Computing Year 10 Cycle 1 Computer Science

1.5 OCR reference language

key word	definition	example
generic variable	Assigned using the **= operator** , **can change/update** throughout the program	age=11 name = "Laura"
constant variable	value **remains the same** throughout the program	const vat = 0.25 drivers = "Mastery" "Autonomy" "Purpose"
global variable	keyword **global** required in assignment	global userID="12345"
variable data type	assumes the **type of data** it is given	name = str(input("type in name")) age=int(input("how old are you?")) height=float(input(How tall are you?"))
casting	**changing the data** type: str, float, int	age = input(" ")) → age=int(input())
outputting to screen	**displaying** data out of the program	print("My name is Bob")
input	request user to **type into** the program	name=input()
iteration	**repeated** sections of code	for x = 0 to 4 while age < 10:
count controlled	For loop, instructions repeated a **set number of times**	for x=0 to 9 print(x) next x
condition controlled	While loop, instructions continue until a **circumstance is met**	while answer !="Computer Science" answer=input("favourite subject?") end while
comment	**//** or **#**. Ignored by the program, but used to describe code	//this section is a function to square a number #this is the rand function
data types	variety of accepted **information types**	str (alphanumeric) float (decimal) real (interchangeable with float) int(integer, whole number) Bool(Boolean, True or False)

selection	choice/ decision. IF, ELIF, ELSE	if answer ==yes: then
		print("that's correct")
		elif
		print ("that's not correct")
		else
		print ("error")

1.2a Programming fundamentals

variables	A **location in memory** that can be changed and reassigned during the running of the program
constants	A **location in memory** that cannot be changed throughout the running of the program.
operators	Can be either **comparison** or **arithmetic operators**
data types	How data is **represented** in a computer program.
casting	**Altering** the **data type** of the information
string manipulation	**Altering** the **characters in a string**
file handling operations	Using **commands** from within **Python** to **open, read, write** and **close** to interact with a document
array	Fixed length static structure to **store data**. Can be one or two **dimensional**
SQL	**Structured query language**
function	Built-in code, **returns a variable**
procedure	Built-in code, **does not** return a variable

1.6 String handling/operations

key word	definition	example
string length	.length , shows the **number of characters** in the string	subject="ComputerScience" subject.length output 15
substrings	.substring(x,i) returns the characters **starting from the first position**, for the length specified	subject ="ComputerScience" subject.substring(3,5) output "puter"
left	.left(x) returns the characters from the **beginning of the string**	subject ="ComputerScience" subject.left(4) output "Comp"
right	.right(x) returns the characters from the **end of the string**	subject ="ComputerScience" subject.right(4) output "ence"
concatenation	+, used to **combine strings**	print("welcome to the quiz" + name)
uppercase	.upper, will convert all characters in the string to **uppercase**	subject ="ComputerScience" subject.upper4) output "COMPUTERSCIENCE"
lowercase	.lower, will convert all characters in the string to **lowercase**	subject ="ComputerScience" subject.upper4) output "computerscience"
ascii conversion	will return the ASCII **binary chart value** for the character, ACS(…)	ASC(A) output 65

1.1 Programming constructs

abstraction	**removing** unnecessary information from a problem
decomposition	**down** a problem into smaller, more manageable chunks known as **subroutines**
algorithmic thinking	Using **logic** and **reasoning** to solve common problems
designing algorithms	producing **flowcharts** or **pseudocode** and identify common errors through **trace tables**
searching algorithms	**binary** and **linear search algorithms** for locating using repetition or mathematical calculations
sorting algorithms	**bubble, merge** and **insertion sorting algorithms** create solutions using logic and pattern recognition

1.4 Arithmetic operators

+	**addition** of int/float values
-	**subtraction** of int/float values
*****	**multiplication** of int/float values
^	**exponent** (to the power of)
/	**division** of int/float values
MOD	modulus, **outputs the remainder** of a int/float division
DIV	integer division to find the **quotient**
quotient	**integer value** before the decimal point

1.3 Comparison operators

==	**equal to**
!=	**not equal to**
<	**less than**
<=	**less than or equal to**
>	**greater** Built-in code, returns a variable **than**
>=	**greater than or equal to**

Networks and protocols

1.1 Network types

LAN	local area network > used in small areas
WAN	wide area network > group of LANs > larger area
Client - server	client requests > server sends
Peer to Peer	no centralised device > each node acts as server

1.2 Network hardware

Name	description	image
router	forwards packets > provides IP addresses > gateway to Internet	
hub	broadcasts data to all devices	
bridge	connect networks	
switch	intelligent > forward packets > uses MAC address	
server	provides service to client > higher spec than PC	
Wireless Access Point	allows wireless capable devices to access the network	
network interface card	circuit board > allows network connectivity > RJ45 port > onboard/external	
GSM chip	standard (ETSI) > mobile device to phone network (SIM)	
transmission media	medium for data transfer	

1.3 transmission media

Name	description
coaxial cable	high frequency > low latency > TV/phone/internet
Ethernet cable	RJ45 port > internet transmission > standard IEEE (1983)
duplex cable	send & receive > used in Fibre Optic
fibre optic cable	coated glass > data =light > very low latency > high speed

1.4a threats to a network

brute force	trial and error > login info > encryption
denial of service	attack target with traffic > shut down servers
data interception	data loss > transmission > packet sniffer
SQL injection	malicious code > attack database
poor network policy	inadequate rules for clients
malware	group term for malicious software
phishing	social engineering > stealing data > email
people	individual lack of responsibility

1.4b malware

viruses	code hidden in files > self-replicate > delete/modify data
worm	not hidden > spread through email > self-replicating
trojan	disguised as legitimate looking programs > user installed
spyware	monitor user activity > passwords > websites
ransomware	blackmail user to pay > encrypts data as ransom

1.5 packet switching

handshake	agreement by devices to send and receive
source	where the data is coming from
destination	where the data is going to
payload	the information to be transmitted
header	contains data for transmission
packet sequence	total # of packets and reload order

1.6 encryption

encryption	applying algorithm > encoding > unreadable
decryption	Data made readable again
cipher text	encrypted data
Caesar Cipher	Julias Caesar > uses a key to change letters
algorithm	a program to scramble data
private key	kept safe > used for decryption
public key	can be used to encrypt a message
asymmetric	encryption using public and private keys
symmetric	using the same key to encrypt/decrypt

1.7 The four layer model

application	encodes/decodes data
transport	break data into packets
network	adds S/D IP address
data link	transfers data

1.8 Protocols

protocol	rules for transmission
standard	agreed, recognised rules
IPv4	4th IP version, uses 32b
IPv6	6th IP version, used 128b
MAC address	unique identifier on the NIC
ethernet	data transmission LAN
HTTP	Hyper Text Transfer
HTTPs	Hyper Text Transfer (secure)
TCP/IP	Transmission Control / Internet
FTP	File Transfer
IMAP	Internet Mail Access
SMTP	Simple Mail Transfer
POP3	Post Office
VOIP	Voice over Internet

1.9 modes of connection

wired	**cabled > Fibre Optic / Ethernet**
wireless	**radio signals > translated by router**
Bluetooth	**radio waves > short distance**

2.0 terms for PC

standalone
node
device
client
workstation

Algorithms and programming

1.1 Algorithm key terms

input	Data which is inserted **into a system** for processing and/or storage
output	Data which is **sent out** of a system
process	An **action** that takes place in an algorithm
decision	A **yes/no/true/false** decision made in an algorithm
logic	The **aim** of the program

1.2 testing

test data	the data **selected for analysis**
boundary/ extreme	data the is accepted, but is **close to** the required checking area
valid	possible data that the program should **accept** and **process**
erroneous	data that the program can not process and **should not accept**
input validation	a test to ensure the **correct data** type has been inserted
iterative	carried out while a program is **being developed**. ... The process repeats (iterates) until the module works as intended.
final	program is tested **as a whole** to ensure that it functions

1.3 maintainability

maintainability	allows edits and updates of created programs **easily**
debug	locate and resolve an **error**
comments	provide **additional information,** ignored by the program
sensible variable names	pertaining to the **data type or function** of the variable
indentation	formatting to show which lines of code are **linked**

1.4 program errors

syntax error	occurs when rules of programming are **not followed**
logic error	an **inaccuracy in the way the program functions**
run-time error	undetected during compilation, but discovered whilst the program is **running**

1.4 data types

casting	**Changing the data type** within the variable
string	**Alphanumeric** characters
integer	**Whole numbers** (no decimal numbers)
float	**Decimal numbers** (no decimal point limitation)
real	Synonymous with **float**
Boolean	Returns only **true/false**

string operations

string length	**Counts all characters** in the string
substrings	**Continuous characters** within a string
concatenation	**Join strings** and variables
change case	Alternate between **upper/lower case** characters
ASCII conversion	**Convert** from ASCII into binary

1.3 Algorithms

abstraction	The **process** of **removing** unnecessary information
decomposition	The **process** of **breaking down** a problem into **subroutines**
algorithmic thinking	Using **logic** and **reasoning** to solve common problems
designing algorithms	Producing **flowcharts** or **pseudocode**
searching algorithms	**Binary** and **linear search algorithms** for locating data
sorting algorithms	**Bubble, merge** and **insertion** used to create solutions

1.4 Programming fundamentals

variables	A **location in memory** that can be changed and reassigned during the running of the program
constants	A **location in memory** that cannot be changed throughout the running of the program.
operators	Can be either **comparison** or **arithmetic operators**
data types	How data is **represented** in a computer program.
string manipulation	**Altering** the **formatting** of the characters
array	Fixed length static structure to **store data**. One/ two D
SQL	**Structured query language**
function	Performs a task – **does return** a value
procedure	Performs a task – **does not** return a value

1.2 Sorting and searching algorithm

linear search	**One by one**
binary search	Discards **half**
bubble sort	**Swaps** two values
merge sort	Splits into **individual** values
insertion sort	Places the value in the **correct location**

File handling

File handling operations	Using **Python commands** to interact with a document
open	Prepares for data to be written **into file**
close	**Closes and applies** changes
read line	**Return a specified line** from within the file
write line	**Adds** an additional **line to the file**
end of file	Sends the cursor to the **last character**
create a new file	Produce a **blank text** file

Systems Architecture

1.1 Systems Architecture	
CPU	**Central Processing Unit**
Cores	An **individual processor** within a CPU
Cache	Incredibly fast, but very expensive volatile (temporary) memory **close to the CPU**
Clock speed	The **number of FDE cycles** that a CPU can carry out **in one second**
Levels of cache	**L1, L2, L3** – L1 is the fastest and most expensive cache level
Overclocking	Processor running **at a higher speed** than the manufacturer has recommended
CU	**Control** Unit
MAR	Memory **address** register
MDR	Memory **data** register
PC	**Program counter**
Accumulator	Small, fast register, used to keep track of the data **currently** being processed
ALU	**Arithmetic Logic Unit**
FDE cycle	Basis of the **Von Neumann** architecture

1.2 Memory and storage	
RAM	Primary, **volatile** internal storage.
ROM	**Non-volatile** memory, stores the boot up sequence for the computer
Virtual memory	**Short-term**, allocated when the RAM is at capacity
Primary storage	**Volatile** storage, used to temporary hold data
Compression	**Reducing** the file size
Lossy	Data is **removed from** a file to reduce the file size.
Lossless	redundant data is **removed** for sending, then **replaced** upon receipt.
Data capacity	How **much data** the storage type can hold, measured in bits
Secondary Storage	Permanent, **non-volatile** methods of keeping data
Features of storage	Capacity, speed, portability, reliability and cost

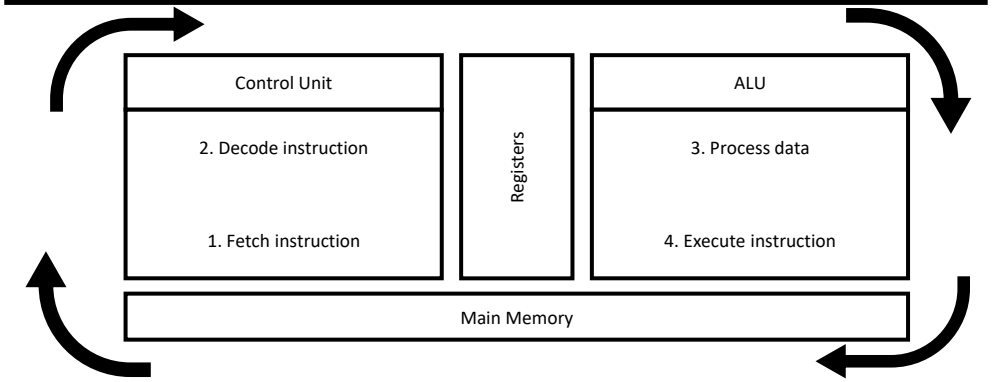
1.3 Systems Software	
Operating System	**controls** all the hardware and software for the PC.
User management	Allocation of an **account**, access rights and security
File management	Naming, allocating to folders, moving **files** and saving
Utility software	Programs on the computer that help the user keep the **computer running**

1.3b Functions of the Operating System	
Functions	The role and responsibilities of the **Operating System**
MUMPS	**An acronym for the functions of the Operating System**
Multi-tasking	Allows more than one **program** to run at the same time
User interface	**Windows, menus, icons and a pointing device (WIMP)** to assist the user
Memory Management	Gives over RAM and CPU **memory** to programs requiring it
Peripheral Management	Allowing **mice/keyboards/printers** to work
Security	Keeping **data protected** from **modification/deletion**

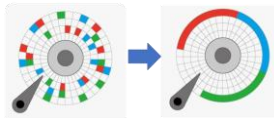
1.4 Computer networks, connections and protocols	
Types of network	**Variations** of network
Client- Server	A network involving the client (user machine) **sending requests** to the server. The server **processes the request** and sends the data
Peer-to-peer	A network where devices are **physically connected to each other** with an Ethernet cable
Network hardware	Devices required to **maintain a network**
The Internet	A **world-wide** collection of hardware
DNS	Domain Naming System **translates** an IP address into a domain name
Hosting	Housing, maintaining and serving **files on a server**
Modes of connection	How **devices are connected**, for example: Wired, wirelessly or Bluetooth
Encryption	The process of **converting** data into code
IP addressing	a **unique string** of numbers separated by full stops
Standards	Standards that allow hardware/software to **interact**
Protocols	A set of **rules** for transmitting data

1.5 Ethical, Legal, Cultural and environmental impacts of digital technology	
Ethical	**morally** right or wrong when discussing computing
Legal	Within or outside the confines of **law**
Cultural	How technology **impacts** on different societies across the globe
Environmental	Discussing how the **environment** is impacted by technology
Privacy	**Regulation, storing and use** of personally identifiable information
Data Protection Act	Updated in **2018 to GDPR**, the law **governs** how people and businesses can use information relating to their clients
Computer Misuse Act	Released in **1990**, the law **governs** use of other peoples computer and outlines the **consequences** of doing so
Copyright, designs and Patents Act	Released in **1988,** the law **governs** who can use the property of others and the information that cannot be used as it belongs to the creator

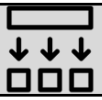
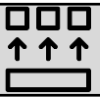
1.6 F-D-E Cycle



1.1 Systems Software	
operating system	The Operating System controls all the **hardware** and **software** for the PC.
memory management	**Allocates memory** to a process when the process requests it and deallocates the memory when the process has terminated
user interface	Provides the user with a WIMP (**Windows, menus, icons and pointer).** Can be either **graphical** or **command line**
multi-tasking	Allows the user to **combine tasks** and processes
peripheral management	Installs the **drivers** for the new hardware so it can be used without error
security	uses password protection to **protect user data** and prevents unauthorized access to programs and user data.
user management	Allocation of an **account, access rights** and **security**
file management	Naming, allocating to folders, moving files and saving
utility software	Programs on the computer that help the user keep the computer running smoothly.

1.1b defragmentation		
	What?	Why?
	Defragmentation > reorganizing the data on the hard drive > related data back together.	 repeated access > data becomes fragmented
		 speed of access

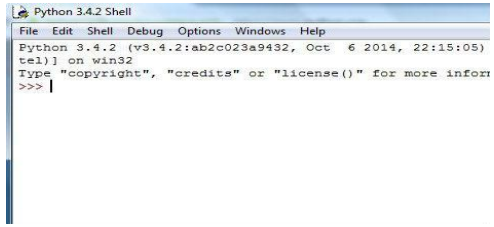
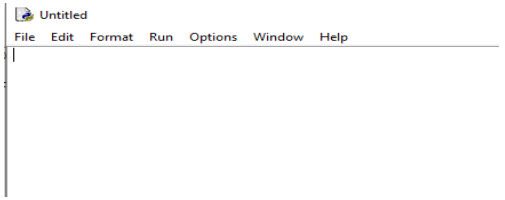
1.2 Ethical, Legal, Cultural and environmental impacts of digital technology	
ethical	**Ethics** relates to what is **morally** right or wrong when discussing computing
legal	Within or outside the confines of law
cultural	How technology **impacts** on different societies across the globe
environmental	Discussing how the environment is impacted by technology
privacy	Laws that relate to the **regulation, storing and use** of personally identifiable information
Data Protection Act	Updated in **2018**, the law **governs** how people and businesses can use information relating to their clients
Computer Misuse Act	Released in **1990**, the law **governs** use of other peoples computer and outlines the **consequences** of doing so
Copyright, designs and Patents Act	Released in **1988,** the law **governs** who can use the property of others and the information that cannot be used as it belongs to the creator
software licences	Understand that software can be **open source** or **proprietary.**

1.3 Algorithms		
abstraction	**decomposition**	**algorithmic thinking**
		

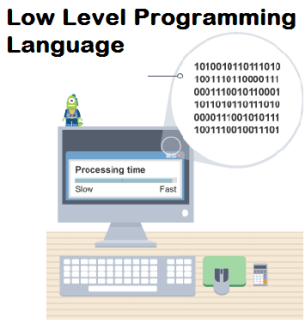
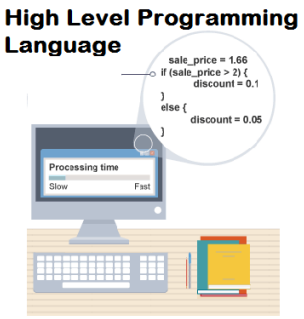
1.3 Algorithms	
abstraction	The **process** of **removing** unnecessary information from a problem, so the solution only focusses on the **essential** information
decomposition	The **process** of **breaking down** a problem into smaller, more manageable chunks known as **subroutines**
algorithmic thinking	Using **logic** and **reasoning** to solve common problems
designing algorithms	Producing **flowcharts** or **pseudocode** and identify common errors through **trace tables**
searching algorithms	**Binary** and **linear search algorithms** for locating a data location using repetition or mathematical calculations
sorting algorithms	**Bubble, merge** and **insertion sorting algorithms** used to create solutions using logic and pattern recognition

1.4 Programming fundamentals	
variables	**A location in memory that can be changed and reassigned during the running of the program**
constants	**A location in memory that cannot be changed throughout the running of the program.**
operators	**Can be either comparison or arithmetic operators**
data types	**How data is represented in a computer program.**
casting	**Altering the data type of the information**
string manipulation	**Altering the case of the characters**
file handling operations	**Using commands from within Python to open, read, write and close to interact with a document**
array	**Fixed length static structure to store data. Can be one or two dimensional**
SQL	**Structured query language**
functions and procedures	**Built-in code**

1.1a interpreter vs compiler

	
Executes one line of code at a time	**Translates** data into **high-level** language
Suitable for **testing**	**Compiles** all lines of code before **executing**
Highlights **errors** in each line, making it easier to fix	Will not run if an **error** exists in the code – can be hard to locate the error
Faster than a **compiler** as it translates and executes at the same time	**Translates** and **executes** entire program
Low **memory** requirement	Requires more **memory**

1.1b high and low level languages

	
written to be applied to the **system architecture**	Close to **human language**
very close to **binary**	often called **pseudocode**
hard for humans to **understand**	easy to modify as it uses ***English-like*** statements
contains **few recognisable** English words	easy to **debug**
fast to **execute**	**portable** – can run on many different machines
examples include: **assembly language, machine code, binary**	examples include: **C++, Java, Pascal, Python, Visual Basic**

1.2 testing

test data	the data selected for analysis
boundary/ extreme	data the is accepted, but is **close to** the required checking area
valid	possible data that the program should **accept** and **process**
erroneous	data that the program can not process and **should not accept**
input validation	a test to ensure the **correct data** type has been inserted
iterative	carried out while a program is **being developed**. ... The process repeats (iterates) until the module works as intended.
final	program is tested **as a whole** to ensure that it functions

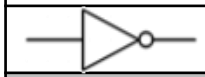
1.3 maintainability

maintainability	allows edits and updates of created programs **easily**
debug	locate and resolve an **error**
comments	provide **additional information,** ignored by the program
sensible variable names	pertaining to the **data type or function** of the variable
indentation	formatting to show which lines of code are **linked**

1.4 program errors

syntax error	occurs when rules of programming are not followed
logic error	an inaccuracy in the way the program functions
run-time error	undetected during compilation, but discovered whilst the program is **running**

1.5 logic gates

Symbol	Logic gate	description
	and	output is 1 when both inputs are 1
	or	output is 1 when 1 of the inputs are 1
	not	output is 1 when input is 0. Contains an invertor
invertor		Circle on the not gate, **inverts** (flips) the input to form the output

OCR ERL Cheatsheet

KNOWITALL NINJA

Commenting

Comment

```
//my note to me
```

Variables

Assignment

```
myAge = 36
```

Constants

```
const pi = 3.14
```

Global Variables

```
global lives = 3
```

Casting

To String

```
str(36)
```

To Integer

```
int("13")
```

To Float

```
float("3.14")
```

To Real

```
real("3.14")
```

To Bool

```
bool("True")
```

Operators

Comparison Operators

```
myAge == 36 //equal
lives != 0  //not equal
health < 1  //less
score > 0   //greater
marks <= 40 //less or
              equal
marks >= 80 //greater
              or equal
```

Arithmetic Operators

```
4 + 5   //add
9 - 6   //subtract
2 * 4   //multiply
5 ^ 3   //exponent
6 / 3   //divide
7 MOD 2 //modulus
8 DIV 3 //quotient
```

Logical Operators

```
age > 18 AND age < 60
hour < 9 OR hour > 17
NOT day == "Sunday"
```

Input/Output

Input

```
pwd = input("Please enter a password")
```

Output

```
print("You have logged in succesfully")
```

Selection

If-Then-Else

```
if hour < 12 then
   print("Good Morning!")
elseif (hour < 18) then
   print("Good Afternoon!")
else
   print("Good Evening!")
endif
```

Switch

```
switch day:
   case 6:
      print("Saturday")
   case 7:
      print("Sunday")
   default:
      print("Weekday")
endswitch
```

Iteration

FOR Loop

```
for i = 0 to 9
  print ("i = " + i)
next i
```

WHILE Loop

```
while password != "Password123"
   password = input("Guess again")
endwhile
```

DO WHILE Loop

```
do
   password = input("Guess again")
while password != "Password123"
```

String Operations

String Length

```
name.length
```

Substrings

```
name.substring(2, 4)
name.left(3)
name.right(5)
```

Concatenation

```
print("Hi" + name)
```

Change Case

```
name.upper
name.lower
```

ASCII Conversion

```
ASC(X)
CHR(75)
```

Arrays

Declaration

```
array score[5]
array ages["Dan","Ali"]
array users[4, 4]
```

Assignment

```
score[0] = 59
users[1,3] = "Ninja01"
```

Length

```
len(score)
```

Random Numbers

Random Numbers

```
i = random(1,9)
r = random(1.1, 7.5)
```

File Handling

Open

```
f = open("data.txt")
```

Close

```
f.close()
```

Read Line

```
f.readLine()
```

Write Line

```
f.writeLine("Hello")
```

End of File

```
while NOT f.endOfFile()
   print f.readLine()
endwhile
```

Create a New File

```
newFile("newdata.txt")
```

Sub Programs

Procedure

```
procedure sum(n1,n2)
  print(n1 + n2)
endprocedure
```

Function

```
function sum(n1,n2)
  return(n1 + n2)
endfunction
```

Call a Procedure

```
sum(8,9)
```

Call a Function

```
result = sum(8,9)
```