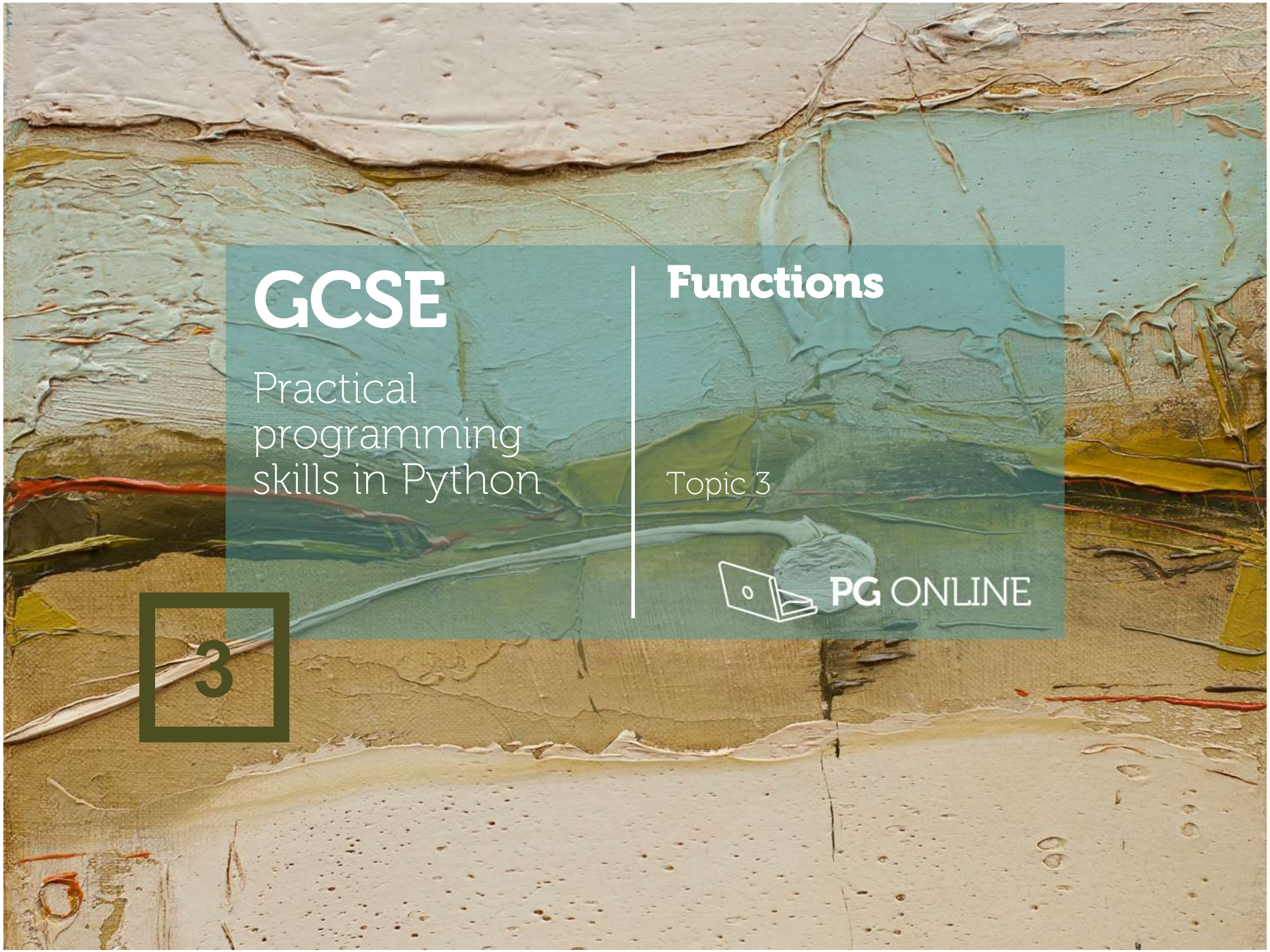

GCSE
Practical programming skills in Python
Functions
Topic 3
PG ONLINE
3

Objectives

- Be able to use decomposition to help solve a larger problem
- Be able to use functions (a type of subroutine) to help make your programs easier to create and more efficient

Starter

- In Python, write a list of 4 or 5 activities you do as part of your morning routine
  - For example:

```
brushYourTeeth()
getDressed()
```

- Note that Python requires the empty parentheses after the function name, when you call or define the function
  - Python does not distinguish between functions and procedures – they are the same thing

Starter

- Adapt your program to look more like this:
  (complete your program for all of your tasks)

```
def brushYourTeeth():
    print("Get toothpaste")
    print("Get toothbrush")
    print("Put toothpaste on toothbrush")
    # etc.
# MAIN PROGRAM
brushYourTeeth()
getDressed()
```

Functions

- A function is a block of code that you can reuse
- You start by **defining** the function(`def`)
- You **call** the function at the bottom
- This technique is called **decomposition**
- It's helpful for **planning**
    1. Take a bigger task (like getting ready in the morning)
    2. Break it down into small or medium-sized tasks
    3. Code each task individually as a procedure

Functions

- By separating the detail from the big picture, you can more easily swap instructions around
  - Do you normally brush your teeth, then get dressed? Or the other way around?
- Simply swap the function calls around:

```
brushYourTeeth()
getDressed()
```

```
getDressed()
brushYourTeeth()
```

Functions

- Functions are also really good for code you will repeat often
  - Think of a task you do often, e.g. check your phone

```
def checkPhone():
    print("Check Facebook")
    print("Check Twitter")
    print("Check Instagram")
    print("Check Reddit")
```

- You can now insert function calls for this into your program, between each task

Worksheet 3a

- Complete **Question 1**

Rock Paper Scissors

- You probably know the rules of Rock Paper Scissors
- If you wanted to make a game you could play against the computer – what tasks (or procedures) would you need?
- Individually, or in pairs, try to write a list of them
  - Hint: Break it down into three steps

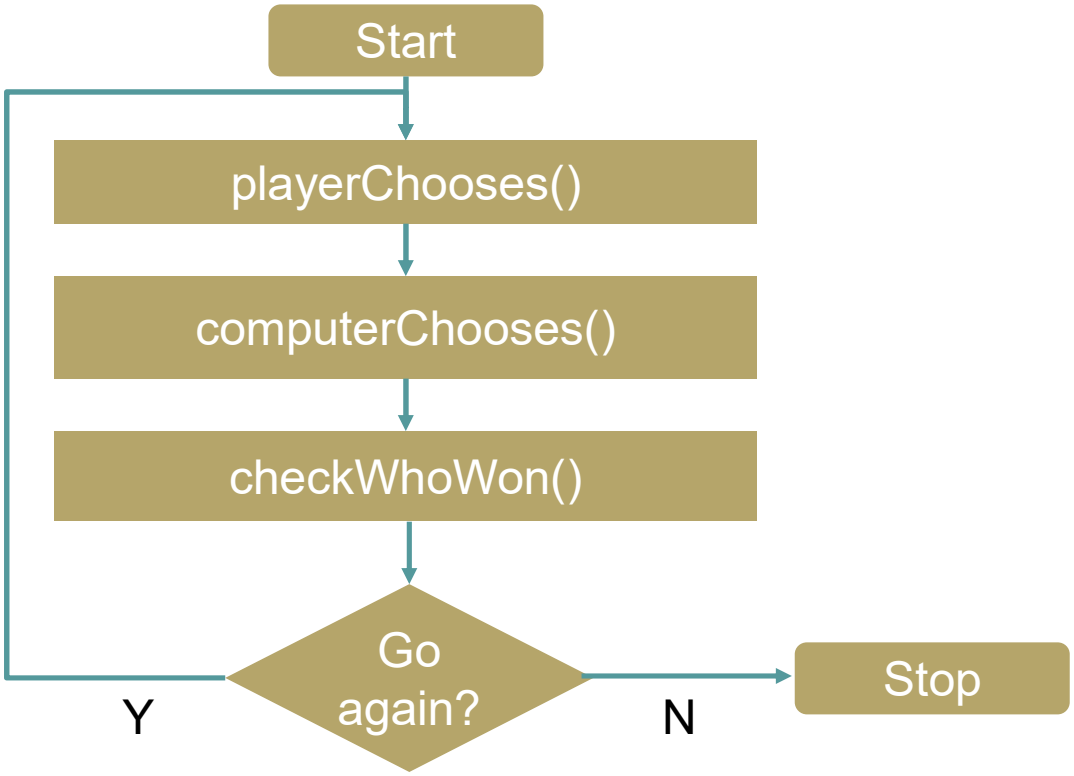
Rock Paper Scissors

- Your answer might look like this:

```
playerChooses()
computerChooses()
checkWhoWon()
```

Rock Paper Scissors

- You could create a flow chart using these steps:



Tic – Tac – Toe

- You probably also know the rules of Tic-Tac-Toe (otherwise known as Noughts & Crosses)
  - What tasks (or functions) would you need to make a game you could play against the computer?
- Individually, or in pairs, write a list of functions

Tic – Tac – Toe

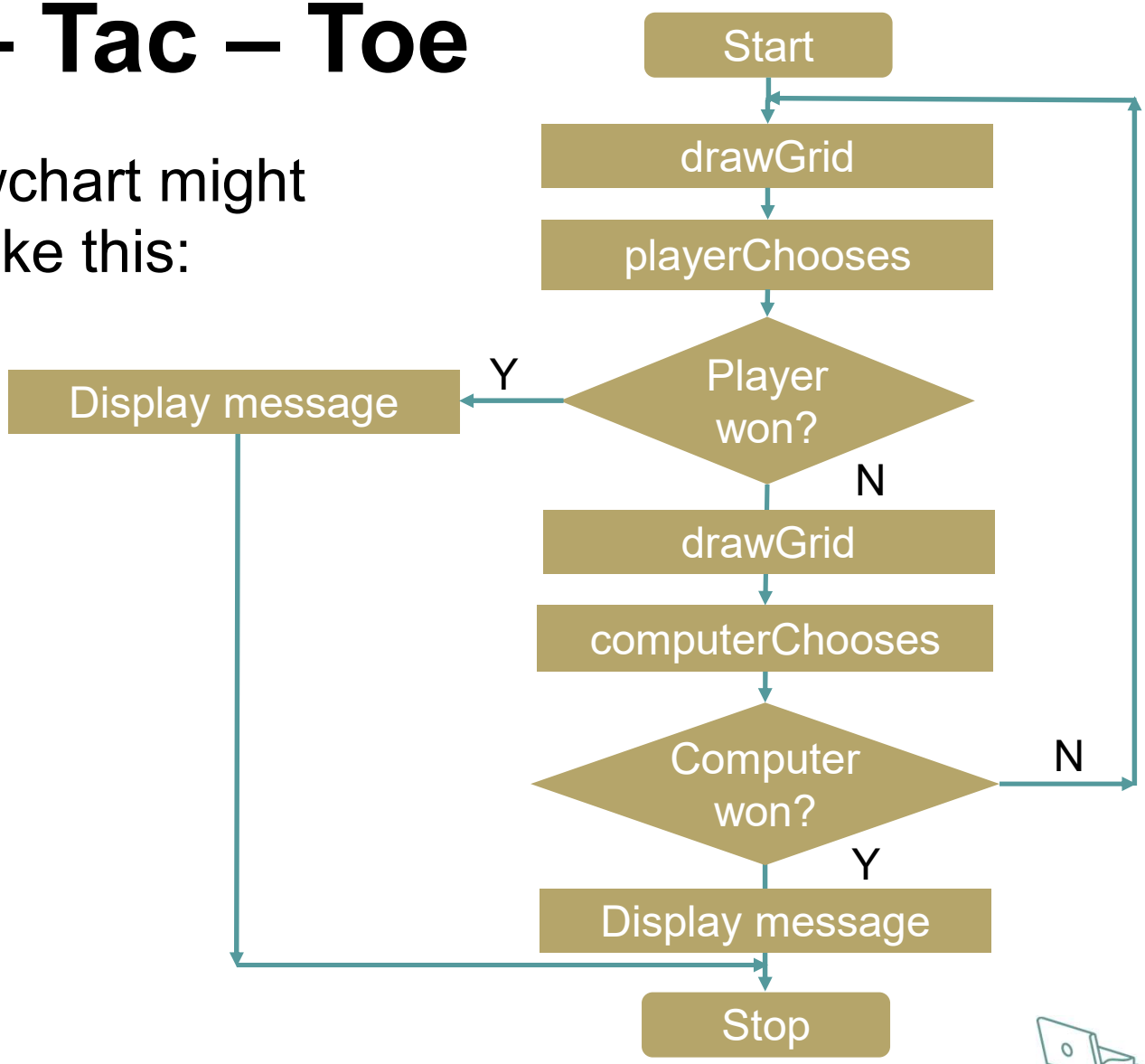
- Your answer might look like this:

```
drawGrid()
playerChooses()
computerChooses()
checkIfSomeoneWon()
```

- Try creating a flowchart

Tic – Tac – Toe

- A flowchart might look like this:



Worksheet 3a

- Complete **Questions 2 and 3**

Conversion tables

- Read the LA Times story about the loss of the $125 million Mars Climate Orbiter (**Worksheet 3b**)
  - Conversion between metric and imperial units is very important!
  - Write a program that will allow a user to convert between a range of different units

Conversion tables

- Start by **defining** a single function that will:
  - ask for a measurement in inches
  - convert this to centimetres (multiply by 2.54)
  - print out the measurement in centimetres
- Test this by putting a **function call** at the bottom of the program

Worksheet 3b

- Complete **Question 1**
- Use the conversion table to complete a range of possible conversions
  - **Extension:**
    Add a menu that will prompt the user for the available options (this could be a function of its own that would call the relevant conversion procedure)

Functions

- A function may return a value

```
def getTotal():
  num1 = int(input("Enter first number: "))
  num2 = int(input("Enter second number: "))
  sum = num1 + num2
  return sum

sum = getTotal()
print("Total:", sum)
```

The return value

- A function may return a value

```
def getTotal():
  num1 = int(input("Enter first number: "))
  num2 = int(input("Enter second number: "))
  sum = num1 + num2
  return sum
# sum and total refer to the same variable
total = getTotal()
print("Total:", total)
```

The return value

- Note that a **function call** needs to store (or assign) the value that has been returned

```
total = getTotal()
```

- The variable used in the main program doesn't need to have the same name as the variable in the function (although it can do)

Returning more than one value

- A function can return more than one value

```
def convertInchToCm():
    inch = int(input("Enter inches: "))
    cm = inch * 2.54
    return cm, "cm"

value, unit = convertInchToCm()
print("Conversion =", value, unit)
```

Worksheet 3b

- Complete **Question 2**

Parameters

- A function might need a value in order to run:

```
def square():
    number = number ** 2
    return(number)
```

- The program doesn't know what `number` is

Parameters

- This function expects to be given a value:

```
def square(number):
    sqnumber = number ** 2
    return(sqnumber)
```

- This is called a **parameter**

Parameters

- The function call needs to include the parameter:

```
def square(number):
    sqnumber = number ** 2
    return(sqnumber)

startNum = 12
answer = square(startNum)
print(startNum, "squared =", answer)
```

Worksheet 3b

- Complete **Questions 3 and 4**

Plenary

- Explain the meaning of these terms:
  - Subroutine
  - Function
  - Call
  - Parameter

Plenary

- Explain the meaning of these terms:
  - Subroutine
    - A separate block of code that can be re-used
  - Function
    - A type of subroutine
  - Call
    - A line of code that runs a subroutine
  - Parameter
    - A value that a subroutine expects to be given when called

Copyright

Licence agreement