

What is this?



It's Called the Enigma Machine



- » Germany used it in World War 2 to send secret messages.
- » Rotating wheels inside scrambled every letter you typed.
- » Only an identical machine, set up the same way, could unscramble it.
- » Millions of possible settings — too many to guess by hand.
- » **Alan Turing** led the Bletchley Park team that cracked it — and his ideas became the basis of the modern computer.

PYTHON CRASH COURSE

Intro to Python

```
$ spyder
```

Zero experience needed. We're using Spyder, live.

— Dr. Hughes

Meet the Terminal

```
// bottom-right panel in Spyder
```

» Spyder's console runs Python **instantly**, one line at a time.

```
>>> # type here, press Enter
```

No saving. No running a file. Just type, and press Enter.

Python Is a Calculator

```
>>> 2 + 2  
4
```

// hit enter after typing this in Spyder's console

The Four Basics

+ add

- subtract

* multiply

/ divide

```
>>> 5 + 3
```

```
8
```

```
>>> 5 - 3
```

```
2
```

```
>>> 5 * 3
```

```
15
```

```
>>> 5 / 3
```

```
1.6666666666666667
```

// type each one into Spyder's console as we go

Try It Yourself

```
>>> 8 / 2 # pizza, split between 2 people
```

```
4.0
```

```
>>> 7 * 6
```

```
42
```

// type these into Spyder's console yourselves before revealing

Leftovers: the % Operator

```
// "modulo" – the remainder after dividing
```

```
>>> 10 % 3 # 10 slices, 3 people, 1 left over  
1
```

Order Matters

```
>>> 2 + 3 * 4
```

```
14
```

```
>>> (2 + 3) * 4
```

```
20
```

Python follows normal maths rules — multiply before add, unless brackets say otherwise.

Giving Things Names

A **variable** is a labelled box that holds a value.

name = value

Your First Variable

```
>>> age = 16
```

```
>>> age
```

```
16
```

// nothing prints after the first line — we just stored it

Strings: Text in Quotes

str

```
>>> name = "Zendaya"
```

```
>>> name
```

```
'Zendaya'
```

// quotes = it's text, not a number

Numbers: int vs float

int

```
1 score = 95
```

float

```
1 height = 1.75
```

// int = whole number. float = has a decimal point.

True or False: Booleans

bool

```
1 is_hungry = True
2 is_raining = False
```

// only ever two values: True or False

Check the Type

```
>>> type(age)
<class 'int'>
```

```
>>> type(name)
<class 'str'>
```

```
>>> type(height)
<class 'float'>
```

```
>>> type(is_hungry)
<class 'bool'>
```

Variables Can Change

```
>>> score = 95  
  
>>> score = score + 5  
  
>>> score  
100
```

In Python, `=` means *store*, not *equals*.

Remember $f(x)$ From Maths?

$$f(x) = x + 1$$

Put a number in, get a number out. A function is a machine for that.

Python Functions Do the Same

```
1 def add_one(x):  
2     return x + 1
```

// def = "define a function". return = "send this value back"

Calling a Function

```
>>> add_one(5)
```

```
6
```

Anatomy of a Function

```
1 def add_one(x):  
2     return x + 1
```

def = keyword

name we chose

input (parameter)

output

Multiple Inputs

```
1 def add(a, b):  
2     return a + b
```

```
>>> add(3, 4)  
7
```

Why Bother?

Write the maths once. Reuse it forever.

```
>>> add(10, 20)
```

```
30
```

```
>>> add(100, 1)
```

```
101
```

Over to You: The Function Ladder

» Work solo or paired. The functions live in a Spyder file called `ladder.py`. Start at rung 1. Check each rung against the expected value on your test chips. Teacher checks your tracker.

Twenty minutes. Your hands on the keyboard — not mine.

Where the Ladder Ends

```
1 def add(a, b):  
    # TODO  
  
3 def subtract(a, b):  
    # TODO  
  
5 def multiply(a, b):  
    # TODO  
  
7 def divide(a, b):  
    # TODO
```

// rung 5 — these four are your target

We Do This One Together

```
1 def add(a, b):  
2     return a + b
```

Testing `add()`

```
>>> add(2, 3)
```

```
5
```

Model Answer: `subtract()`

```
1 def subtract(a, b):  
2     return a - b
```

// check yours against this after work time

Testing `subtract()`

```
>>> subtract(10, 4)
```

```
6
```

Model Answer: `multiply()`

```
1 def multiply(a, b):  
2     return a * b
```

// check yours against this after work time

Testing `multiply()`

```
>>> multiply(6, 7)
```

```
42
```

Model Answer: `divide()`

```
1 def divide(a, b):  
2     return a / b
```

// check yours against this after work time

Testing `divide()`

```
>>> divide(20, 4)  
5.0
```

```
# Division always gives back a float, even when it divides evenly.
```

Compare Yours

```
1 def add(a, b):  
  2 return a + b  
  
3 def subtract(a, b):  
  4 return a - b  
  
5 def multiply(a, b):  
  6 return a * b  
  
7 def divide(a, b):  
  8 return a / b
```

Same shape, different maths.

Before the Calculator

» The function names must be exactly `add`, `subtract`, `multiply`, `divide`. All four must run in the console before the calculator works.

// calculator.py calls them by name. One wrong name and it finds nothing

Making Decisions

```
# If it's raining → bring an umbrella. Otherwise, don't.
```

Code needs to make decisions too.

The `if` Statement

```
1 temperature = 30
2 if temperature > 25:
3     print("It's hot!")
```

```
It's hot!
```

if / else

```
1 temperature = 10
2 if temperature > 25:
3     print("It's hot!")
4 else:
5     print("It's chilly")
```

It's chilly

// else only runs when the if didn't

Comparison Operators

`==` equal to

`!=` not equal to

`>` greater than

`<` less than

`>=` greater or equal

`<=` less or equal

// note the double `==` — a single `=` would try to store a value instead

elif : More Than Two Choices

```
1 score = 72
2 if score >= 90:
3     print("A")
4 elif score >= 70:
5     print("B")
6 elif score >= 50:
7     print("C")
8 else:
9     print("Needs work")
```

B

// Python checks each condition top to bottom, stops at the first true one

elif : One Branch at a Time

```
1 score = 72
2 if score >= 90:
3     print("A")
4 elif score >= 70:
5     print("B")
6 elif score >= 50:
7     print("C")
8 else:
9     print("Needs work")
```

B

// Python checks each condition top to bottom, stops at the first true one

Combining Conditions

```
1 age = 16
2 if age >= 13 and age <= 19:
3     print("You're a teenager!")
```

```
You're a teenager!
```

// "and" means both sides need to be true

Let's Build a Calculator

Variables + functions + if statements, together.

Step 1: The Four Functions

add

subtract

multiply

divide

// paste YOUR ladder functions into the top of calculator.py

Step 2: Ask the User

```
1 operation = input("Pick: + - * / ")
2 num1 = float(input("First number: "))
3 num2 = float(input("Second number: "))
```

`input()` always gives text back — `float()` turns it into a number

Step 3: Decide What To Do

```
1 if operation == "+":  
    2 result = add(num1, num2)  
3 elif operation == "-":  
    4 result = subtract(num1, num2)  
5 elif operation == "*":  
    6 result = multiply(num1, num2)  
7 elif operation == "/":  
    8 result = divide(num1, num2)
```

Step 4: Show the Result

```
1 print(f"Answer: {result}")
```

// f-strings drop a variable straight into text with { }

calculator.py

```
1 operation = input("Pick: + - * / ")
2 num1 = float(input("First number: "))
3 num2 = float(input("Second number: "))

4 if operation == "+":
5     result = add(num1, num2)
6 elif operation == "-":
7     result = subtract(num1, num2)
8 elif operation == "*":
9     result = multiply(num1, num2)
10 elif operation == "/":
11     result = divide(num1, num2)

12 print(f"Answer: {result}")
```

// plus YOUR four ladder functions pasted at the top of the file

Run It!

Pick: + - * / *

First number: 6

Second number: 7

Answer: 42.0

Doing Things Again and Again

```
# Loops = don't repeat yourself.
```

The `for` Loop

```
1 for i in range(5):  
2     print(i)
```

0

1

2

3

4

// range(5) counts 0, 1, 2, 3, 4 — five numbers, starting at 0

The `for` Loop, One Lap at a Time

```
1 for i in range(5):  
2     print(i)
```

```
0  
1  
2  
3  
4
```

// range(5) counts 0, 1, 2, 3, 4 — five numbers, starting at 0

for Over a List

```
1 snacks = ["pizza", "chips", "soda"]  
2 for snack in snacks:  
3     print(snack)
```

pizza

chips

soda

The `while` Loop

```
1 count = 0
2 while count < 3:
3     print(count)
4     count += 1
```

0

1

2

// keeps going while the condition is true — forget `count += 1` and it never stops

for vs while

for while

for = a known number of times. while = until something changes.

What You Built Today

- ✓ terminal maths
- ✓ variables
- ✓ functions
- ✓ a set of functions you wrote yourself
- ✓ if statements
- ✓ a working calculator
- ✓ for & while loops

You're a Programmer Now

Every big program is just this — smaller pieces, put together.

» Challenge: the stretch rung. Add `power(a, b)` to `ladder.py` and to the calculator (that's a new elif for the `**` symbol).