COM4013 Introduction to Software Development

- Week 2
- Variables, Operators & Conditions

- *Umar Arif – u.arif@leedstrinity.ac.uk*

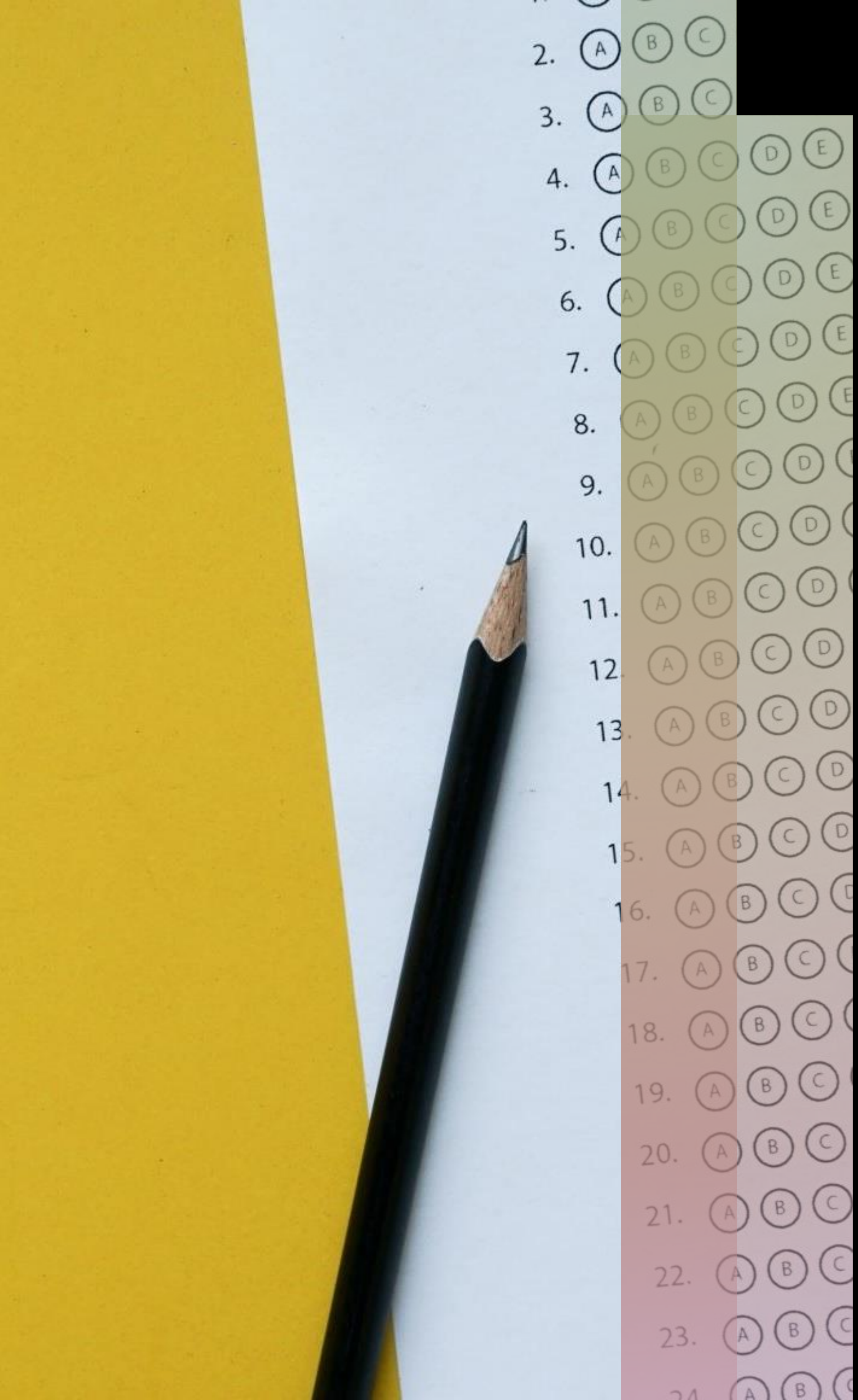
Numeric Types

- Programs often work with numbers: for counting, measurements, cash values, etc.
- Python lets you use several different types of number
  - Integers: whole numbers with no decimal point (e.g., 5, 1265)
    - int : is a Python type most often used for integers (whole numbers)
  - Floating point numbers: with a decimal point (e.g., 3.14, 20.0)
    - float : medium precision, good for general use
  - Complex numbers:
    - represents complex numbers (numbers with both real and imaginary parts).
    - This module will not cover the usage of these. But I thought it would be interesting to note

String Type

- Python also has a type to hold text:
  - String : a sequence of characters
  - Strings are represented in Python surrounded by either single or double quotes:
  - string = "Hello World!" // Crappy variable name

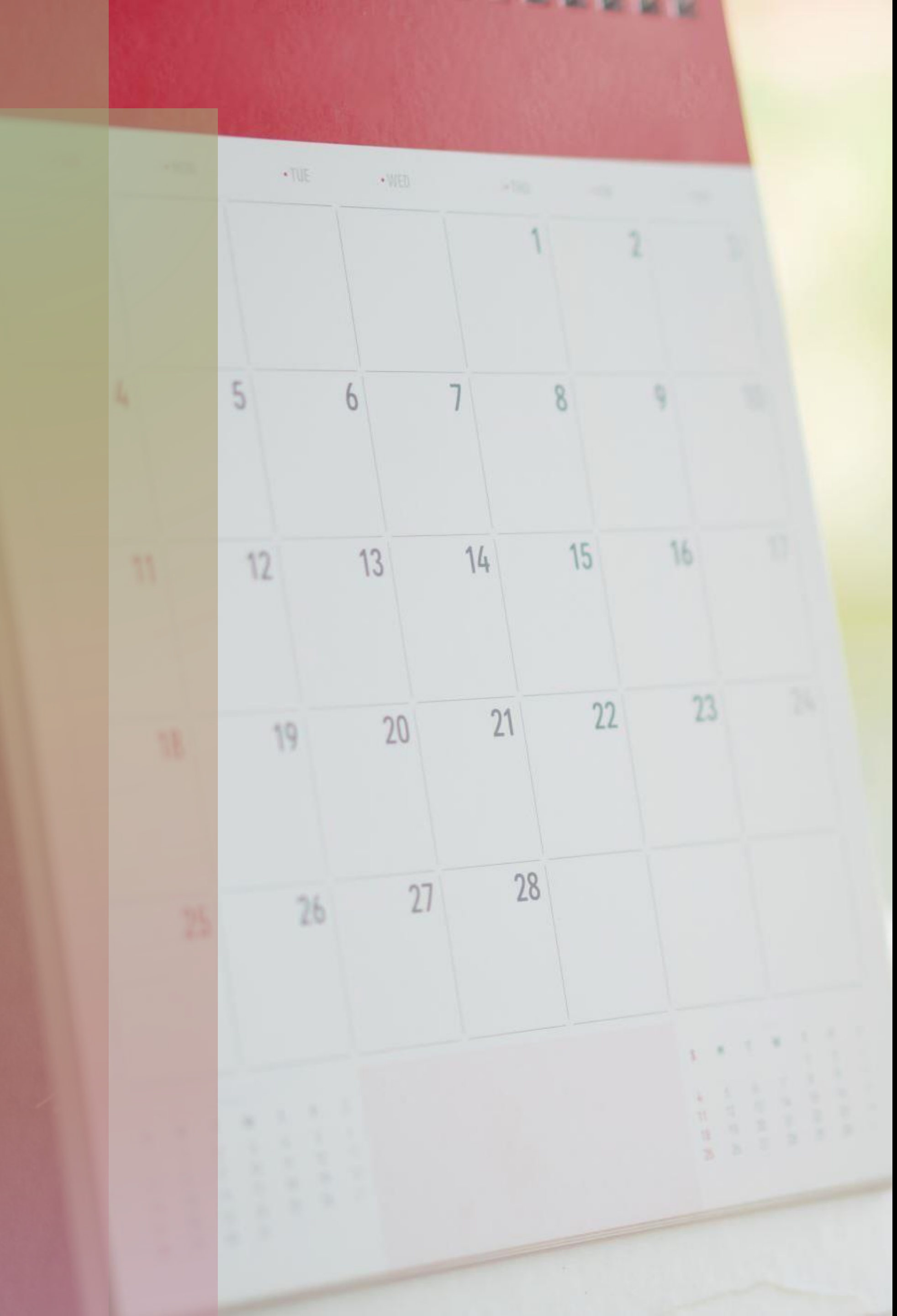
- As I have previously mentioned, Python does not have a character type
  - A String with a length of 1 could be used to represent a Char
  - string = 'H'
  - Most languages use single quotation marks for characters. Use this convention to help you to possibly differentiate them from Strings in your code



Variables: Declaration

- A variable is like a box with a name, that holds values of a given type
  - Whenever you want a new variable, you must declare it
  - You do not need to explicitly declare a variable type in Python. It will infer the type from what you have written

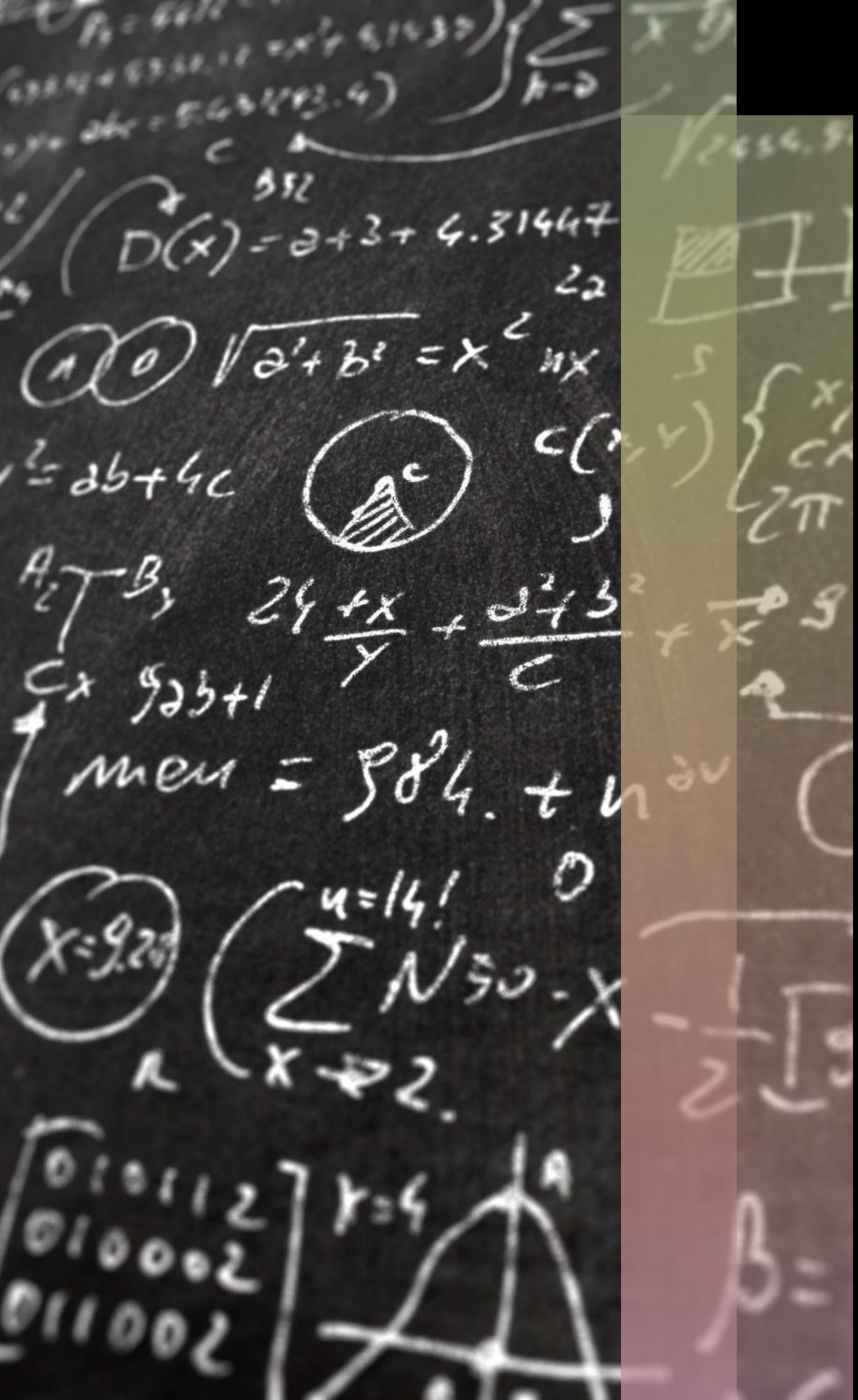
```
numberSides  = 5  # Python knows that this is an int
sideLength   = 10.0
shapeName    = 'Triangle'
```



Variables: Declaration

- Names can contain letters, digits and the underscore character ('_')
  - Names cannot contain spaces and cannot begin wit h a digit
- Invalid variable names:
  - Side Length
  - 3DShape

- Don’t be afraid to create new variables
- You can create variables at almost any point in your program.



Variable Names: Readability

- Last week we saw that names in Python are case sensitive
  - But we didn't set any rules about how to choose variable names
- As programs get larger it is important that they are easy to read
  - I will often focus on readability

- To help readability use the same conventions in all your code
- The most common convention for variable names is:
  - Lower case letter for first word, upper case for subsequent words
  - Words all together, no underscores

```
numberSides
isBalanced
```

- An alternative you might see in some places (I prefer above):

```
int number_sides
```

Variable Legality

Exercise: Which variables are legal in Python?

- 2ndPlace
- counter
- user_name
- TotalAmount
- True
- _private_var
- temp123
- Counter
- @special_char
- For
- my variable

3 mins

Good Variable Names

- Even more important is to choose sensible variable names
  - Here are some good variable names:

```
numberSides
studentID
```

  - And some poor ones:

```
j5
Thingy
```

- A good variable name is often two strong words together:
  - numberSides, currentHP, overallGrade
- Very occasionally simple variable names are OK
  - E.g., single letters for math formulas (e.g., x = y + z)

Comments

- To further help readability we can add comments to our code
  - Use # to make the rest of the line a comment
  - Use (triple quotes) ''' Comment ''' for a block comment over several lines

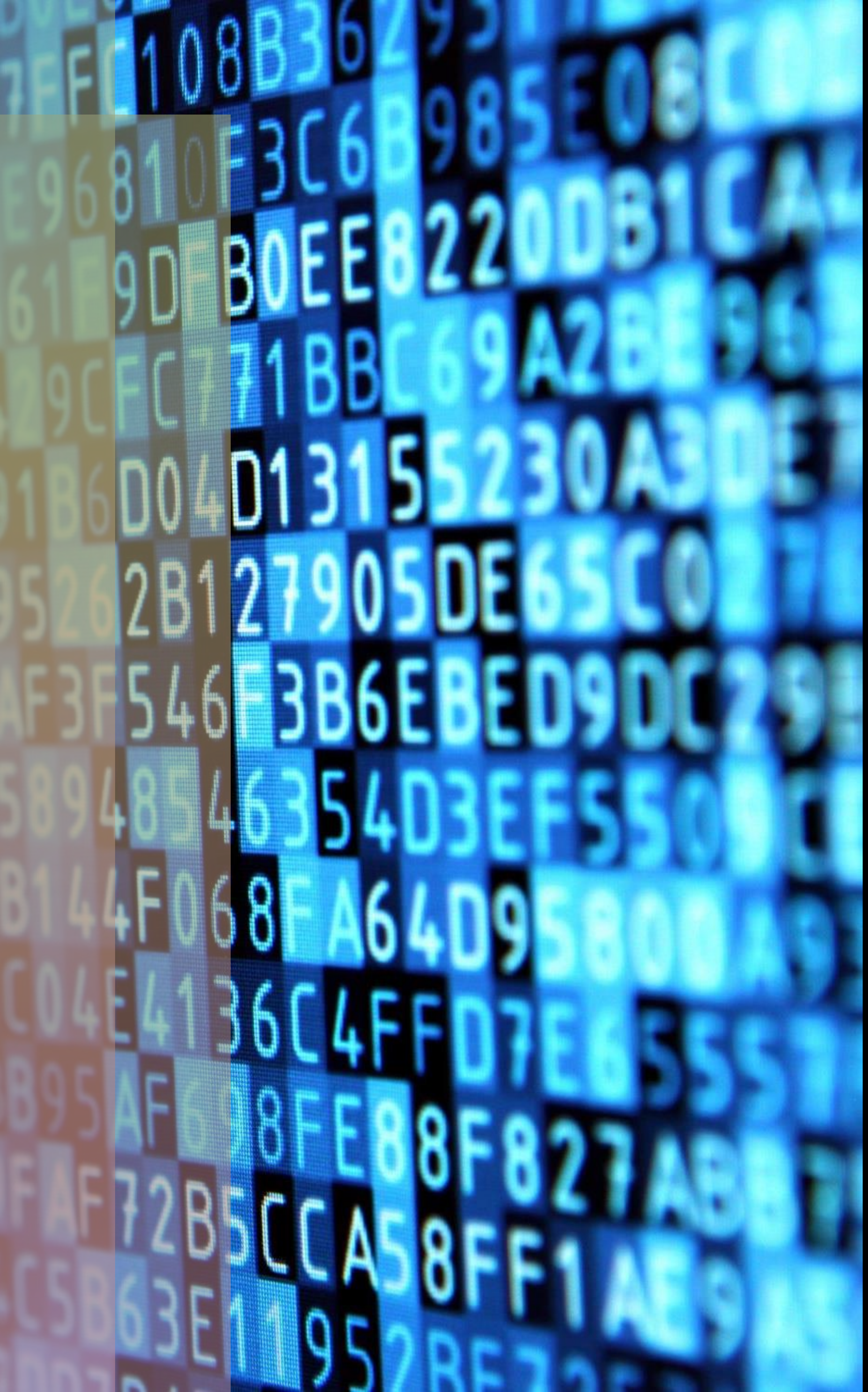
- Some good commenting:

```
''' Collect all variables for student details here,
    their values will be extracted from the database '''
studentID = '2122345' # Student ID format "1234567"
overallGrade = 'A' # Average of this year's marks so far
```

- Some poor commenting (1st comment vague, 2nd too obvious):

```
# Do the problem
```

- `A = X + Y  # Add X and Y together and put it in A`



Comments

- Comments should be concise and helpful
  - Write comments that give an overview of small sections of code
  - Take special care to comment unusual or tricky code
  - Don’t comment obvious things
  - Especially don’t just write the Python in English (as shown on last slide)

You have not written good code unless it is well-commented

- Even if your program works, comments are needed for others to understand how it works
  - Most programming is done in teams
- Your commenting will be marked in your assignment!

Good Comments

Exercise: Write comments for this code

```python
length = 10
width = 5
area = length * width
print("The area of the rectangle is:", area)
```

```python
n = int(input("Enter a positive integer: "))
factorial = 1

for i in range(1, n + 1):
    factorial *= I

print(f"The factorial of {n} is {factorial}")
```

Variables: Initialisation

- When declaring a variable, we can also **initialise** it on the same line
    - Generally, we give the variable an initial value:
    - `numberSides = 5`

- In Python we can also declare a variable, without intialising it to a value.
    - `username = None`
    - You can assign None to indicate no value
    - In Python, using an uninitialized variable will not crash the program, but it will raise a **NameError** exception
    - `print(username)` # This will print 'None'
    - We can initialise a value to the variable later, as can be seen below.

    -
      ```
      username = input("Enter your username: ")
      print(username)
      ```
    - Unless it's an input or container, intialise the variable on the same line

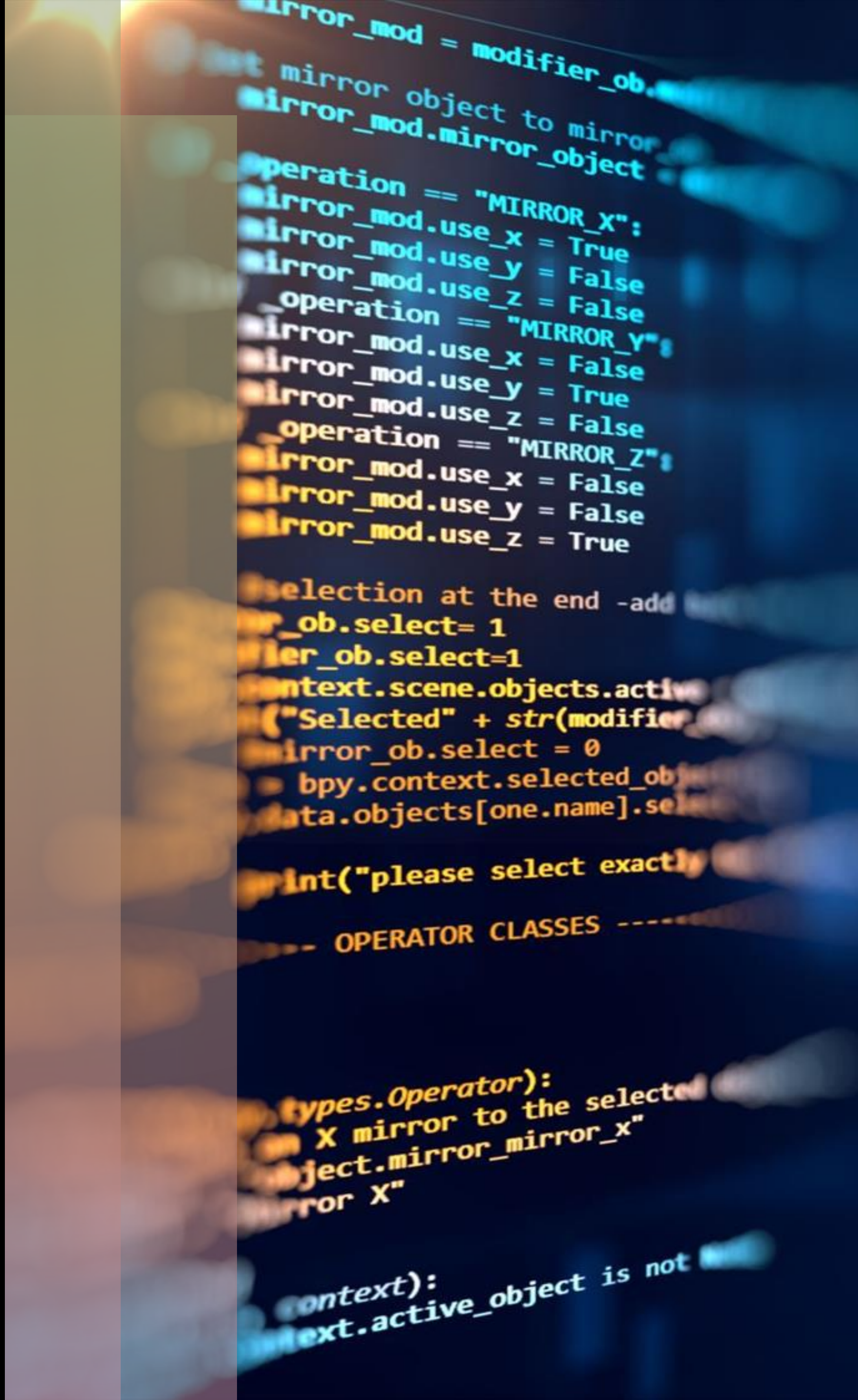
Variables: Assignment

- We can set or change the value of a variable after its declaration:
  - `sideLength = 8.2`
  - This is called assignment; we **assign** 8.2 to the variable

- Clearly assignment is very similar to initialisation
  - Except initialisation occurs on the same line as the declaration:

- `sideLength = 5.4` # Declare and initialize to 5.4
- `sideLength = 8.2` # Assign 8.2, overwriting the previous value

- The difference between assignment and initialisation is less important until you meet some much more advanced topics
  - However, you should remember the basic difference



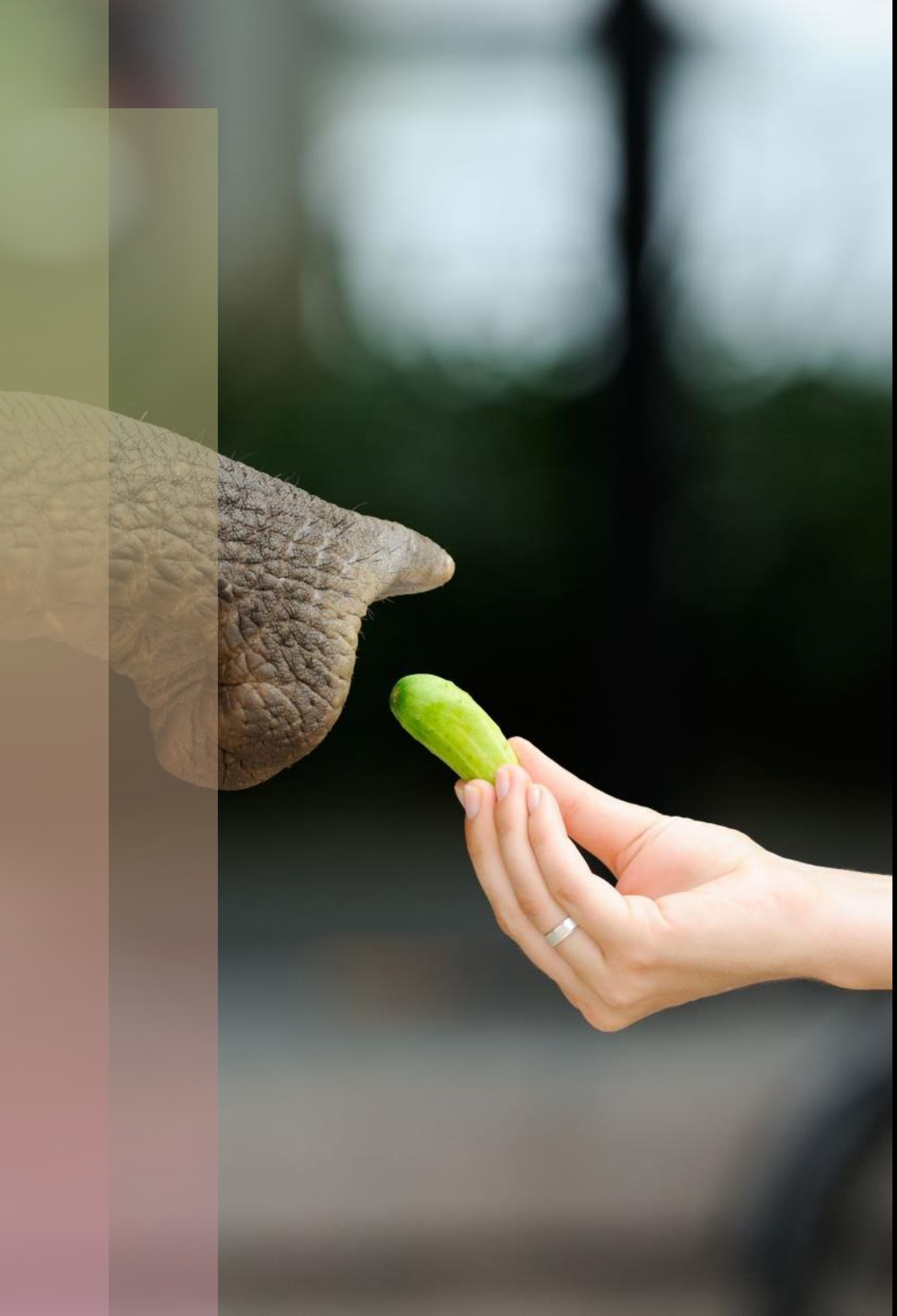
Let's Program in Silence

- **Exercise:** Put in your headphones and do these small tasks. Submit the code you have written to our Teams area.

10 mins

1. Declare a variable name and initialise it with your first name.
2. Declare a variable age and initialise it with your age as an integer.
3. Declare a variable city without initialising it.
4. Assign the value "New York" to the city variable.
5. Calculate the product of your age and the length of your name and store it in a variable called result.
6. Print the values of name, age, city, and result.
7. Try to print the value of an uninitialised variable country. Observe what happens.
8. Assign the value "USA" to the country variable.
9. Print the value of country.
10. Use the input() function to ask the user for their favorite color and store it in a variable called color.
11. Print a message that includes the user's favorite color.

POST: (200 words) Summarise the difference between variable assignment and initialisation in your own words.



Conversion/Casting

- In Python, you might encounter situations where you want to assign or initialise variables with different types, for example:
- We generally use this for user input.
  - UserAge = int(input("What is your age?"))

```
userAge = input("What is your age")
print(type(userAge)) # Prints string type
userAge = int(userAge)
print(type(userAge)) # Prints integer type
```

Using round() is recommended when converting from float to integer to limit decimal places and minimise accuracy loss. Casting floats to integers in Python ALWAYS rounds down the value to the nearest integer.

Operators

- A computer is just a calculator with a big ego
  - So of course, it can perform calculations
- All the usual arithmetic symbols are available
  - They are called operators
  - The basic operators: + - / (division) * (multiplication)
  - Another useful operator is modulus (remainder): %
- You can use operators almost anywhere, especially in assignments and initialisation:
  - heightInches = heightCm * 2.54
  - clock12hr = clock24hr % 12
- [Imagine it is 20:00, what is 20 / 12? Answer, 1 with 8 left over. 20 % 12 gives you the 8 remainder. I.e., 20:00 is 8:00 on a 12hr clock]

Updating Variables with Assignment

We have seen that assignment is used to set the value of a variable, overwriting what was already in it

This is frequently used to increase or decrease variable values:

```
numStudents = numStudents + 1;   # Increase numStudents by 1
discount = price * 10.0 / 100.0; # 10% of price
price = price - discount;        # Decrease price
```

Be careful if you have a mathematical mind:

x = x + 1

- In math's this means “there is a value x that is equal to x + 1", which is impossible
- But in programming it means "x becomes its old value + 1"

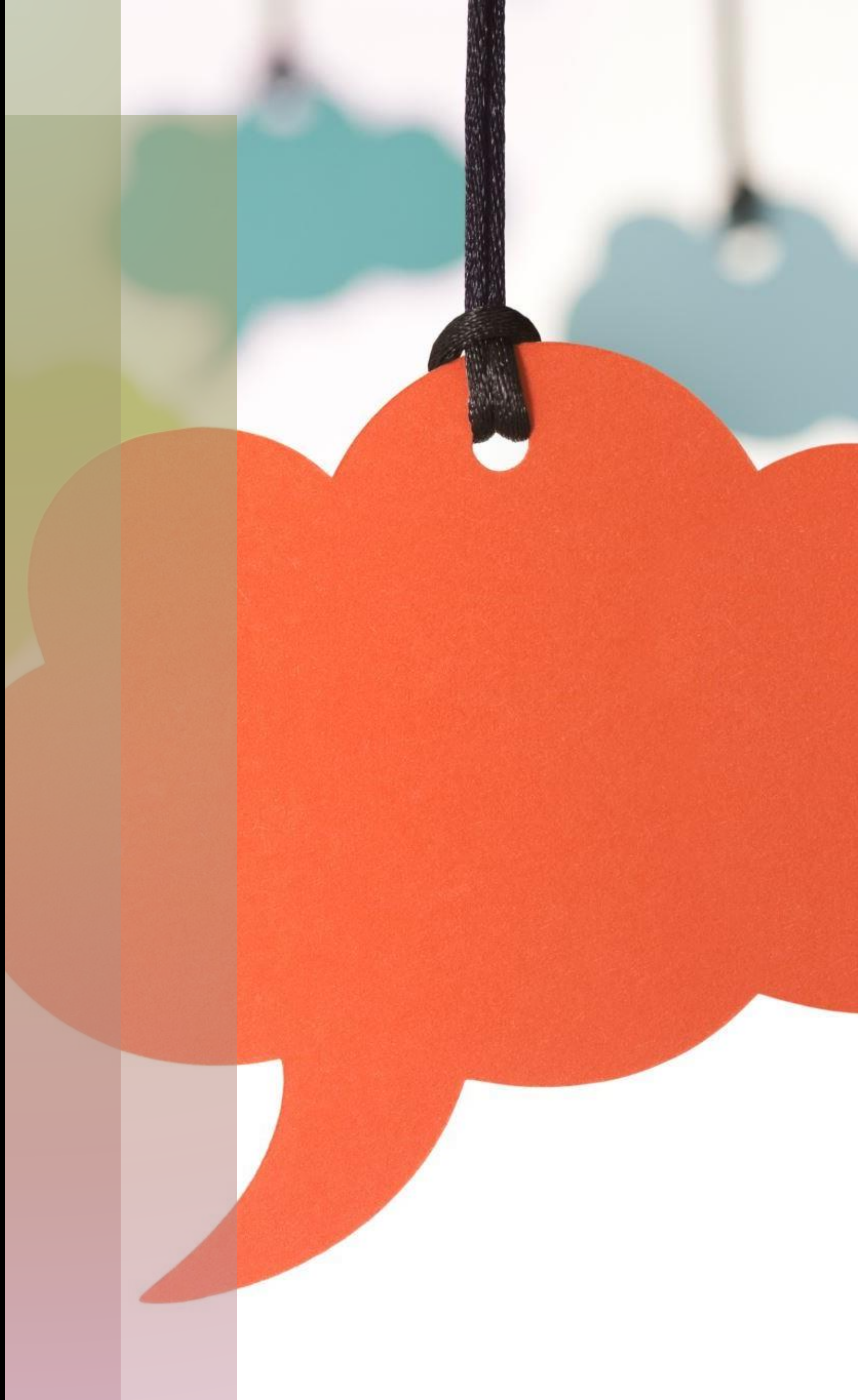
Common Mistake: Missing Assignment

- Changing variables in this way leads to a common mistake
  - I want to increase totalMark by 25, so I write this:

    ```
    totalMark + 25
    ```
  - This asks the computer to calculate totalMark + 25...
  - ...then does nothing at all with the result, which is thrown away...

- We need an assignment statement to make a variable change
  - This is the correct line:

    ```
    totalMark = totalMark + 25
    ```
  - Why?



The Boolean Type

- The Boolean type simply holds the value True or False
  - `giveDiscount = True`
  - Note that you do not use quotes around true or false

- We use Booleans/Bools for choices in our program
- When programming we need **comparisons** and **conditions**, introduced next

- Here's an example of a **comparison**:
  - `bool player1Wins = player1Score > player2Score`
  - Here player1Wins will become True if player 1 score is higher
  - It will become False otherwise
  - This is a nice way to use bools with if statements.

Comparisons & Conditions

- Python has several comparison operators:

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

- These are used on variables or values like the arithmetic operators:
  - `tooExpensive = price > 150.00`
  - `isDraw = player1Score == player2Score`
  - Be very careful to use double `==` for comparisons
  - Single `=` is for assignment, understand the difference
- Boolean variable names are best chosen so they sound like they will be true or false (using words like is... has...).

The ‘if’ Statement

- The if statement allows you to execute an extra piece of program code if a certain condition is met
- The condition must be a Boolean value
- Either a Boolean variable or more usually some kind comparison
- A well written if statement reads quite simply
  - Here’s an example, do you understand what it does?

```
if (playerScore > computerScore):
    print( "You Win!)
    print( "Do you want to play again?" )
```

String Type

- An if statement always has this form:

```
if (condition):
    additional code

continue here
```

- When the program reaches the if statement, it tests if the condition is true or false
- If it is true, the additional code in the scope is executed before the program continues as normal after the indentation
- If it is false, the additional code is skipped, and the program continues after the indentation

If Statement: Example

```
price = 150.00
print("Do you want a discount?")
yesOrNo = input()

if yesOrNo == "y":
  price -= 20.00
  print("OK, if only life were really so
  simple...")

print("Total Price =", price)
```

Avoid declaring new variables inside if statements for now. We will cover the reasons for that in a few weeks

Improve this code?

```python
price = 150.00
print( "Do you want a discount (y/n)?" )
yesOrNo = input()

if yesOrNo == "y":
  price -= 20.00
  print("If only life was this easy.")

print( "Total Price =", price)
```

5 minutes

Exercise: How could this code be improved?

HINT: Input validation, merge steps.

A Possible Solution:

```python
price = 150.00

# Remove leading spaces and lower the input case - input validation
yesOrNo = input("Do you want a discount (y/n)? ").strip().lower()

if yesOrNo == "y":
   price -= 20.00
   print("OK, if only life were really so simple...")

print("Total Price =£", price)
```

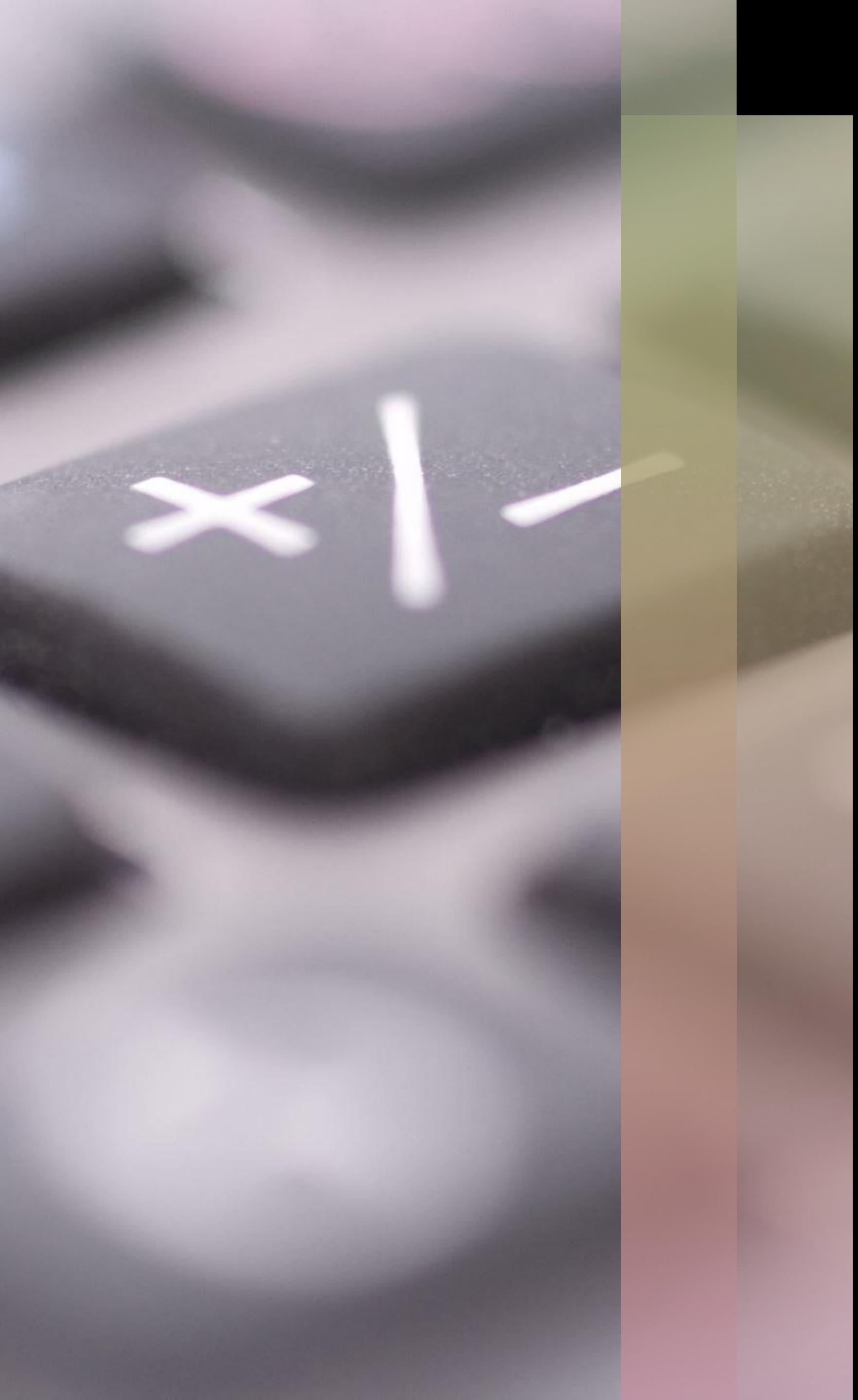
The ‘else’ Statement

An if statement can optionally have an else statement:

```
if (condition):
    code if condition is true
else:
    code if condition is false

continue here
```

- The program tests if the condition is true or false
- If it is true it executes the first block of code (red), otherwise it executes the second block of code (green).
- *Only one block of code will be executed*



The ‘if-else’ Statement: Example

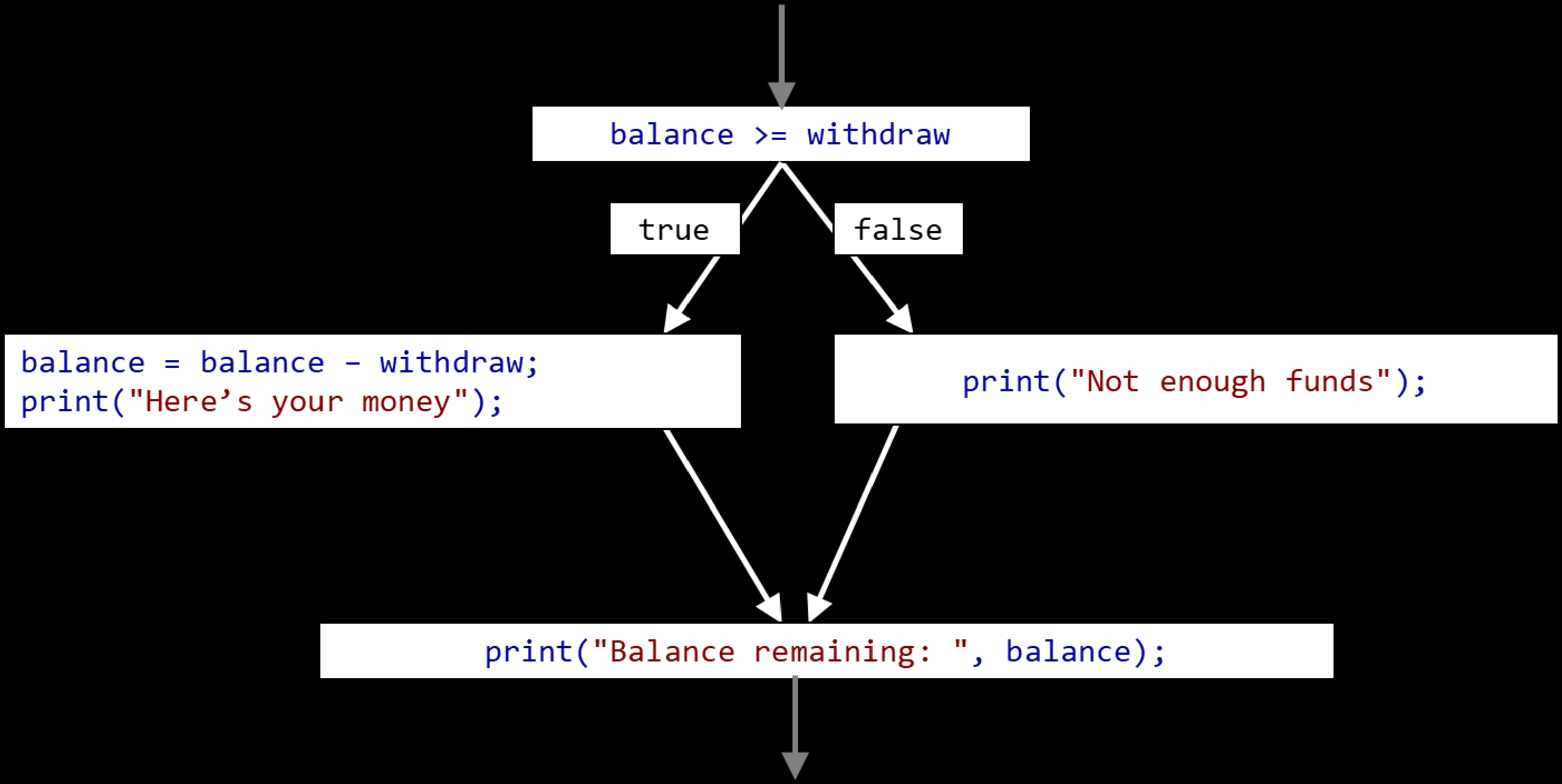
Here’s an example of an if-else statement:

```
# Assume we already have the integers, withdraw and balance

if (balance >= withdraw):
    balance -= withdraw
    print("Here's your money")
else:
    print("Not enough funds")
    print("Balance remaining: ", balance)
```

The ‘if-else’ Statement Visualised

Flow of the `if-else` statement example:



balance >= withdraw
true
false
balance = balance – withdraw;
print("Here’s your money");
print("Not enough funds");
print("Balance remaining: ", balance);

Summary

- We have covered several topics today:
- Types: integer, floating point, string, and Boolean
- Variables: declaration, initialisation and assignment
- Variable naming conventions
- Commenting
- Arithmetic operators
- Comparison operators
- The if and if-else statements / conditions

- That’s a lot of new topics in one session. However, you will have considerable practice to make sure you have understood it all.

What is Scope in Software Devlopment?

Compose a 250-word answer to differentiate between block scope, function scope, class scope, and global scope in programming. Provide an example of situations where each scope might be used.

(The examples will not be counted towards the word count). Be sensible with any example code.