Computer Science · Revision

Programming Guide

The building blocks of programming, shown in Python. Your exam may use pseudocode or another language, but the ideas are the same.

Variables and data types

```
name = "Sam"      # string
age = 15          # integer
height = 1.68     # float
is_ready = True   # Boolean
print(name, age)
```

Use clear names. A **constant** holds a value that doesn't change while the program runs.

Input, output and casting

```
name = input("Your name: ")
age = int(input("Your age: "))
print("Next year you will be", age + 1)
```

Common slip: `input()` always returns a string. Without `int()`, you can't do maths with it.

Operators

Operator	Meaning	Example
+ − * /	Add, subtract, multiply, divide	7 / 2 gives 3.5
**	Power	2 ** 3 gives 8
//	Integer division	7 // 2 gives 3
%	Remainder (MOD)	7 % 2 gives 1
== != < > <= >=	Comparison	5 == 5 gives True
and or not	Boolean logic	True and False gives False

Selection

```
score = 72
if score >= 70:
    print("Grade A")
elif score >= 50:
    print("Grade B")
else:
    print("Try again")
```

Types of error

Error	What it means	Example
Syntax	Breaks the language's rules, so the code won't run	Missing a colon
Runtime	Crashes while running	Dividing by zero
Logic	Runs, but gives the wrong answer	Using + instead of *

Iteration

```
# Count-controlled: repeats a set number of times
for i in range(3):
    print(i)         # prints 0, 1, 2

# Condition-controlled: repeats while True
count = 3
while count > 0:
    print(count)
    count = count - 1
```

Common slip: `range(3)` stops before 3. Forgetting to change the variable in a `while` loop makes it run forever.

Lists

```
marks = [55, 80, 67]
print(marks[0])      # 55: indexes start at 0
marks.append(90)     # add an item
print(len(marks))    # 4
for mark in marks:
    print(mark)
```

Strings

```
word = "Python"
print(len(word))     # 6
print(word[0])       # P
print(word[0:3])     # Pyt
print(word.upper())  # PYTHON
```

Functions

```
def area(width, height):
    return width * height

print(area(3, 4))    # 12
```

Why use them? They avoid repeated code and make programs easier to read, test and change.

Files

```
with open("scores.txt", "a") as file:
    file.write("72\n")

with open("scores.txt", "r") as file:
    for line in file:
        print(line.strip())
```

`"r"` reads, `"w"` overwrites and `"a"` adds to the end.