**IB Pseudocode Syntax**

Basic instructions

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Syntax** | **Description** | **Examples** |
| Assign | *variable* = *value* | Assigns a value to the variable | a = 1 |
| Input | input *variable* | Inputs the variable | input a |
| Output | output *variable/expression* | Outputs a value of the variable or expression | output a  output “hello”  output 2 + 2 |
| Delete | delete *variable* | Deletes the variable | delete a |

Conditions

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Syntax** | **Description** | **Examples** |
| If | if *condition* | Indicates the start of a condition block and states the first condition | if a = 1 |
| Else if | else if *condition* | States an additional condition | else if a = 2 |
| Else | else | Indicates the start of the part of a condition block which will be executed if all conditions above are False | else |
| End if | end if | Indicates the end of a condition block | end if |

Loops

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Syntax** | **Description** | **Examples** |
| While loop | loop while *condition* | Executes a loop block while the condition is true | loop while a < 5 |
| Until loop | loop until *condition* | Executes a loop block until the condition is true | loop until a == 5 |
| For loop | loop *variable* from *start value* to *end value* | Executes a loop block for every value of the variable between start value and end value | loop a from 1 to 5 |
| End loop | end loop | Indicates the end of a loop block | end loop |

Functions

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Syntax** | **Description** | **Examples** |
| Function | function *name*(*arg1, …)* | Indicates the start of a function block with name and arguments | function f(a, b) |
| Return | return *variable/expression* | Returns value or expression from function | return a  return “hello”  return 2 + 2 |
| End function | end function | Indicates the end of a function block | end function |
| Run function | *name*(*arg1, …)* | Runs a function block with given name and arguments | f(1, 2) |

Procedures

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Syntax** | **Description** | **Examples** |
| Procedure | procedure *name*(*arg1, …)* | Indicates the start of a procedure block with name and arguments | procedure p(a, b) |
| End procedure | end procedure | Indicates the end of a procedure block | end procedure |
| Run procedure | *name*(*arg1, …)* | Runs a procedure block with given name and arguments | p(1, 2) |

Arrays

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Syntax** | **Description** | **Examples** |
| Create array | array *name* | Creates an empty array with given name | array a |
| Get item | *name*[*index*] | Returns an item with given index from an array with given name | a[0] |
| Set item | *name*[*index*] = *value* | Assigns a value to given index from an array with given name | a[0] = 1 |
| Assign array | *name* =[*val1, val2, …*] | Assigns an array with given values to variable with given name | a = [1, 2, 3] |

Dictionaries

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Syntax** | **Description** | **Examples** |
| Create dictionary | dictionary *name* | Creates an empty dictionary with given name | dictionary a |
| Get item | *name*[*key*] | Returns an item with given key from a dictionary with given name | a[“a”] |
| Set item | *name*[*key*] = *value* | Assigns a value to given key from a dictionary with given name | a[“a”] = 1 |
| Assign dictionary | *name* ={*key1*: *val1, key2*: *val2, …*} | Assigns a dictionary with given values to variable with given name | a = {“a”: 1, “b”: 2} |

Collections

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Syntax** | **Description** | **Examples** |
| Create collection | collection *name* | Creates an empty collection with given name | collection a |
| Add item | *name*.addItem(*value*) | Adds a value to the end of the collection | a.addItem(1) |
| Get next | *name*.getNext() | Returns next value from the collection | a.getNext() |
| Reset next | *name*.resetNext() | Resets next element of the collection | a.resetNext() |
| Has next | *name*.hasNext() | Checks does the collection have next element | a.hasNext() |
| Is empty | *name*.isEmpty() | Check does the collection contains elements | a.isEmpty() |

Stackss

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Syntax** | **Description** | **Examples** |
| Create stack | stack *name* | Creates an empty stack with given name | stack a |
| Push | *name*.push(*value*) | Adds a value to the stack | a.push(1) |
| Pop | *name*.pop() | Gets a value from the stack | a.pop() |
| Is empty | *name*.isEmpty() | Check does the stack contains elements | a.isEmpty() |

Queues

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Syntax** | **Description** | **Examples** |
| Create queue | queue *name* | Creates an empty queue with given name | queue a |
| Enqueue | *name*.enqueue(*value*) | Adds a value to the queue | a.enqueue(1) |
| Dequeue | *name*.dequeue() | Gets a value from the queue | a.dequeue() |
| Is empty | *name*.isEmpty() | Check does the queue contains elements | a.isEmpty() |