Programming

Variables, data types, selection and loops, strings, operators, procedures and functions, library routines, arrays and file handling, with the Cambridge pseudocode and Python side by side.

NOTE

The programming language used in these notes is Python. Each example is shown in the pseudocode the exam uses, and then in Python.

8.1 Programming concepts

Variables and Constants

A **variable** is a named data store that contains a value that may change during the execution of a program.

A **constant** is a named data store that contains a value that does not change during the execution of a program. The value only needs to be changed once, if circumstances change.

Pseudocode:

```
DECLARE Num1 : INTEGER
Num1 ← 30

CONSTANT Constant1 ← 500
```

Python:

PYTHON

```
Num1 = 30
Constant1 = 500
```

NOTE

Python does not need a separate statement to declare variables, nor does it have a way to differentiate between constants and variables.

Note Academy

Basic Data Types

The purpose of data types in programming is to specify the type of data that can be stored and manipulated within a program. For example, mathematical calculations can only be performed on integers and float values.

Pseudocode:

```
DECLARE Number1 : INTEGER
DECLARE RealNumber : REAL
DECLARE Text1 : STRING
DECLARE Character1 : CHAR
DECLARE BooleanValue : BOOLEAN
```

Python:

```python
Number1 = 4
RealNumber = 3.4
Text1 = "Note Academy"
Character1 = "E"
BooleanValue = True
```

Python does not differentiate between single quotes and double quotes.

Input and Output

Pseudocode:

```
OUTPUT "Please enter your name: "
INPUT Name
```

Python:

```python
firstname = input("Please enter your name: ")
print("My name is ", firstname)
```

NOTE

Note Academy

Python has the output and input commands in the same statement. Also remember that Python always takes input as a **string**.

To only output something, use the `print` function. Use a comma to combine variables and strings. Remember to use speech marks.

Selection Statements

IF ELSE statement

PYTHON

```
if NameOfAcademy == "Note Academy":
    print("The best academy to ever exist. Period.")
else:
    print("Nope, not even close to Note Academy.")
```

NOTE

To check if one thing is equal to another in Python, use the equals sign (`=`) two times (`==`). A single `=` assigns a value.

MATCH CASE statement

PYTHON

```
match grade:
    case "A":
        print("Grade = A")
    case "B":
        print("Grade = B")
    case "C":
        print("Grade = C")
    case _:
        print("Failureeeee")
```

NOTE

Remember to use a colon after every case. The case with an underscore (`_`) is the case that is used when there is no exact match with the other cases.

Note Academy

Loops

Count-controlled loop (FOR loop)

PYTHON

```
for x in range(10):
    print("-")
```

NOTE

`range(start, stop, step)`, `range(start, stop)` and `range(stop)`.

These are the three ways you can use the range function. The step parameter is used to set the increment or decrement. By default, which is the third one in the examples above, the range function starts at 0 and ends at stop − 1. The loop in the example will use the values 0 1 2 3 4 5 6 7 8 9, so it prints the dash 10 times.

Pre-condition loop (WHILE loop)

PYTHON

```
x = 0
while x < 10:
    print("Note Academy = Success")
    x = x + 1
```

String Handling

Finding the length

Pseudocode:

```
LENGTH("Note Academy")
LENGTH(Name)
```

Python:

PYTHON

```
len("Note Academy")
len(Name)
```

Output: `12`

Note Academy

Substring

Pseudocode:

```
SUBSTRING("Note Academy", 1, 4)
SUBSTRING(String1, 1, 4)
```

Python:

```python
"Note Academy"[0:4]
String1[0:4]
```

Output: `"Note"`

> **NOTE**
>
> The first character in a string in pseudocode is at position 1. In Python it is at position 0. Also note that the character at position 4 will not be included.

Upper

Pseudocode:

```
UCASE("Note Academy")
UCASE(String1)
```

Python:

```python
"Note Academy".upper()
String1.upper()
```

Output: `"NOTE ACADEMY"`

Lower

Pseudocode:

```
LCASE("Note Academy")
LCASE(String1)
```

Note Academy

Python:

```python
"Note Academy".lower()
String1.lower()
```

Output: `"note academy"`

Operators

Action	Pseudocode	Python
Add	+	+
Subtract	-	-
Multiply	*	*
Divide	/	/
Raised to the power of	^	**
Less than	<	<
Greater than	>	>
Equal	=	==
Less than or equal	<=	<=
Greater than or equal	>=	>=
Not equal	<>	!=
Both true	AND	and
Either true	OR	or
Not true	NOT	not

Subroutines: Functions and Procedures

A **subroutine** is a sub-program that performs a frequently used operation within a program. It can be called when needed and can be reused by another program.

Procedures and functions are both subroutines. However, in contrast to procedures, **functions always return a value**.

Note Academy

Both functions and procedures can have parameters. **Parameters** are variables used to accept input values when a subroutine is called. They allow subroutines to be flexible and reusable by operating on different data inputs.

Procedure (without parameters)

Pseudocode:

```
PROCEDURE Greetings
    OUTPUT "Hello. Welcome to Note Academy."
ENDPROCEDURE

CALL Greetings
```

Python:

```python
def Greetings():
    print("Hello. Welcome to Note Academy.")

Greetings()
```

Output: `Hello. Welcome to Note Academy.`

Procedure (with parameters)

Pseudocode:

```
PROCEDURE Division(Num1 : INTEGER, Num2 : INTEGER)
    OUTPUT Num1 / Num2
ENDPROCEDURE

CALL Division(30, 5)
```

Python:

```python
def Division(Num1, Num2):
    print(Num1 / Num2)

Division(30, 5)
```

Output: `6` (Python prints `6.0`, because `/` always gives a float)

Note Academy

Function

Pseudocode:

```
FUNCTION Division(Num1 : INTEGER, Num2 : INTEGER) RETURNS REAL
    RETURN Num1 / Num2
ENDFUNCTION

DECLARE answer : REAL
answer ← Division(30, 5)
OUTPUT answer
```

Python:

PYTHON

```
def Division(Num1, Num2):
    return Num1 / Num2

answer = Division(30, 5)
print(answer)
```

It must be kept in mind that the **arguments**, which are the actual values or data passed to a subroutine's parameters when the subroutine is called, should be of the same data type according to which the function was made. For example, if you pass text as an argument, it will give an error.

The first line in which the function or procedure is declared is called the **header**.

Local and Global Variables

Local variables are defined within a function or block and can only be accessed within that function or block.

Global variables are defined outside of any function and can be accessed by any part of the program.

Note Academy

PYTHON

```
global_var = "I am global"

def example_function():
    local_var = "I am local"
    print(local_var)
    print(global_var)

example_function()
print(global_var)
print(local_var)
```

Output:

```
I am local
I am global
I am global
Error
```

`global_var` is a global variable accessible both inside and outside `example_function`.

`local_var` is a local variable accessible only within `example_function`. Attempting to access it outside the function will result in an error as shown above.

Library Routines

Library routines are pre-written subroutines that are available for use by programmers. Using library routines can save time and effort, as programmers can leverage existing, tested code instead of writing these functions from scratch.

Pseudocode:

```
Num1 ← MOD(7, 3)       // returns the remainder
Num2 ← DIV(7, 3)       // returns the quotient
Num3 ← ROUND(3.142, 1) // rounds it to 1 decimal place
Num4 ← RANDOM()        // generates a random number between 0 and 1
OUTPUT Num1, ",", Num2, ",", Num3, ",", Num4
```

Python:

Note Academy

PYTHON

```
Value1 = 7 % 3        # MOD
Value2 = 7 // 3       # DIV
Value = divmod(7, 3)   # returns 2 numbers, div and then mod
Value3 = round(3.142, 1)
from random import random
Value4 = random()
print(Value1, ",", Value2, ",", Value3, ",", Value4)
```

NOTE

The text after `//` in pseudocode and after `#` in Python is called a comment. **Comments** are non-executable lines in code used to describe and explain the purpose or functionality of specific parts of the code. They are intended to make the code more readable and maintainable for humans.

In pseudocode, use 2 forward slashes to start a comment. In Python, use a hash to start a comment.

8.2 Arrays

Arrays

Arrays are used to store and manage collections of data, particularly when dealing with multiple values of the same type. They allow efficient access and manipulation of data using indices.

One-dimensional array

1D arrays are like lists. They store multiple values of the same type in one variable.

Pseudocode:

```
DECLARE Array1 : ARRAY[0:3] OF INTEGER
Array1[0] ← 5
Array1[1] ← 9
Array1[2] ← 7
Array1[3] ← 3
FOR i ← 0 TO 3
    OUTPUT Array1[i]
NEXT i
```

Python:

Note Academy

```python
Array1 = [5, 9, 7, 3]
print(Array1)
```

Output: `[5, 9, 7, 3]`

Two-dimensional array

2D arrays are collections of elements arranged in rows and columns, forming a grid. They are useful for representing matrices or tables.

Pseudocode:

```
DECLARE Array1 : ARRAY[0:2, 0:1] OF INTEGER
Array1[0,0] ← 1    // Array[rows, columns]
Array1[0,1] ← 2
Array1[1,0] ← 3
Array1[1,1] ← 4
Array1[2,0] ← 5
Array1[2,1] ← 6
FOR RowCounter ← 0 TO 2
   FOR ColumnCounter ← 0 TO 1
      OUTPUT Array1[RowCounter, ColumnCounter]
   NEXT ColumnCounter
NEXT RowCounter
```

Python:

```python
Array1 = [[1, 2], [3, 4], [5, 6]]
print(Array1)
```

Output: `[[1, 2], [3, 4], [5, 6]]`

8.3 File handling

File Handling

Files can be used to save data between program executions, ensuring that information is not lost when the program is closed. They can also handle large amounts of data that may not fit into the program's memory at once.

Pseudocode:

Note Academy

```
DECLARE Line1 : STRING
DECLARE File1 : STRING
File1 ← "NoteAcademy.txt"

OPEN File1 FOR WRITE        // opens the file in write mode

OUTPUT "Enter a line of text: "
INPUT Line1

WRITEFILE File1, Line1      // stores Line1 in the file

CLOSEFILE File1             // closes the file

OPEN File1 FOR READ         // opens the file in read mode

READFILE File1, Line1       // stores the line from the file into Line1
OUTPUT Line1

CLOSEFILE File1
```

Python:

PYTHON

```
File1 = open("NoteAcademy.txt", "w")
Line1 = input("Enter a line of text: ")
File1.write(Line1)
File1.close()
File1 = open("NoteAcademy.txt", "r")
Line1 = File1.read()
print(Line1)
File1.close()
```

Note Academy