First thing’s first

Lets go back to our naughts and crosses board

Write an if statement that checks if the items on the top row meet a winning condition. So the top row are all ‘o’s or all ‘x’s.

First thing's first

Let's create a ticket machine for a cinema

Write an if statement that checks the ages of cinema goers, and display the ticket prices:

- Child (below age of 18): £8
- Adult (18+): £10.95
- Senior (60+): £7.50

Code Nation

Python Fundamentals

Functions

{codenation}®

Learning Objectives

- To understand how functions work
- To write programs with functions

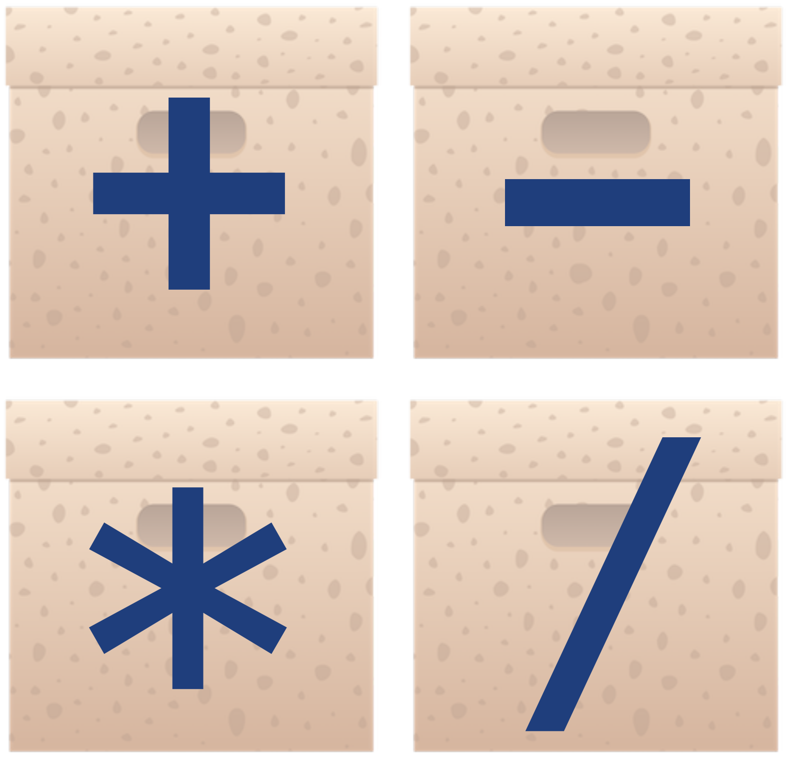
Introducing Functions

Functions let us do the things we need our code to do

We call functions by using their identifiers

{cn}®

They break our code up into small chunks



Separate functions for each operator

Let's take this in

```
def press_grind_beans():
    print("Grinding for 20 seconds")

press_grind_beans()
```

Let's take this in

```
def press_grind_beans():
    print("Grinding for 20 seconds")

press_grind_beans()
```

Declare new function

Let's take this in

```
def press_grind_beans():
    print("Grinding for 20 seconds")

press_grind_beans()
```

Declare new function

Start grinding the coffee

Let's take this in

```
def press_grind_beans():
    print("Grinding for 20 seconds")

press_grind_beans()
```

Declare new function

Start grinding the coffee

Run the function pressGrindBeans

What if we want to print something different based on the status of the coffee grinder?

Let's take this in

```
coffee_is_grinding = False

def press_grind_beans():
    if coffee_is_grinding:
        print('The coffee is grinding')
    else:
        print('The coffee is not grinding')

press_grind_beans()
```

Let's take this in

```
coffee_is_grinding = False

def press_grind_beans():
    if coffee_is_grinding:
        print('The coffee is grinding')
    else:
        print('The coffee is not grinding')

press_grind_beans()
```

Declare new variable with boolean value

Let's take this in

```
coffee_is_grinding = False

def press_grind_beans():
    if coffee_is_grinding:
        print('The coffee is grinding')
    else:
        print('The coffee is not grinding')

press_grind_beans()
```

Declare new function

Let's take this in

```
coffee_is_grinding = False

def press_grind_beans():
    if coffee_is_grinding:
        print('The coffee is grinding')
    else:
        print('The coffee is not grinding')

press_grind_beans()
```

Check if coffee_is_grinding is true

Let's take this in

```
coffee_is_grinding = False

def press_grind_beans():
    if coffee_is_grinding:
        print('The coffee is grinding')
    else:
        print('The coffee is not grinding')

press_grind_beans()
```

Print that it is grinding

Let’s take this in

```
coffee_is_grinding = False

def press_grind_beans():
    if coffee_is_grinding:
        print(‘The coffee is grinding’)
    else:
        print(‘The coffee is not grinding’)

press_grind_beans()
```

Else if coffee_is_grinding is false

Let's take this in

```
coffee_is_grinding = False

def press_grind_beans():
    if coffee_is_grinding:
        print('The coffee is grinding')
    else:
        print('The coffee is not grinding')

press_grind_beans()
```

Print that it is not grinding

Let's take this in

```
coffee_is_grinding = False

def press_grind_beans():
    if coffee_is_grinding:
        print('The coffee is grinding')
    else:
        print('The coffee is not grinding')

press_grind_beans()
```

Run the function press_grind_coffee

Parameters

... these really make functions tick

Parameters give functions their flexibility

They provide the ability to call functions to act on different data inputs

Let's take this in

```
def cash_withdrawal(amount, accnum):
   print('Withdrawing {} from account {}'.format
(amount, accnum))

cash_withdrawal(300, 50449921)
cash_withdrawal(30, 50449921)
cash_withdrawal(200, 50447921)
```

Activity:

Create a function that takes two parameters for a coffee order (size, type of drink) and prints them out in a sentence

Let’s take this in

```
def take_order(size, drink_type):
    print("I’d like a {} {} please".format(size, drink_type))

take_order("Tall","Latte")
```

No longer the point of no return

We can call on functions to do a job and when they've done it, they can **return** the result

Let's take this in

```
def add_up(num1, num2):
    return num1 + num2

add_up(7,3)
print(add_up(2,5))
```

Let's take this in

```
def add_up(num1, num2):
    return num1 + num2

add_up(7,3)
print(add_up(2,5))
```

Add up two numbers and return the answer

Let's take this in

```
def add_up(num1, num2):
    return num1 + num2

add_up(7,3)
print(add_up(2,5))
```

Add up two numbers and return the answer

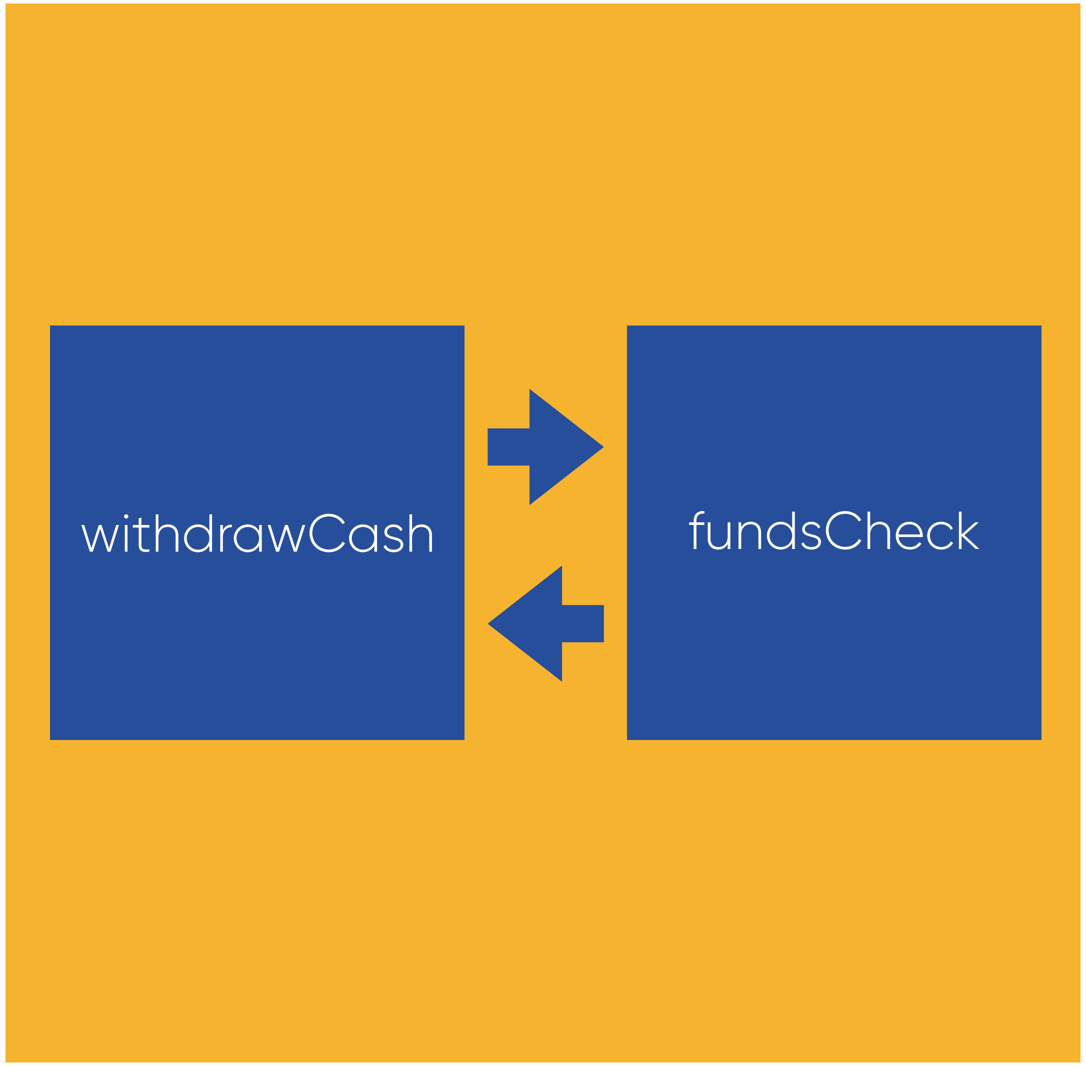
Add up two numbers, return the answer, and then print the result

So, you see...

one function might call another function

and use the result of that function to achieve its goal

For example, a cash machine might have something like ...



Does customer have enough funds requested?

Check and return result to withdrawCash

Let’s take this in

```
def multiply_by_nine_fifths(celsius):
    return celsius * (9/5)

def get_fahrenheit(celsius):
    return multiply_by_nine_fifths(celsius) + 32

print("The temperature is {}°F".format(get_fahrenheit(15)))
```

Functions

Functions are written to perform a task.

Functions are written to perform a task.

Functions take data, perform a set of tasks on the data, and then return the result.

Functions are written to perform a task.

Functions take data, perform a set of tasks on the data, and then return the result.

We can define parameters to be used when calling the function.

Functions are written to perform a task.

Functions take data, perform a set of tasks on the data, and then return the result.

We can define parameters to be used when calling the function.

When calling a function, we can pass in arguments, which will set the function's parameters.

Functions are written to perform a task.

Functions take data, perform a set of tasks on the data, and then return the result.

We can define parameters to be used when calling the function.

When calling a function, we can pass in arguments, which will set the function's parameters.

We can use return to return the result of a function which allows us to call functions anywhere, even inside other functions.

Learning Objectives

- To understand how functions work
- To write programs with functions
- To write programs with all three types of functions

{codenation}®

Activity(1):

Here's an example of a function that includes a parameter. Parameters are responsible for functions being able to work on different data inputs. Edit the snippet below to include two parameters.

```
def take_order(topping):
    print('Pizza with {}'.format(topping))

take_order(“pineapple")
```

Activity(2):

Cash machine time. Let's create one that :

} Takes an input of pin number and amount
} Prints dispensing cash if the pin number is correct and there's enough money to withdraw
} Displays the new bank balance

Be creative!