Variables and Constants

Variables are assigned using the = operator.

- A good variable name describes the data it contains
- It is convention in Python to use all lower case letters for variable name, using underscore_separators or CamelCase

```
num = 3
name = "Bob"
```

Declaring constants

PSEUDOCODE	PYTHON
const pi = 3.14	Import math PI = math.pi

Input	myName = input("Please enter your name")
Casting	num = int(input("Please enter a number") price = float(input("Please enter a number")

Output	print("Hello world")
	print(variable)

Random number	import random
	number=random.randint(1,6)
	print(number)

Comparison operators

==	Equal to
!=	Not equal to
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to

Arithmetic operators

+	Addition e.g. x=6+5 gives 11
-	Subtraction e.g. x=6-5 gives 1
*	Multiplication e.g. x=12*2 gives 24
/	Division e.g. x=12/2 gives 6
MOD	Modulus e.g. 12MOD5 gives 2 In python: %
DIV	Quotient e.g. 17DIV5 gives 3 In Python: //
^	Exponentiation e.g. 3^4 gives 81 In python: **

Selection – using IF statement

PSEUDOCODE

```
if entry == "a" then
   print("You selected A")
elseif entry=="b" then
   print("You selected B")
else
   print("Unrecognised ")
endif
```

PYTHON

```
if entry == "a":
   print("You selected A")
elseif entry=="b":
   print("You selected B")
else:
   print("Unrecognised ")
```

Count-controlled loop

PSEUDOCODE

Print hello 8 times:

```
for i=0 to 7
   print ("Hello")
next i
```

PYTHON

Print hello 8 times:

```
for i in range(8)
   print ("Hello")
```

Print hello 4 times:

```
for i in range(0,8,2)
   print ("Hello")
```

range(start_value, stop_value, step_value)

Condition-controlled loop

PSEUDOCODE

This will loop until the user inputs "x". It will check the condition before entering the loop.

```
while answer!="computer"
    answer=input("What is the password?")
endwhile
```

PYTHON

```
while answer!= "x":
   answer = input("Press any key to continue or x to quit")
```

Working with strings	
Length of a string	
PSEUDOCODE	**PYTHON**

subject = "Computer Science"
print(subject.length)
 	
will return 15
	subject = "Computer Science"
	print(len(subject))

Substring	
PSEUDOCODE	**PYTHON**

stringname.subString(startingPosition, numberOfCharacters)
 	
The string will start with the 0th character.
 	
Example:
 	
someText=”Computer Science”
print(someText.substring(3,3))
 	
Will display: put
	someText=”Computer Science”
	someText[start:end] extracts characters from start (inclusive) to end (exclusive).
	
	print(someText[3:6])
	Will display: put
	
	substring = text[:8] # Extracts the first 8 characters
	print(substring)
	Will display: Computer
	
	substring = text[9:] # Extracts from index 9 to the end
	print(substring)
	Will display: Sceince

Python string functions	
split () a string into a list subject = "Computer Science" print(subject.split()) [“Computer”,”Science”] **strip() –** used to remove whitespaces after and before a string	
This converts the case of the string to either upper or lower case.	
PSEUDOCODE	**PYTHON**

subject.upper
subject.lower
	subject.upper()
	subject.lower()

This converts to and from ASCII.	
PSEUDOCODE	**PYTHON**

ASC(A) will return 65 (numerical)
CHR(97) will return "a" (char)
	ord("A")
	chr(97)

Sub-routines

Easily reuse a block of code within a program
Logical structure so easier to maintain and debug.
Reused by other programmers
Speeds up development time

Subroutine/Subprogram	A sequence of instructions to perform a specific task with an identifiable name.
Function	A subroutine that returns a value.
Procedure	A subroutine that executes a block of code when called. It does not return a value.
Parameter	Used in a subroutine to allow values to be passed into them.
Argument	The values held in the brackets of a subroutine call. These are passed into a subroutine via the parameters.
Decomposition	Breaking down a problem into smaller subproblems to make the more manageable.

Functions

A <u>function</u> is also a small section of a program that performs a specific task that can be used repeatedly throughout a program, but functions perform the task and return a value to the main program.

Pseudocode:

```
function triple(number)
    cubedNumber=number*3
    return cubedNumber
endfunction
```

Python:

```
y= triple(7)
def triple(number):
    cubedNumber=number*3
    return cubedNumber
y= triple(7)
```

Procedures

A <u>procedure</u> is a small section of a program that performs a specific task. Procedures can be used repeatedly throughout a program.

Pseudocode:

```
procedure greeting(name)
   print("hello"+name)
endprocedure
greeting("Gemma")
```

Python:

```
def greeting(name):
   print("hello"+name)


greeting("Gemma")
```

Array	
PSEUDOCODE	**PYTHON**
Arrays will be 0 based	
array names[5]	names = []
names[0]=”Ahmad”	names.append("Ahmad")
names[1]=”Ben”	names.append("Ben")
names[2]=”Catherine”	names.append("Catherine")
names[3]=”Dana”	names.append(“Dana")
names[4]=”Elijah”	names.append(“Elijah")
print(names[3])	print(names[3])

Python Printing the list:

```
names = ["sarah", "max", "jake”]
 for i in range(len(names))
    print(names[i])
```

```
for name in names:
    print(name)
```

Looking for an item in a list (Linear Search)

```
found = False
index = 0
while not found and index < len(names):
    if searchItem = names[index]:
        found = True
        print(“found”)

if not found:
    print(“Not found”)
```

Counting items in a list:

Counting the number of times ‘sarah’ appears in the list:

```
counter = 0
for i in range(len(names))
     if names[i] = ‘sarah’:
         counter = counter + 1
print(counter)
```

Checking if an item is not in a list:

```
if newName not in names:
    print(newname + " is not in the list")
```

Python Functions to Sort a list	
mylist.sort()	
mylist.sort(reverse = True)	

2D List	
PSEUDOCODE	**PYTHON**
array board[4,4]	n = 4
board[0,0]="rook"	board = [[] for i in range(n)]
	board[0,0]="rook"

Open a file to read	
PSEUDOCODE	**PYTHON**

myFile = openRead("sample.txt")
 	
# Read the first line 	
x = myFile.readLine()
 	
# Read all remaining lines into a list 	
 	
list = myFile.readLines()
myFile.close()
	myFile = open("myFilename",'r')
	# Read the first line
	x = myFile.readline()
	# Read all remaining lines into a list
	lines = myFile.readlines()
	myFile.close()
	
	Another way:
	with open("myFilename", 'r') as myFile:
	    x = myFile.readline()

To open a file to write to	
PSEUDOCODE	**PYTHON**

myFile = openWrite("sample.txt")
myFile.writeLine("Hello World")
myFile.close()
	myFile = open("myFilename","w")
	line = “Hello World” + ‘\n’
	myFile.write(line)
	myFile.close()
	w - write
	a – open for appending only