First thing’s first

All Around the World

Display the 8th character in upper case on the console

First thing’s first

```
print("All around the world"[7].upper())
```

Alternative solution

```
print("All around the world".upper()[7])
```

Which one is more efficient and why?

PYTHON FUNDAMENTALS

Variables

LEARNING OBJECTIVES

- To understand and use variables and operators to store values and do calculations
- To use snake_case when naming variables
- To understand how to access data in variables

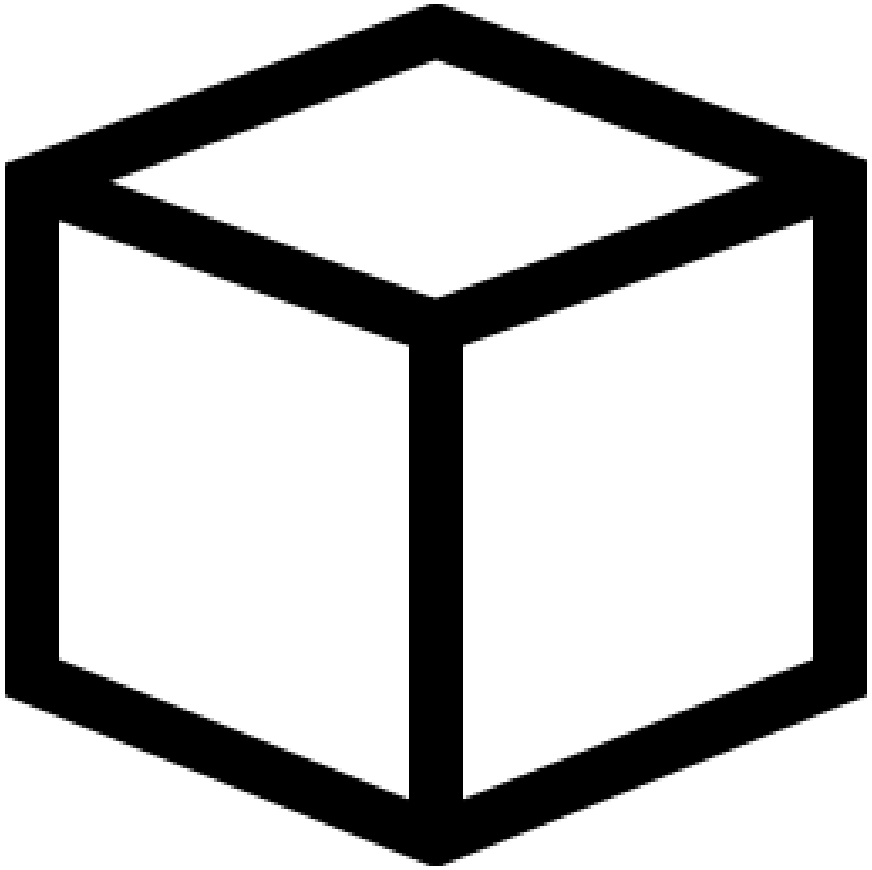
Things are getting interesting

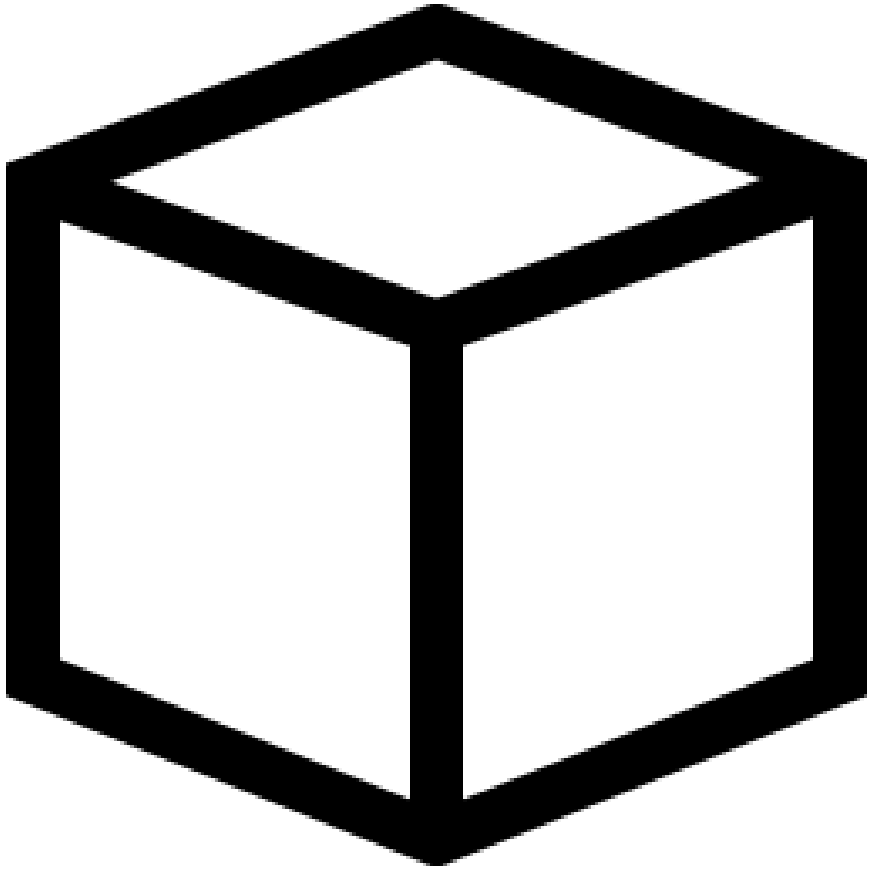
Introducing

Variables

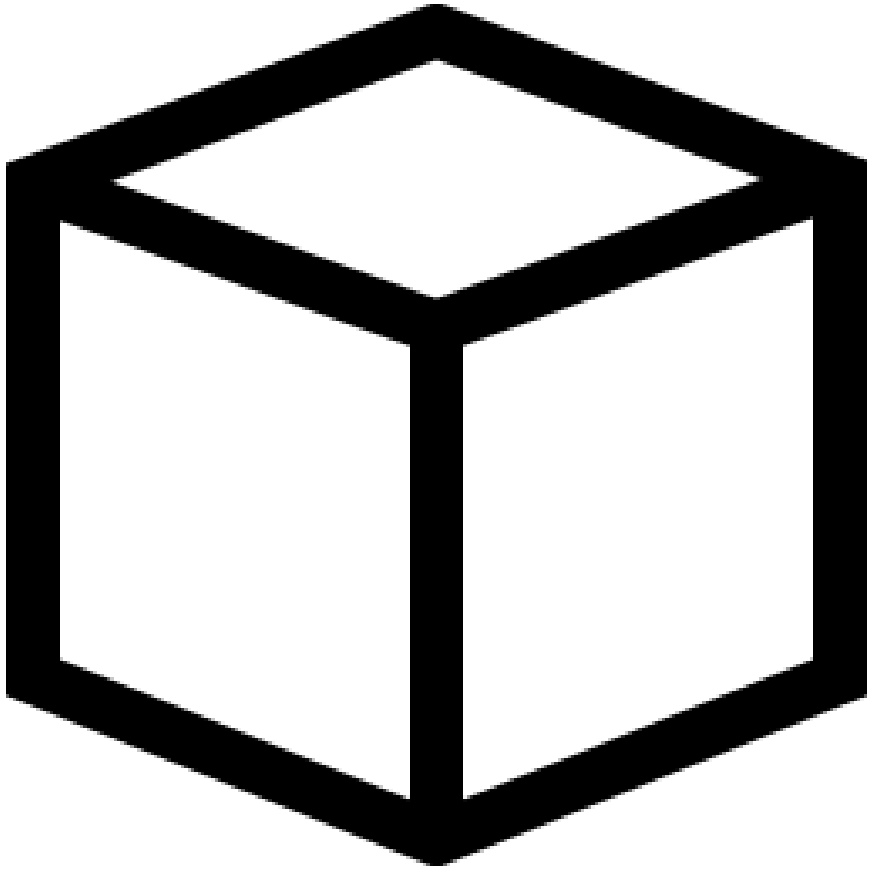
They're like boxes.

Not very technical that, is it?



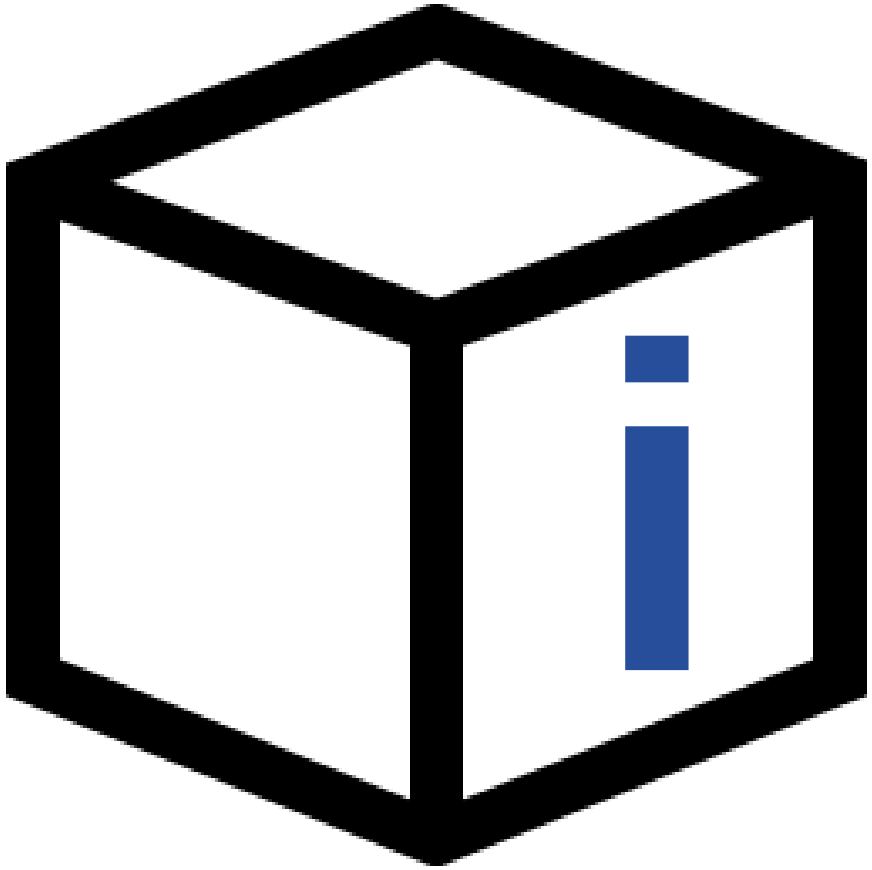


We store items in boxes to retrieve later



We store items in boxes to retrieve later

Different items can be stored in the box at different times



We store items in boxes to retrieve later

Different items can be stored in the box at different times

In code, we give variables names so we can access things inside them. Exactly like saying “get me that thing from the blue box over there”

Imagine a cash machine

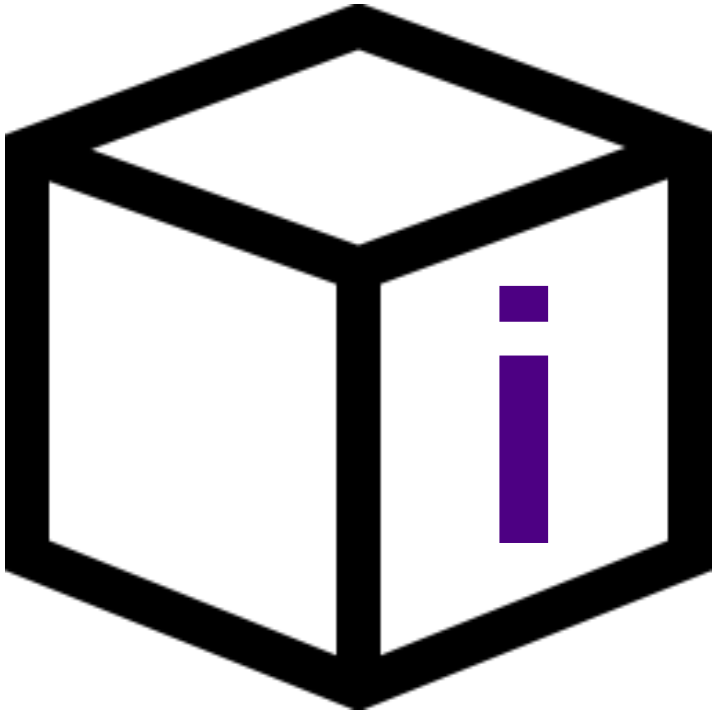
We will be able to reuse code

```
WITHDRAW 10_POUNDS
FROM 82929201
```

should be

```
WITHDRAW AMOUNT
FROM ACCOUNTNUM
```

So variables…



1) allow us to store data inside them

2) access them via a name

3) then place new data in them whenever we want

We don’t need to tell Python what kind of data will be stored in a variable

Because it’s a “loosely typed” language

So if I want to store my name in a variable, what kind of data type would that be?

Yes, string. You're just too good.

```
my_name = "Ann"
```

Create a variable called name which holds the string “Ann”

```python
my_name = "Ann"

print(my_name)
```

You can print `my_name` by referring to the variable name that has been assigned

All we have to do is to give our variable a name and assign it a value

```
my_name = "Ann"
my_age = 18
student = True
```

You can change what is stored in a variable

```python
my_name = "Ann"
my_age = 18
student = True


student = False
```

```
my_name = "Ann"
my_age = 18
student = True
```

Assign student to True

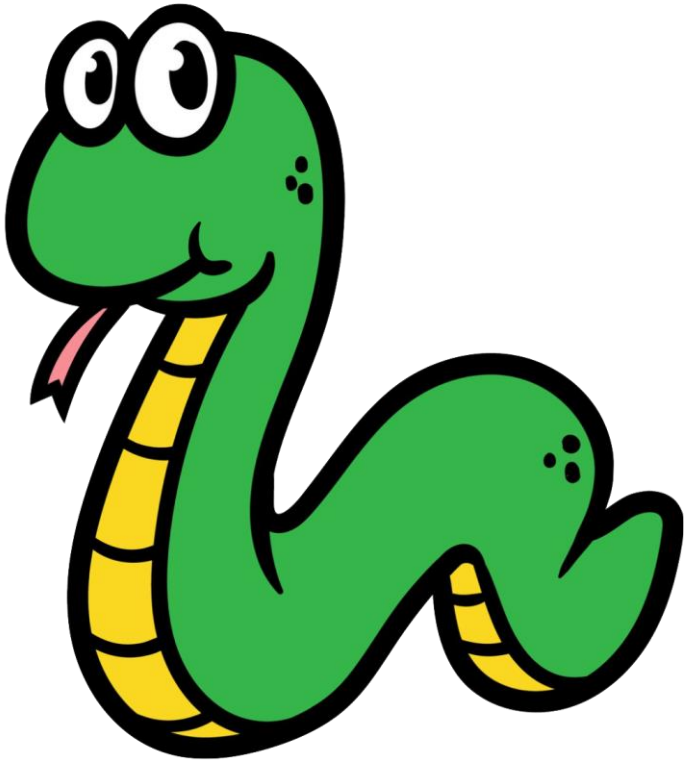
```
student = False
```

```
my_name = "Ann"
my_age = 18
student = True

student = False
```

Updates the variable `student` from `True` to `False`

Have you noticed we've stuck to a particular convention when naming variables?



```
favourite_drink
this_number
first_name
```

…It’s called snake_case.

This is Python after all.

It’s best practice and enhances code readability

How to access data in variables

```
print("Ann")
```

```
print("A\n")
```

To VSCode

```python
my_name = "Ann"

print(my_name)
```

```python
fav_drink = "hot chocolate"

print(fav_drink)
```

```python
fav_drink = "hot chocolate"

print("My favourite drink is ", fav_drink)
```

```python
fav_drink = "hot chocolate"

print("My favourite drink is " + fav_drink)
```

.format() method

You can use `{ }` and `.format()` method to create sensible outputs

```
fav_drink = "hot chocolate"

print("My favourite drink is {}".format(fav_drink))
```

```python
name = "Ann"

fav_drink = "hot chocolate"

print("{}'s favourite drink is {}".format(name, fav_drink))

#Ann's favourite drink is hot chocolate
```

```
name = "Ann"
fav_drink = "hot chocolate"
print("{}'s favourite drink is {}".format(name, fav_drink))
#Ann's favourite drink is hot chocolate


fav_drink = "coffee"
print("{}'s favourite drink is {}".format(name, fav_drink))
#Ann's favourite drink is coffee
```

```
name = "Ann"
fav_drink = "hot chocolate"
print("{}'s favourite drink is {}".format(name, fav_drink))
#Ann's favourite drink is hot chocolate


fav_drink = "coffee"
print("{}'s favourite d                                    k))
#Ann's favourite drink is coffee
```

Remember you can change what is stored in a variable.

f"...{variable_name}" method

```
fav_drink = "hot chocolate"

print("My favourite drink is {}”.format(fav_drink))

print(f"My favourite drink is {fav_drink}")
```

Another way of using format

```
name = "Ann"

fav_drink = "hot chocolate"

print("{}'s favourite drink is {}".format(name, fav_drink))


print(f"{name}'s favourite drink is {fav_drink}")
```

Another way of using format

“Quotes” within ‘quotes’?

```
name = "Ann"

fav_drink = "hot chocolate"

print("{}'s favourite drink is {}".format(name, fav_drink))

#Ann's favourite drink is hot chocolate
```

You can use either " or ‘ to declare a string - if you need to use one within another, this is generally fine

```
name = "Ann"

fav_drink = "hot chocolate"

print("{}'s favourite drink is {}".format(name, fav_drink))

#Ann's favourite drink is hot chocolate
```

```
name = 'Ann'

fav_drink = 'hot chocolate'

print('{}'s favourite drink is {}'.format(name, fav_drink))

#Ann's favourite drink is hot chocolate
```

However, *apostrophes* cannot be used in a string declared with a single quote- Python starts to think you are trying to use multiple strings

```
name = 'Ann

fav_drink = 'hot chocolate'

print('{}'s favourite drink is {}' format(name, fav_drink))

#Ann's favourite drink is hot chocolate
```

```
name = 'Ann'

fav_drink = 'hot chocolate'

print('{}\'s favourite drink is {}'.format(name, fav_drink))

#Ann's favourite drink is hot chocolate
```

```
name = 'Ann'

fav_drink = 'hot chocolate'

print('{}\'s favourite drink is {}'.format(name, fav_drink))

#Ann's favourite drink is hot chocolate
```

\ is the escape backslash - use this to ignore whatever the subsequent character is

```
name = 'Ann'

fav_drink = 'hot chocolate'

print('{}\'s favourite drink is {}'.format(name, fav_drink))

#Ann's favourite drink is hot chocolate
```

\ is the escape backslash - use this to ignore whatever the subsequent character is

In this case, the \ tells Python not to end the string, but simply display the quotation mark

```
name = 'Ann'
fav_drink = 'hot chocolate'
print('{}\'s favourite drink is {}'.format(name, fav_drink))
#Ann's favourite drink is hot chocolate


fav_drink = 'coffee'
print('{}\'s favourite drink is {}'.format(name, fav_drink))
#Ann's favourite drink is coffee
```

What data types?

String

Boolean

Integer

None

Floating point

String : for representing text

Integer: for representing whole number

Floating point: for decimals

None : for nothing

Boolean: for True and False

Time for sum maths

+

-

*

**

/

%

Arithmetic Operators for calculations

=

*=

+=

/=

-=

Assignment Operators to store values

=

Assign operator

```
i = 10
```

Assigning i to the number 10

+

Addition

Assigning i to the number 10

```
i = 10
```

Then add 2 to the value of i

```
i = 10
```

```
i = i + 2
```

i = 12

*Arithmetic operator

Then add 2 to the value of i

```
i = 10


i += 2

# i = 12
```

*Arithmetic operator

LEARNING OBJECTIVES

- To understand and use variables and operators to store values and do calculations
- To use snake_case when naming variables
- To understand how to access data in variables

Activity 1:

Create a program that stores someone’s name, age and favourite colour that prints these in a complete sentence

Activity 2:

Create a program that stores what you eat today for breakfast, lunch and dinner, print these.

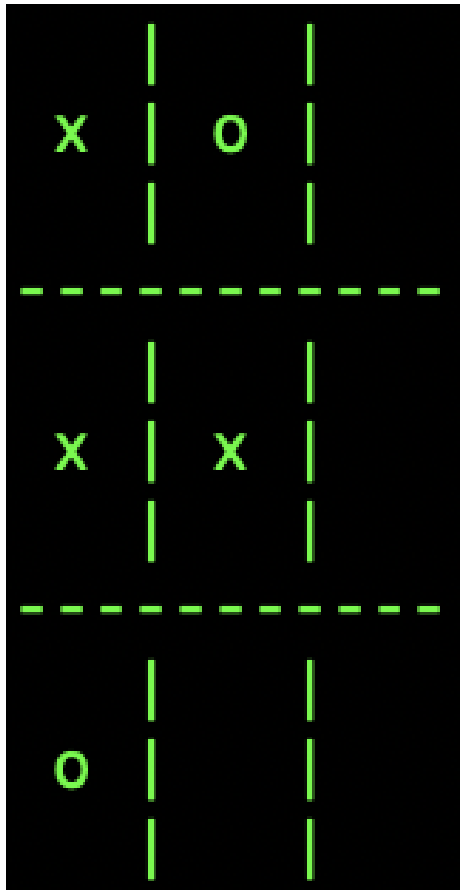
Update each of these variables to what you will eat tomorrow, print these.

Activity 3:

(1) Create a 9 variables space1, space2… space9

(2) Assign either the value ‘x’, ‘o’, ‘ ‘ to each of these variable

(3) Insert the variables into the board using the {} .format() syntax and make your board look like the one displayed



Activity 4:

Research into all operators mentioned in this session, give an example of each (see next two slides)

=

*=

+=

/=

-=

And check out these assignment operators

Extra: Activity 5:

Create a program that calculate the number of days from today to your birth date, and print this out.