Summary of CAS Language (CASL) Fundamentals

General CAS Concepts

- Use the CAS statement to make a connection to CAS.

```
cas <sessionName> <options>;
```

Example:

```
cas conn sessopts=(timeout=1800 metrics=TRUE);
```

CASL Programming Components

- The CAS language is very different from the traditional SAS programming language. CASL which is more like Python, includes 4 major components:
  - CAS actions
  - variable data types
  - statements
  - functions

- Use the CAS Procedure to execute CASL code:

```
proc cas;
     …;
quit;
```

- **CAS Actions**

```
proc cas;
    action-set.action-name < / parameter=value, parameter=value, ...>;
quit;
```

- **CAS Variable Data Types**

	Data Types
Numeric	double, INT32 and INT64
String	varchar and string
Boolean	stores the values of TRUE or FALSE
Other	array, dictionary, and table

- **Statements**

Statement Syntax	What it does
DESCRIBE *variable-name* | *expression*;	writes the structure and data type of CASL variables and expressions to the log
PRINT *value-1 <value-2>...<value-n>*;	writes the values of constants, variables, and expressions to the current output location

- **Arrays**
  - A CASL array is one of the two list data types.
  - Arrays are most useful with CASL programming for grouping a series of strings or numbers in a variable and then using the variable as a parameter to a CAS action.

Syntax	What it does
array-name=**{***value-1* <**,***value-2* ...>**}**;	defines an array
array-name[*position***]** **array-name[***lower-bound***:***upper-bound***]** **array-name[{***position-n*, …, *position-z***}]**	access array elements

  - Loop over an array:

```
DO <variable> OVER <array-name>;
    ... repetitive CASL code ...
END;
```

  - Array Operators

Syntax	What it does
array-1 **=** *array-1* **||** *value*; *array-1* **=** *array-1* **||** *array-2*;	appends to an array
variable **=** *array-1* **/** *array-2*;	finds unique values in arrays
variable **=** *array-1* **&** *array-2*;	finds common values in arrays
variable **=** *array-1* **==** *array-2*;	compares two arrays
variable **=** *value* **==** *array*;	checks for a single value in an array

- Array Functions

Function	What it does
DIM(*array***);**	returns the number of elements in an array
SORT(*array***);**	return an array in ascending order
SORT_REV(*array***);**	returns an array in descending order

- **Dictionaries**
  - Creating dictionaries

```
dictionary-name = {key-1=value-1 <, key-n=value-n,...>};

dictionary-name.key-1 = value-1;
<dictionary-name.key-n = value-n;>

dictionary-name["key-1"] = value-1;
<dictionary-name["key-n"] = value-n;>
```

  - Accessing dictionary values

```
dictionary-name["key"]

dictionary-name.key
```

  - Deleting a dictionary key

```
DELETE dictionary-name["key"]
DELETE dictionary-name.key
```

  - Loop over an dictionary

```
DO <key> ,<value> OVER <dictionary>;
    ... repetitive CASL code ...
END;
```

- **CAS Actions Overview**

- CAS actions return a dictionary back to the client.
- There are no rules about how many keys are contained in the dictionary, or what data types are returned.
- You can store the results of a CAS action in a variable:

```
action-set.action-name <result = results-variable> / ...;
```