior of simple programs involving the fundamental programming construct of expressions [[SDF2o1](https://www.inf.usi.ch/faculty/hauswirth/cc13/unit/SDF2#SDF2o1)]

|  |
| --- |
| Identify the type of each expression:   * (1 + 3) > 4 * (6/2) != 3.0 * not (1 < 1) and (4 > 3) |

## **[****State] State**

* [PredictSequentialAssignment] Predict the behavior of simple sequential programs involving the fundamental programming construct variables and assignment [[SDF2o1](https://www.inf.usi.ch/faculty/hauswirth/cc13/unit/SDF2#SDF2o1)]

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Trace the value of the two variables by filling out the table below. If a value has not been assigned to a variable, write X in the box. 1| up = 4  2| down = 20 - up  3| up = 3  4| down = up+up   |  |  |  | | --- | --- | --- | | **Step** | **up** | **down** | | 1 | \_\_\_ | X | | 2 | \_\_\_ | \_\_\_ | | 3 | \_\_\_ | \_\_\_ | | 4 | \_\_\_ | \_\_\_ | |

## **[Lists] Homogeneous Data Structures**

*We accept any “Homogeneous” data structure to cast a wide net of data types (e.g., arrays/lists/vectors/recursive list definition)*

*Our goal is not to demand mutation or immutability - either should be fine.*

* [CreateList] Write code to create a sequentially-accessed, homogenously-typed data structure based on an existing structure.

|  |
| --- |
| Define a function make\_lower that consumes a list of names (strings) and produces a new list with all the names lowercase. |

* [IndexList] Write programs that index a sequentially-accessed, homogenously-typed data structure [[SDF3o3](https://www.inf.usi.ch/faculty/hauswirth/cc13/unit/SDF3#SDF3o3)]
  + *First/Rest kind of access would be valid too*

|  |
| --- |
| Assuming that we ran lst=[3,2,1], for each expression, determine the value or error produced:   * Lst[1] * Lst[2] * lst[3] |
| Define a function chomp\_end that consumes a list and produces a new list without the last element. If the list is empty, return the empty list. |

LOs intentionally not included, but often recommended:

* Manipulating a sequentially-accessed, homogenously-typed data structure
  + E.g., mutation of a list, such as updating the value at a specific index.
* Creating a new sequentially-accessed, homogenously-typed data structure based on an old one immutably or mutably
  + E.g., “filter a list”
* Calling functions that use a sequentially-accessed, homogenously-typed data structure
  + E.g., len, sum, map in Python

## **[Records] Heterogeneous Data Structures**

*We accept any “Heterogenous” data structure to cast a wide net of data types (e.g., Dictionary/Map/Class+Object/Tuple/Struct/Record/Compound Type)*

*We have not established any Object-Oriented concepts here, or the concept of Objects or Classes specifically. POJOs and Structs would be acceptable.*

*Our goal is not to demand mutation or immutability - either should be fine.*

* [DefineRecord] Write programs that define an associatively-accessed, fixed-size, hetergenously-typed data structure [[SDF3o3](https://www.inf.usi.ch/faculty/hauswirth/cc13/unit/SDF3#SDF3o3)]

|  |
| --- |
| Create three dictionary literal values that are examples of a Dog, where each Dog is a dictionary with the keys "Name" (string), "Breed" (string), and "Age" (integer). A Dog's breed can be either 'Corgi', 'Poodle', or 'Beagle'. Be sure to document your dictionary Record's structure. |
| Create a class Dog that has three fields: name (str), breed (str), and age (int). Create two instances of a Dog. |

* [UpdateRecord] Write programs that update an associatively-accessed, fixed-size, hetergenously-typed data structure [[SDF3o3](https://www.inf.usi.ch/faculty/hauswirth/cc13/unit/SDF3#SDF3o3)]

|  |
| --- |
| Given the following code:  bart = {'age': 0, 'name': "Dr. Bart"}  bart\_clone = {'age': 0, 'name': "Dr. Bart"}  me = bart  def increase\_age(somebody):  somebody['age'] = somebody['age'] + 1  increase\_age(bart)  Mark any of the following expressions that produce the value 1:  bart['age'] bart\_clone['age'] me['age'] |

* [AccessRecord] Write programs that access data in an associatively-accessed, fixed-size, hetergenously-typed data structure [[SDF3o3](https://www.inf.usi.ch/faculty/hauswirth/cc13/unit/SDF3#SDF3o3)]

|  |
| --- |
| Define a function get\_dog\_age that consumes a Dog and produces the dog's age in dog years (multiplied by 7). Each Dog is a dictionary with the keys "Name" (string), "Size" (string, either "small" or "Large"), and "Age" (integer). If the dog's "Size" is "Large" then multiply by 8 instead. |

## **[****NestedData] Nested Data**

* [ProcessNestedHomogeneous] Write programs that process data in at least two levels of nested homogenous structure - for example, a 2-Dimensional array or list-of-lists.

|  |
| --- |
| Define a function sum-of-sums that consumes a lists of lists of integers and produces their grand total sum. |

* [ProcessNestedHeteroHomo] Write programs that process data in at least two levels of combined homogeneous and heterogenous structure - for example, a list of dictionaries or classes.

|  |
| --- |
| Define a function get\_corgi\_ages that consumes a list of Dogs and produces a new list of integers representing the ages of all the dogs of the breed "Corgis". Each Dog is a dictionary with the keys "Name" (string), "Breed" (string), and "Age" (integer). A Dog's breed can be either 'Corgi', 'Poodle', or 'Beagle'. |

## **[****Functions] Functions**

* [CallWithArgs] Write function calls that use arguments [[SDF2o1](https://www.inf.usi.ch/faculty/hauswirth/cc13/unit/SDF2#SDF2o1)]

|  |
| --- |
| How would you call a function named increase? Assume it takes only the value 1 as an argument. |

* [DefineFunctions] Write functions that incorporate parameters [[SDF2o1](https://www.inf.usi.ch/faculty/hauswirth/cc13/unit/SDF2#SDF2o1)]

|  |
| --- |
| You are responding to emails. Create a function named check\_emails that consumes a number of emails (an integer) and an amount of time you have (an integer), and then produces whether you have enough time (a boolean), by multiplying the number of emails by 10 minutes/emails and testing if the result is less than the amount of time you have. |

* [TraceFunction] Trace control and data flow within and between functions

|  |
| --- |
| What is printed after the code below is executed?  def change\_value(a\_variable):  a\_variable = 0  return a\_variable  a = 5  change\_value(a)  print(a) |
| Read the following code, and then fill in the blanks below.  1| def increase(number):  2| number = number + 1  3| print(number)  4| return number  5| age = 0  6| age = increase(age)  7| print(age)  What line is executed first? What line is executed second? After line 4 is executed, what will be the next line executed? |
| Given the program below,  response = input("Do you have a pet?")  def test\_pet(response):  return response.lower() == "yes"  def print\_message(response):  print("Do they have a pet?", response) The user will type in: Yes  And expect the output: Do they have a pet? True |
| Which of the following lines of code will have the data flow correctly through both functions in order to print the message with the result?  *Answers are various combinations of test\_pet and print\_message, either composing them or using assignment.* |

## **[****Selection] Selection**

*(e.g., If statements or functions/Switch/Cond)*

* [ConditionalExecution] Write programs that selectively branch execution based on a condition [[SDF2o5](https://www.inf.usi.ch/faculty/hauswirth/cc13/unit/SDF2#SDF2o5)]

|  |
| --- |
| Define a function named classify\_day which consumes a day of the week (string) and whether you are working tomorrow (a boolean), and produces a string. If you are not working tomorrow, produce the string "good". If the day of the week is either "friday" or "saturday", produce the string "good". Otherwise, produce the string "bad" |
| Read the following code, and then fill in the blanks below.  name = "Ada"  if name.lower() == "ada":  me = name  else:  me = "Missing"  has\_a = "A" in me  print(name)  print(me)  print(has\_a)  What will be printed first? What will be printed second? What will be printed third? |

.

## **[****Iteration] Iteration**

*(e.g., Foreach Loop/While or For Loop/Recursion)*

*Note: only one of either LoopList or LoopConditionally is critical. However, both are required.*

* [LoopList] Write programs that traverse a data structure using a loop [[SDF2o5](https://www.inf.usi.ch/faculty/hauswirth/cc13/unit/SDF2#SDF2o5)]

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Trace the values of this code. If a variable is not yet defined, put an "X" (without quotes).  profits = [5, 2]  total = 0  for profit in profits:  total = total + profit \* 2   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **step** | **line** | **profits** | **profit** | **total** | | 1 | 1 | [5, 2] | X | X | | 2 | 2 | [5, 2] | \_\_\_ | \_\_\_ | | 3 | 3 | [5, 2] | \_\_\_ | \_\_\_ | | 4 | 4 | [5, 2] | \_\_\_ | \_\_\_ | | 5 | \_\_\_ | \_\_\_ | 2 | \_\_\_ | | 6 | \_\_\_ | [5, 2] | \_\_\_ | 14 | |

* [LoopConditionally] Write programs that loop based on a condition [[SDF2o5](https://www.inf.usi.ch/faculty/hauswirth/cc13/unit/SDF2#SDF2o5)]

|  |
| --- |
| How many times does the loop body below execute?  command = ""  words = []  while command != "quit":  command = input("Type a word:")  words.append(command)  print(words)  Possible answers: None, At least once, At least twice, No more than 10 times |

## **[****Recursion] Recursion**

* [ExplainRecursiveFunction] Explain or apply the concept of a recursive function
  + Could just use this as an essay prompt, as-is. However, the better practice would be to have students actually implement a recursive function`.

|  |
| --- |
| We have a number of bunnies and each bunny has two big floppy ears. We want to compute the total number of ears across all the bunnies recursively (without loops or multiplication). Define a function count\_ears that uses recursion to count the bunnies ears. |

* [ExplainRecursiveData] Explain or apply the concept of a recursive data type
  + *Similar note to previous: could be at the explain level, or the apply.*

|  |
| --- |
| Corgis are composed of a name (string), fuzziness level (int), a mother (another Corgi), and a father (another Corgi). Create a recursive data type that can model Corgis. Then, create an instance of a Corgi family tree with at least three generations. |
| Given the following code:  1| **def** **count\_symbol**(character: str) -> bool:  2| **if** character **in** "!.?\*(),;":  3| **return** 1  4| **else**:  5| **return** 0  6|  7| **def** **count\_tree\_symbols**(tree: Tree) -> int:  8| **if** tree == None:  9| **return** 0  10| **else**:  11| left\_count = count\_tree\_symbols(tree['left'])  12| right\_count = count\_tree\_symbols(tree['right'])  13| this\_count = count\_symbol(tree['value'])  14| **return** left\_count + right\_count + this\_count  When the code reaches line 14, what is the most accurate type of the following expressions?   * tree * tree[‘left’] * right\_count * tree[‘value’] * this\_count * ‘left’ |

* [PredictRecursion] Predict the output of a recursive function

|  |
| --- |
| The following function is recursive. Predict the exact output from running the code.  def f(n):  if n == 0:  print("Done")  else:  print(n)  f(n-1)  f(4) |

## **[****Console] Simple Console I/O**

* [ConsoleOutput] Write programs that output information to the user [[SDF2o1](https://www.inf.usi.ch/faculty/hauswirth/cc13/unit/SDF2#SDF2o1)]
  + Bart: For those teaching Python, I strongly advocate against teaching “functions that print” and instead focus solely on pure functions that return values. “Print vs. Return” is a monster of our own creation that doesn’t really get us anything. Yes, they should know how to print - but no, they don’t need to write functions that print, just programs.

|  |
| --- |
|  |

* [ConsoleInput] Write programs that process input from the user [[SDF2o1](https://www.inf.usi.ch/faculty/hauswirth/cc13/unit/SDF2#SDF2o1)]
  + Bart: Same rule as above; they don’t need to know that functions can get input sideways. Stick to using `input` outside of the function definition.

|  |
| --- |
| Users are too excited. Create a function named calm\_down that takes in user input to get a sentence ending in !, and then prints out the same sentence with a period (.) by changing only the last character. Note that this function should take no parameters and returns nothing. |

## **[****Files] File I/O**

* [FileInput] Write programs that read data from a file [[SDF2o6](https://www.inf.usi.ch/faculty/hauswirth/cc13/unit/SDF2#SDF2o6)]

|  |
| --- |
| Each line of the attached file is a pair of comma-separated last and first names (e.g., Washington,George). Print the names in the correct order (e.g., George Washington) each on their own line. |

* [FileOutput] Write programs that store data in a file [[SDF2o6](https://www.inf.usi.ch/faculty/hauswirth/cc13/unit/SDF2#SDF2o6)]

|  |
| --- |
| Define a function store\_high\_score that consumes a list of integers and stores the highest number in a local file named high\_score.txt. |

## **[****ControlFlow] Synthesis of Control Structures**

* [MultipleControlStructures] Write programs that incorporate multiple control structures (e.g., selection, iteration, and functions).

|  |
| --- |
| Define a function find\_dollars that consumes a list of dollar amounts (list of non-empty strings) and produces a new list of strings with only strings that began with dollar signs. So the string "$100" would be kept but the string "99" would not. |
| A Present is a dictionary with the following keys:   * 'Weight' which is a float representing the Present's weight/size * 'For You?' which is a boolean indicating whether the Present is for you or not * 'Giver' which is a string representing the name of the person who gave you the present * 'Color' which is a string representing the color of the Presents' wrapping paper   Define a function color\_of\_my\_biggest\_present that consumes a list of Presents and produces a string representing the color of the highest weight Present that is for you. If no present is found, return the string value 'nothing' instead.  Pay close attention to the key's capitalization and spelling. |
| A Table is a dictionary with the following keys:   * 'Width' which is an integer representing the width of the table * 'Height' which is a integer representing the height of the table * 'Color' which is a string representing the color of the table   Define a function get\_areas that consumes a list of Tables and produces a list of integers representing the areas of each table (width multiplied by height). |

## **[****Style] Style**

* [EvaluateIdentifiers] Evaluate identifier names for their quality [[SDF4o11](https://www.inf.usi.ch/faculty/hauswirth/cc13/unit/SDF4#SDF4o11)]

|  |
| --- |
| You have a variable that represents your allowance, measured in dollars. Which of the following are *good* variable names? In this case, "good" means that they are not too long, not too short, and the intent of the variable is unambiguous. Options are: allowance, d, allowance\_in\_dollars, integer, a\_variable\_that\_will\_hold\_my\_allowance\_money. |

* [ImproveIdentifiers] Propose a better name for an identifier

|  |
| --- |
| Read the code below, meant to do some basic calculations.  # Get user's name  n = input("What is your name?")  # Print a welcome message  print("Welcome", n)  # Get user's age; input returns a string  ageAsStr = input("How old are you?")  # Convert string to integer using built-in int function  # The int function turns strings into integers - we'll learn more about it soon  aai = int(ageAsStr)  # Calculate age in dog year's  dg\_yrs = aai \* 7  # Print result  print("You are", dg\_yrs, "in dog years!")  Some of the variable names are poor. For all the variables in the program, do the following: 1. Clearly write the name of the variable and list the lines where it is written and read.  2. Label whether the variable's name is \*\*good\*\* or \*\*bad\*\*.  3. If you identify it as bad, write an argument for why it does not meet the criteria established above.  4. If you identify it as good, write an argument defending it from someone who wants to change it.  5. Suggest a better variable name. |

* [FixRepetition] Propose modifications to code that avoids unnecessary repetition [[SDF4o7](https://www.inf.usi.ch/faculty/hauswirth/cc13/unit/SDF4#SDF4o7)]

|  |
| --- |
| The expression below extracts the first half of a string by using the len built-in function to calculate the end of the string, halving that length, and then converting the result to an integer (recall that division results in a float). Extract this expression into a function named get\_half and replace the existing expressions with function calls.  # Starting Code:  print("Banana"[:int(len("Banana")/2)])  print("ABCDEFG"[:int(len("ABCDEFG")/2)])  print("Half & Half"[:int(len("Half & Half")/2)]) |

## **[****Documentation] Documentation**

* [DocumentFunctions] Document programs at the function level using contracts [[SDF4o6](https://www.inf.usi.ch/faculty/hauswirth/cc13/unit/SDF4#SDF4o6)]

|  |
| --- |
| Provide documentation for the following function. The function consumes a string, an integer, and a float; the function produces a boolean. Do not worry about the implementation of the function or its unit tests. You need to provide descriptions of the function, its arguments, and the return value.  from cisc108 import assert\_equal  def is\_expected\_user(name, age, height):  return name == "Ada" and age == 1 and height == 11.7  assert\_equal(is\_expected\_user("Ada", 1, 11.7), True)  assert\_equal(is\_expected\_user("Pumpkin", 4, 9.8), False) |

* [ReadDocumentation] Interpret documentation of externally created code
  + Bart: In projects, I could have students use functions I have defined in an external library, that they must read up on and use.

|  |
| --- |
|  |

## **[****Testing] Testing**

* [CreateUnitTests] Create unit tests for code with sufficient case coverage [[SDF4o8](https://www.inf.usi.ch/faculty/hauswirth/cc13/unit/SDF4#SDF4o8)]
  + Bart: In my Fall 2019 course assignments, I secretly used 75% code coverage for projects as a rule of thumb for “sufficiently covered”. However, it’s dangerous to rely on numbers - the goal is really to have them think about discrete cases of input rather than lines of code.

|  |
| --- |
|  |

* [InterpretUnitTests] Interpret the results of executing externally provided unit tests

|  |
| --- |
| Given this function, which of the following are correct unit tests?  from cisc108 import assert\_equal  def calculate(x):  return 2 \* x + 1 assert\_equal(4, 9), calculate(4) == 9, assert\_equal(calculate(4), 9), calculate(assert\_equal(4), 9), assert\_equal(calculate, 0, 1), assert\_equal(calculate(0), 1) |

* [JustifyUnitTests] Explain why the creation of individually-tested program components is important in the production of high-quality software. [[SDF4o2](https://www.inf.usi.ch/faculty/hauswirth/cc13/unit/SDF4#SDF4o2)]
  + Bart: Could just use this as an essay prompt, as is.

|  |
| --- |
|  |

## **[****Design] Design Ideas**

* [DecomposeProgram] Write programs that decompose an algorithmic solution into smaller pieces [[SDF1o8](https://www.inf.usi.ch/faculty/hauswirth/cc13/unit/SDF1#SDF1o8)]
  + *I am thinking about having a really complex problem (in the style of the Rainfall question) with many "phases" that is best when decomposed. Students are told to use functional decomposition in their solution, and then they can be judged based on that.*

|  |
| --- |
|  |

* [AbstractReality] Write programs that represent reality using data [Custom]:

|  |
| --- |
| A person can be described by a dictionary with the 5 keys "Name" (String), "Age" (Integer), "Height" (Float), "Is nice?" (Boolean), and "Recent Facebook Status" (A list of strings). I could create a list of 3 people by making person dictionaries to represent myself, my best friend, and my neighbor. Identify 4 things that you can identify with dictionaries. Those dictionaries must have at least 5 keys, each with a different type - and one type must be a non-empty composite type (e.g., a list or a dictionary). For each kind of dictionary, make 3 instances. |

## **[****Algorithms] Specific Algorithms**

* [ImplementLinearSearch] Implement and algorithmically analyze a linear time search algorithm

|  |
| --- |
| Write a function find\_number that consumes a list of integers and another integer, and produces a boolean indicating whether the integer is in the list. What is the Big O time complexity of your solution? |

* [ImplementBinarySearch] Implement and algorithmically analyze a logarithmic binary search algorithm

|  |
| --- |
| Write a function find\_number\_fast that consumes a *sorted* list of integers and another integer, and produces a boolean indicating whether the integer is in the list. Your solution must take advantage of the sortedness of the list in order to perform more efficiently. What is the Big O time complexity of your solution? |

* [ImplementQuadraticSort] Implement and algorithmically analyze a quadratic time sorting algorithm (e.g., selection or insertion sort)

|  |
| --- |
| Write a function sort\_values that consumes a list of dictionaries representing books, and produces the same list sorted by the book's 'Price' attribute (a float). You may not use the sort or sorted built-in functions. For any credit, you must explain how your algorithm works and what its Big O time complexity is. |

* [AnalyzeLinearithmicSort] Algorithmically analyze a linearithmic time sorting algorithm (e.g., merge sort)

|  |
| --- |
| Review the following implementation of merge\_sort, explain how it works, and give its Big O time complexity. |

## **[****TimeComplexity] Algorithmic Time Complexity**

* [IdentifyTimeComplexity] Identify code as having Constant, Linear, or Quadratic time complexity.

|  |
| --- |
| Mark each of the following operations as having Constant, Linear, or Quadratic time complexity: 1) determining the sum of a list of numbers, 2) determining if an element is in an unsorted list, 3) adding together five numbers, 4) adding together 10000 numbers, 5) determining the lowest value in a list of numbers. |

* [DefineTimeComplexity] Explain the meaning of algorithmic time complexity.
  + *Could just use this as an essay prompt, as-is. -acbart*

## **[****Plotting] Plotting**

*These requirements are only for CISC106, currently!*

* [MakePlot] Write a program that generates a graphical visualization of structured data using a plotting library.

|  |
| --- |
| The JSON file salaries.json contains salary information for employees in a small company. Create a histogram to visualize the distribution of salaries among all employees. |

# **3. Changelog**

* Version 0.0.4 (Started September 12, 2024):
  + Minor edits to some wording
* Version 0.0.3 (Started January 8, 2022):
  + Converted from Github Markdown to Google Doc, for convenience
  + Added some prelude information
  + Added some examples to of MultipleControlStructures
  + Modified [Algorithms] a little to make it a bit more flexible while still focusing on specific types of algorithms (searching and sorting)
  + Renamed the Heterogeneous and Homogenous categories/LOs to be “Lists” and “Records” so that they are more accessible concepts. Retaining the wording in the full names so that they can still cast a wide net.
  + Renamed “Synthesis” category to “ControlFlow” to emphasize this is about combining control structures, as opposed to “NestedData” which is about combining data structures.
* Version 0.0.2 (Started August 8/19/2019):
  + Separated out a new section on Nested data (2D arrays and lists-of-dictionaries)
  + Modified exemplar question of [AccessHeterogeneous] to better demonstrate individual record access
  + Added in term "Compound Type" as another valid kind of Heterogenous data structure
  + Added in term "fixed-size" as part of the Heterogenous data structure's requirements, to differentiate from arbitrarily-sized uses (e.g., a dictionary used as a counter for an unknown set of elements).
  + Reordered Functions to come before control structures (Data -> Functions -> Control -> I/O -> Design)
  + Expanded learning objectives for Recursion to have both "recursive functions" and "recursive data types". Also specified that although "Explain" is fine, the recommended practice is "Define".