Grade 10 Computer Science

Term 3 - 63 complete lesson plans

Ready to teach, in the KICD lesson-plan format | 40-minute lessons | PDF and editable Word

ABOUT THIS PACK

63 complete, ready-to-teach Grade 10 Computer Science lesson plans for Term 3, covering identifiers and operators, control structures, containers and functions.

TOPICS IN THIS PACK

Topic	Name	Lessons
3.3	Identifiers and Operators	18
3.4	Control Structures	17
3.5	Containers	14
3.6	Functions	14

INSIDE EVERY LESSON PLAN

- The specific learning outcomes, key inquiry question, core competencies, values and PCIs, and the resources needed.
- A timed lesson flow: what the teacher does and what the learners do, from introduction to summary.
- The key content (a worked example, notes or a practical with its expected results), then four practice questions with the answers for the teacher.
- The common mistake to watch for, and ideas for learners who need support and for those ready for more.
- A four-level CBC assessment rubric (EE, ME, AE, BE), homework, and a blank reflection space to fill in after teaching.

A TASTE OF LESSON 1

Elementary elements of a program I: keywords, identifiers and literals

By the end of the lesson, the learner should be able to: a) define keywords, identifiers and literals as elements of a computer program; b) give an example of each

> **Keywords, identifiers and literals**
> Every computer program, no matter which language it is written in, is built from a small set of elementary elements. A keyword, or reserved word, is a word that has a special, fixed meaning built into the programming language itself, and cannot be used for anything else, such as naming a variable; examples in Python include if, else, while, for and print.

You also receive this pack as an editable Word file, so you can change any lesson to suit your class before you print it. Lessons follow the order of the KICD curriculum design.

CONTENTS

No.	Lesson	Topic	Page
1	Elementary elements of a program I: keywords, identifiers and literals	3.3 Identifiers and Operators	4
2	Elementary elements of a program II: data types	3.3 Identifiers and Operators	8
3	Variables: declaring and assigning values	3.3 Identifiers and Operators	12
4	Constants: values that do not change	3.3 Identifiers and Operators	16
5	Practical: declaring variables and constants	3.3 Identifiers and Operators	20
6	Input statements: getting data from the user	3.3 Identifiers and Operators	24
7	Output statements: displaying results	3.3 Identifiers and Operators	28
8	Practical: writing a simple program with input and output statements	3.3 Identifiers and Operators	32
9	Arithmetic operators	3.3 Identifiers and Operators	36
10	Practical: using arithmetic operators and operator precedence	3.3 Identifiers and Operators	40
11	Relational (comparison) operators	3.3 Identifiers and Operators	44
12	Logical operators	3.3 Identifiers and Operators	48
13	Assignment operators	3.3 Identifiers and Operators	52
14	Practical: combining operators in expressions	3.3 Identifiers and Operators	56
15	Structured programming: what it means and why it matters	3.3 Identifiers and Operators	60
16	Practical: writing a complete structured program	3.3 Identifiers and Operators	64
17	Common errors with identifiers and operators	3.3 Identifiers and Operators	68
18	Identifiers and operators: review and appreciation	3.3 Identifiers and Operators	72
19	Control structures: an overview	3.4 Control Structures	76
20	Boolean expressions in control structures	3.4 Control Structures	80
21	Selection I: the simple if statement	3.4 Control Structures	84
22	Selection II: the if/else statement	3.4 Control Structures	88
23	Selection III: if/elif/else for multiple branches	3.4 Control Structures	92
24	Practical: nested if statements	3.4 Control Structures	96
25	Practical: writing selection-based programs for real scenarios	3.4 Control Structures	100
26	Iteration I: the while loop	3.4 Control Structures	104
27	Iteration II: the for loop and range()	3.4 Control Structures	108
28	Practical: writing while-loop programs	3.4 Control Structures	112
29	Practical: writing for-loop programs	3.4 Control Structures	116
30	Nested loops	3.4 Control Structures	120
31	Loop control: break and continue	3.4 Control Structures	124

32	Selecting the right control structure for a situation	3.4 Control Structures	128
33	Practical: combining selection and iteration in one program	3.4 Control Structures	132
34	Why control structures matter	3.4 Control Structures	136
35	Control structures: review and appreciation	3.4 Control Structures	140
36	What containers are and why programs need them	3.5 Containers	144
37	Lists I: creating lists and reading items by index	3.5 Containers	148
38	Lists II: changing a list	3.5 Containers	152
39	Looping through lists: totals, averages and the largest value	3.5 Containers	156
40	Tuples: collections that cannot be changed	3.5 Containers	160
41	Dictionaries: storing values by key	3.5 Containers	164
42	Practical: a marks program using a dictionary and a loop	3.5 Containers	168
43	Sets: collections of unique items	3.5 Containers	171
44	Practical: choosing the right container	3.5 Containers	175
45	Searching I: linear search	3.5 Containers	179
46	Searching II: binary search	3.5 Containers	182
47	Sorting I: bubble sort	3.5 Containers	186
48	Sorting II: selection sort and built-in sorting	3.5 Containers	190
49	Containers: review and appreciation	3.5 Containers	194
50	Modular programming: what a function is and why programs use them	3.6 Functions	198
51	Built-in functions	3.6 Functions	202
52	Functions from modules: math and random	3.6 Functions	206
53	Defining and calling your own functions	3.6 Functions	210
54	Parameters and arguments: passing values into a function	3.6 Functions	214
55	Return values: giving a result back	3.6 Functions	218
56	Practical: writing useful functions for real tasks	3.6 Functions	222
57	Scope I: local variables	3.6 Functions	226
58	Scope II: global variables and why to use them carefully	3.6 Functions	230
59	Passing values and lists into functions	3.6 Functions	234
60	Practical: a modular program made of several functions	3.6 Functions	238
61	Functions calling functions, testing and documenting	3.6 Functions	242
62	Why modularity matters	3.6 Functions	246
63	Functions: review and appreciation	3.6 Functions	250

Elementary elements of a program I: keywords, identifiers and literals

Strand 3.0 Software Development (92 Lessons) > Sub-strand 3.3 Identifiers and Operators | Lesson 1 of 18 | 40 minutes

SCHOOL	TEACHER	DATE
CLASS / STREAM	TIME	ROLL (B / G)

SPECIFIC LEARNING OUTCOME(S)

By the end of the lesson, the learner should be able to: a) define keywords, identifiers and literals as elements of a computer program; b) give an example of each

KEY INQUIRY QUESTION(S)

- What is the importance of structured programming language?
- How are variables declared in programming languages?

CORE COMPETENCIES, VALUES AND PCIS IN THIS LESSON

COMPETENCY	**Creativity and imagination**	Learners make up their own examples, patterns or ways of showing the idea.
COMPETENCY	**Learning to learn**	Learners say what they found easy or hard and decide their next step.
COMPETENCY	**Critical thinking and problem solving**	Learners choose a method, check their answers and give reasons for their choices.
VALUE	**Unity**	Learners share materials fairly and work as one team.
VALUE	**Respect**	Learners listen while others explain and accept correction politely.
PCI	**Citizenship education**	Learners connect the work to their rights and duties in the community.
PCI	**Life skills**	Learners practise decision making and working with others.

LEARNING RESOURCES

A computer or laptop with Python installed, or a printed reference of Python syntax, where available | Manila paper and marker pens | Notebooks and pencils

ORGANISATION OF LEARNING

Whole class for introducing new syntax; pairs for hands-on or simulated coding practicals; individual work for tracing and review exercises.

KEY WORDS

keyword | identifier | literal | data type | variable | constant | input | output | operator

THE LESSON AT A GLANCE

Introduction	Teacher models	Guided practice	On their own	Wrap-up
6 min	9 min	10 min	9 min	6 min

LESSON FLOW

Stage	Minutes	The teacher	The learners
Introduction	6	Greet the class and ask: "A sentence in English is built from nouns, verbs, and other kinds of words. What do you think the equivalent 'building blocks' of a line of program code might be called?" Take two or three answers, link them to earlier work, then state today's outcome.	Answer the starter question, recall what they already know and read the outcome aloud.
Step 1: Teacher models	9	Work through the worked example on the next page, thinking aloud. Use a computer or laptop with Python installed, or a printed reference of Python syntax, where available, manila paper and marker pens, notebooks and pencils. Ask learners to predict each step before you show it.	Watch, predict the next step, ask questions and note the key steps in their exercise books.
Step 2: Guided practice	10	Give questions 1 and 2. Move round the groups, ask "How do you know?" and correct errors kindly on the spot.	Work in pairs or small groups, explain each step to a partner and check one another's answers.
Step 3: Independent practice	9	Set questions 3 and 4 for individual work. Mark as you walk round and note who needs help in the next lesson.	Work on their own, then swap books with a partner to check.
Summary and conclusion	6	Ask two or three learners to state the key idea in their own words. Give this exit question and check the answers: "In the code score = 75, identify the identifier and the literal."	Say the key idea in their own words and answer the exit question in the exercise book.

WORKED EXAMPLE

Keywords, identifiers and literals

Every computer program, no matter which language it is written in, is built from a small set of elementary elements. A keyword, or reserved word, is a word that has a special, fixed meaning built into the programming language itself, and cannot be used for anything else, such as naming a variable; examples in Python include if, else, while, for and print.

An identifier is a name a programmer chooses for something in a program, such as a variable, a constant, or, as covered in a later sub-strand, a function, so that it can be referred to and used later in the code; for example, in the Python code age = 15, age is an identifier chosen by the programmer.

A literal is a fixed value written directly into the code exactly as it is, rather than calculated or stored under a name; for example, in age = 15, the value 15 is a literal, specifically a numeric literal, and in name = "Amina", the text "Amina" is a literal, specifically a string literal.

PRACTICE QUESTIONS - WITH ANSWERS FOR THE TEACHER

No.	Question	Answer
1	Define a keyword, in the context of a programming language.	**A word that has a special, fixed meaning built into the programming language itself, and cannot be used for anything else.**
2	Give two examples of Python keywords mentioned in the lesson.	**For example: if, else, while, for, print.**
3	Define an identifier.	**A name a programmer chooses for something in a program, such as a variable, so it can be referred to and used later.**
4	In the code score = 75, identify the identifier and the literal.	**score is the identifier; 75 is the literal.**

WATCH OUT FOR

Learners think any word that appears in code, including identifiers the programmer made up, must be a special keyword. Only the language's own fixed, reserved words are keywords; identifiers are names the programmer is free to choose, within the language's naming rules, covered in the next lesson.

ASSESSMENT RUBRIC - THE FOUR CBC LEVELS

Level	What the learner can do
Exceeds expectations (EE)	Can define keywords, identifiers and literals as elements of a computer program confidently and accurately, explain the method used, and apply the idea to new or unfamiliar problems.
Meets expectations (ME)	Can define keywords, identifiers and literals as elements of a computer program accurately in familiar situations; notices and corrects slips without help.
Approaches expectations (AE)	Can partly define keywords, identifiers and literals as elements of a computer program, or does so with prompts from the teacher or a partner; makes some errors.
Below expectations (BE)	Cannot yet define keywords, identifiers and literals as elements of a computer program without direct help; needs the idea taught again with concrete objects.

SUPPORT AND EXTENSION

Learners who need support	Give a short line of real code, for example age = 15, with each part labelled with a blank for learners to fill in as keyword, identifier, or literal.
Learners ready for more	Explain why a programming language reserves certain words as keywords and does not allow a programmer to use them as identifiers, for example, naming a variable if. (Answer: Keywords have a special, fixed meaning that the language itself needs to correctly understand instructions such as decisions and loops; if a programmer could also use the same word as an identifier, the language would not be able to reliably tell whether the word meant its special built-in instruction or the programmer's own named item, causing confusion and errors)
Inclusion	Seat learners with low vision near the board and read aloud whatever you write; use large, clear, tactile materials; allow extra time; pair with a patient partner; use gestures or Kenyan Sign Language for the key words.

HOMEWORK AND FOLLOW-UP

Redo questions 3 and 4 neatly in the exercise book. Then tell someone at home what you learned about "elementary elements of a program I: keywords, identifiers and literals" and bring one question of your own to ask next lesson.

TEACHER SELF-CHECK

- Did every learner get a turn to handle materials or explain their thinking?
- Did most learners meet the outcome? Write down the names of those who did not.
- Was the pace right — did any step take too long or too short a time?
- Which competency and value did I see learners showing?
- What will I change the next time I teach this lesson?

REFLECTION ON THE LESSON / SELF-REMARKS

Write this after teaching - what worked, what to change, which learners need follow-up.

Elementary elements of a program II: data types

Strand 3.0 Software Development (92 Lessons) > Sub-strand 3.3 Identifiers and Operators | Lesson 2 of 18 | 40 minutes

SCHOOL	TEACHER	DATE
CLASS / STREAM	TIME	ROLL (B / G)

SPECIFIC LEARNING OUTCOME(S)

By the end of the lesson, the learner should be able to: a) name common basic data types used in programming; b) match a real value to its correct data type

KEY INQUIRY QUESTION(S)

- What is the importance of structured programming language?
- How are variables declared in programming languages?

CORE COMPETENCIES, VALUES AND PCIS IN THIS LESSON

COMPETENCY	**Creativity and imagination**	Learners make up their own examples, patterns or ways of showing the idea.
COMPETENCY	**Learning to learn**	Learners say what they found easy or hard and decide their next step.
COMPETENCY	**Critical thinking and problem solving**	Learners choose a method, check their answers and give reasons for their choices.
VALUE	**Unity**	Learners share materials fairly and work as one team.
VALUE	**Respect**	Learners listen while others explain and accept correction politely.
PCI	**Citizenship education**	Learners connect the work to their rights and duties in the community.
PCI	**Life skills**	Learners practise decision making and working with others.

LEARNING RESOURCES

A computer or laptop with Python installed, or a printed reference of Python syntax, where available | Manila paper and marker pens | Notebooks and pencils

ORGANISATION OF LEARNING

Whole class for introducing new syntax; pairs for hands-on or simulated coding practicals; individual work for tracing and review exercises.

KEY WORDS

keyword | identifier | literal | data type | variable | constant | input | output | operator

THE LESSON AT A GLANCE

Introduction	Teacher models	Guided practice	On their own	Wrap-up
6 min	9 min	10 min	9 min	6 min