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

PYTHON FUNDAMENTALS

Functions

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

They break our code up into small chunks

What does a function look like?

Well, you already know.

```
print()
                    lower()
len()
                              input()
```

…and many more….

These are just some examples of function syntax we've already seen

```
function_name()
```

or

```
function_name(parameters)
```

Is what we write to get the function to do its job

```
function_name()
```

or

```
function_name(parameters)
```

Is what is job

When our function needs some form of data input to work we can give it a parameter or multiple parameters…More on this later

```
print()
                    lower()
len()
                              input()
```

...and many more....

These are already defined in Python, or built in. But we can write our own functions to do even more stuff.....

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 press_grind_beans

Think of it a different way…

```
def say_hello():
    print("Hello!")

say_hello()
```

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)
```

Let’s take this in

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


cash_withdrawal(300, 50449921)
```

We can use parameters instead of variables to be more efficient

Let's take this in

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


cash_withdrawal(300, 50449921)
```

Values for parameters are fed into the function and helps it do its job

We can use parameters instead of variables to be more efficient

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)
```

Think of it a different way…

```
def say_something(something):
    print("{}".format(something))


say_something(“Hello!")


say_something("Goodbye.")
```

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")
```

*Extra reading

Global variables and parameters?

Let’s take this in…

```
w_amount = 100

account_num = 12345678


def cash_withdrawal(amount, accnum):

    print("Withdrawing {} from account {}".format(amount, accnum))


cash_withdrawal(w_amount, account_num)

cash_withdrawal(300, 50449921)

cash_withdrawal(30, 50449921)
```

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
```

Add up two numbers and return the answer (without printing)

Let’s take this in

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


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

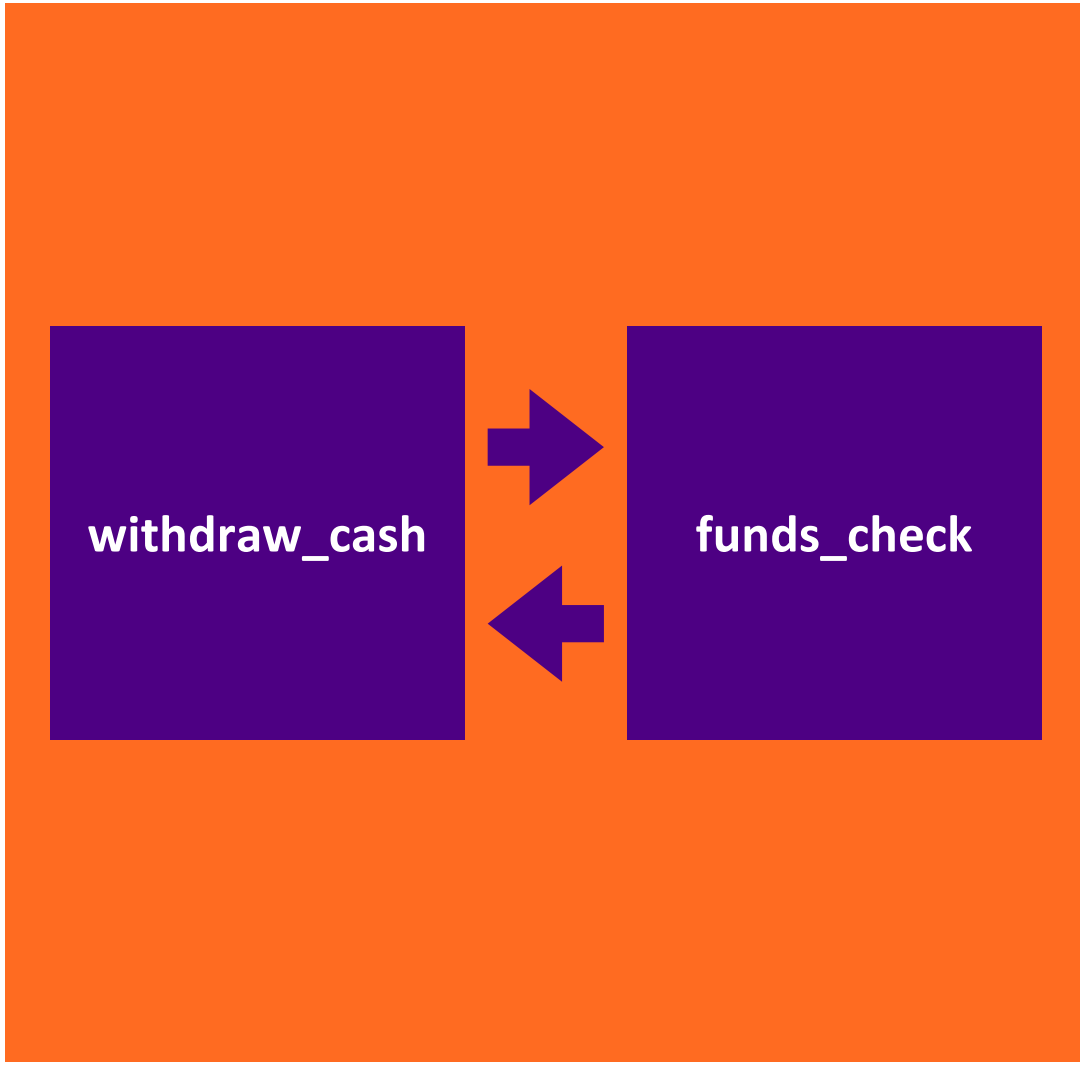
Add up two numbers, return the answer, and then print the result

So,
you see…

one function might call another function

and use the result of that function to achieve its goal

For example, in our wonderful cash machine, we might have something like …



Does customer have enough funds requested?

Check funds_check and return result to withdraw_cash

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)))
#Output: The temperature is 59°F
```

Functions

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

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

We call functions by using their identifiers

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

We call functions by using their identifiers

They break our code up into small chunks

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

We call functions by using their identifiers

They break our code up into small chunks

Use parameters to give our functions data inputs and flexibility (if we need them)

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

We call functions by using their identifiers

They break our code up into small chunks

Use parameters to give our functions data inputs and flexibility (if we need them)

We can use functions together, and one function might call another, and use the result of that function to achieve its goal

LEARNING OBJECTIVES

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

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 or more parameters and a running order count updated when the function is called :

```
order_count = 0

def take_order(topping):

    global order_count

    print("Pizza with {}".format(topping))

    order_count += 1

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!

Extra reading 1: Global and local variables

Research on the differences between global and local variables used in functions. Here's a link to start with:

https://www.guru99.com/variables-in-python.html

Example below of an improved version of our coffee bean grinder with an 'on/off' button

Let’s take this in

```
coffee_is_grinding = False

def press_grind_beans():
    global coffee_is_grinding
    if coffee_is_grinding:
        print("Stopping the grind")
        coffee_is_grinding = False
    else:
        print("Grinding is about to begin")
        coffee_is_grinding = True

press_grind_beans()
press_grind_beans()
```

Let's take this in

```
coffee_is_grinding = False

def press_grind_beans():
    global coffee_is_grinding
    if coffee_is_grinding:
        print("Stopping the grind")
        coffee_is_grinding = False
    else:
        print("Grinding is about to begin")
        coffee_is_grinding = True

press_grind_beans()
press_grind_beans()
```

Declare new variable with boolean value

Let's take this in

```
coffee_is_grinding = False

def press_grind_beans():
    global coffee_is_grinding
    if coffee_is_grinding:
        print("Stopping the grind")
        coffee_is_grinding = False
    else:
        print("Grinding is about to begin")
        coffee_is_grinding = True

press_grind_beans()
press_grind_beans()
```

Declare new function

Let’s take this in

```
coffee_is_grinding = False

def press_grind_beans():
    global coffee_is_grinding
    if coffee_is_grinding:
        print("Stopping the grin
        coffee_is_grinding = Fal
    else:
        print("Grinding is about
        coffee_is_grinding = Tru

press_grind_beans()
press_grind_beans()
```

Specify you want to use this variable inside this function

Let's take this in

```
coffee_is_grinding = False

def press_grind_beans():
    global coffee_is_grinding
    if coffee_is_grinding:
        print("Stopping the g
        coffee_is_grinding = False
    else:
        print("Grinding is about to begin")
        coffee_is_grinding = True

press_grind_beans()
press_grind_beans()
```

If coffee_is_grinding is true…

Let's take this in

```
coffee_is_grinding = False

def press_grind_beans():
    global coffee_is_grinding
    if coffee_is_grinding:
        print("Stopping the grind")
        coffee_is_grinding = False
    else:
        print("Grinding is about to begin")
        coffee_is_grinding = True

press_grind_beans()
press_grind_beans()
```

Stop it grinding

Let's take this in

```
coffee_is_grinding = False

def press_grind_beans():
    global coffee_is_grinding
    if coffee_is_grinding:
        print("Stopping the grind")
        coffee_is_grinding = False
    else:
        print("Grinding is about to begin")
        coffee_is_grinding = True

press_grind_beans()
press_grind_beans()
```

Turn it off

Let's take this in

```
coffee_is_grinding = False

def press_grind_beans():
    global coffee_is_grinding
    if coffee_is_grinding:
        print("Stopping the grind")
        coff
    else:
        print("Grinding is about to begin")
        coffee_is_grinding = True

press_grind_beans()
press_grind_beans()
```

Else if coffee_is_grinding is false…

Let's take this in

```
coffee_is_grinding = False

def press_grind_beans():
    global coffee_is_grinding
    if coffee_is_grinding:
        print("Stopping the grind")
        coffee_is_grinding = False
    else:
        print("Grinding is about to begin")
        coffee_is_grinding = True

press_grind_beans()
press_grind_beans()
```

Start grinding the coffee

Let’s take this in

```
coffee_is_grinding = False

def press_grind_beans():
    global coffee_is_grinding
    if coffee_is_grinding:
        print("Stopping the grind")
        coffee_is_grinding = False
    else:
        print("Grinding is about to begin")
        coffee_is_grinding = True

press_grind_beans()
press_grind_beans()
```

Turn it on

Let's take this in

```
coffee_is_grinding = False

def press_grind_beans():
    global coffee_is_grinding
    if coffee_is_grinding:
        print("Stopping the grind")
        coffee_is_grinding = False
    else:
        print("Grinding is about to begin")
        coffee_is_grinding = True

press_grind_beans()
press_grind_beans()
```

The function press_grind_beans()

Let’s take this in

```
coffee_is_grinding = False

def press_grind_beans():
    global coffee_is_grinding
    if coffee_is_grinding:
        print("Stopping the grind")
        coffee_is_grinding = False
    else:
        print("Grinding is about to begin")
        coffee_is_grinding = True

press_grind_beans()
press_grind_beans()
```

Run the function again

Let’s take this in

```
coffee_is_grinding = False

def press_grind_beans():
    global coffee_is_grinding
    if coffee_is_grinding:
        print("Stopping the grind")
        coffee_is_grinding = False
    else:
        print("Grinding is about to begin")
        coffee_is_grinding = True

press_grind_beans()#the coffee is not grinding
press_grind_beans()#the coffee is grinding
```