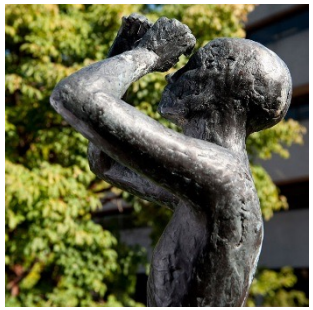


# Lesson 2 – Variables, I/O and Conditionals

MaREI Python Course – September 2022

**Cathal Hoare**

A TRADITION OF INDEPENDENT THINKING

UCC

University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

# Lecture Contents

Variables

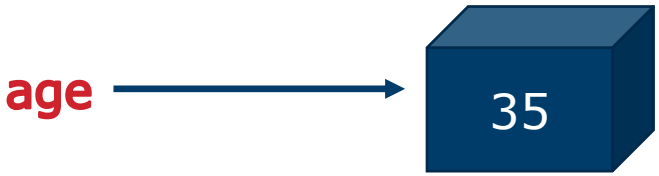
Arithmetic

Basic I/O

Conditionals

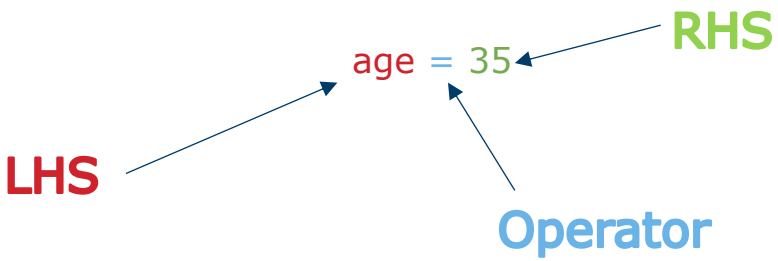
# Variables

- A variable can be thought of:
  - As a place where we can place, or assign some information
  - That place has a nickname so that we can refer to it




- In reality, it is an area of computer memory that has an address. Since the address is a long number, its easier for us to give that an understandable name
  - When the code runs that name is converted to the complex address

# Assigning Variables

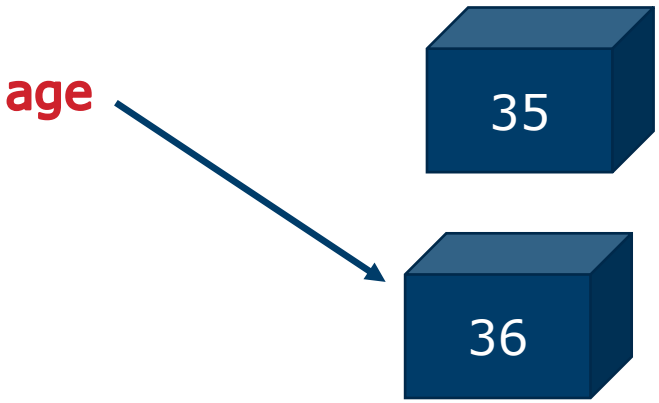



- 'age' (Left Hand Side (LHS)) is the nickname of the location
  - This should be descriptive, have no spaces, not start with a number, not be a reserved word. Case matters.
  - A variable must always have been LHS before it can appear RHS
- '=' is the assignment operator
- The Right Hand Side (RHS) is whatever we are assigning as a value
  - This can be a constant value, for example a number
  - It can be the value of another variable
  - It can be the result of a function
- Once assigned a variable has a type. This allows to answer 'What type of thing does it contain' – in this case a number

# Reassigning Variables

- A variable can be reassigned:

age = 36

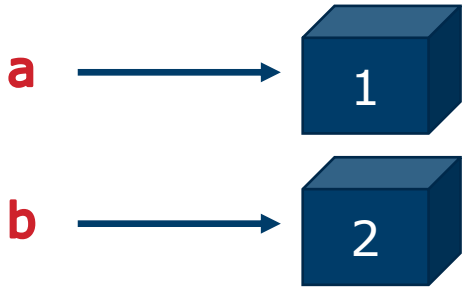


# An example – swapping two numbers

- Supposing we have two variables as follows:

a = 1

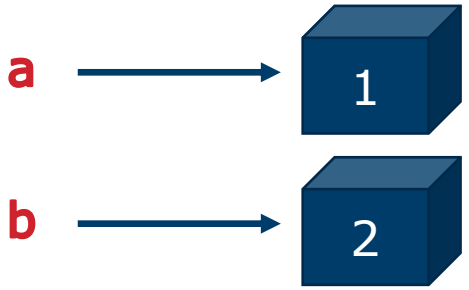
b = 2



# An example – swapping two numbers

- Supposing we have two variables as follows:

```
a = 1
b = 2
```



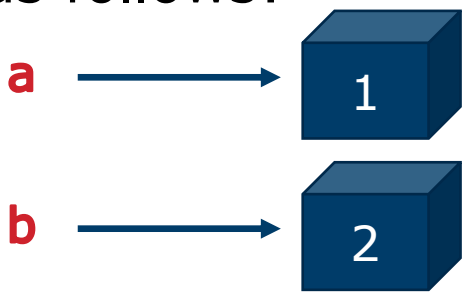

- To Swap we could try:

```
a = b
b = a
```

# An example – swapping two numbers

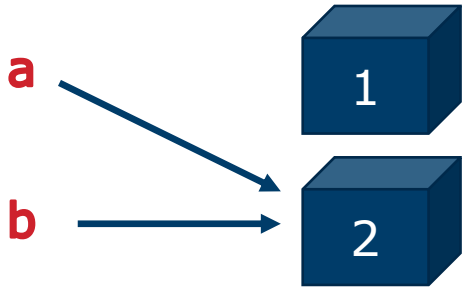
- Supposing we have two variables as follows:

```
a = 1
b = 2
```




- To Swap we could try:

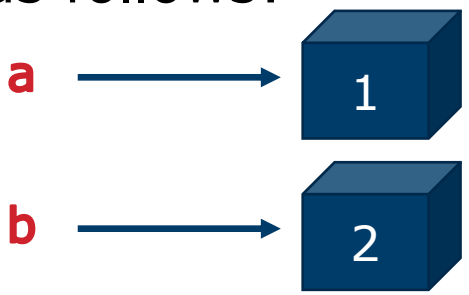
```
a = b
b = a
```



# An example – swapping two numbers

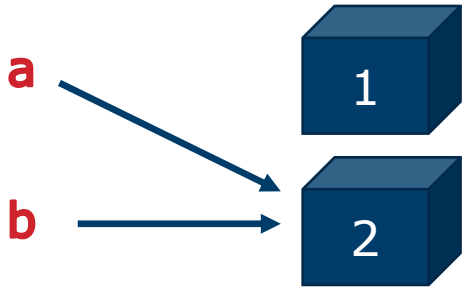
- Supposing we have two variables as follows:

```
a = 1
b = 2
```




- To Swap we could try:

```
a = b
b = a
```



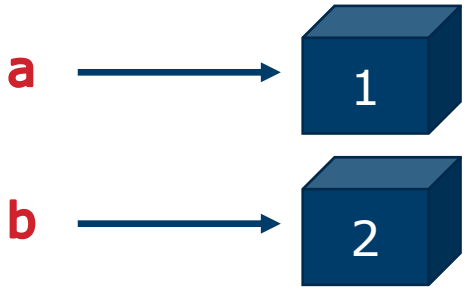

NO GOOD!

# An example – swapping two numbers

- Supposing we have two variables as follows:

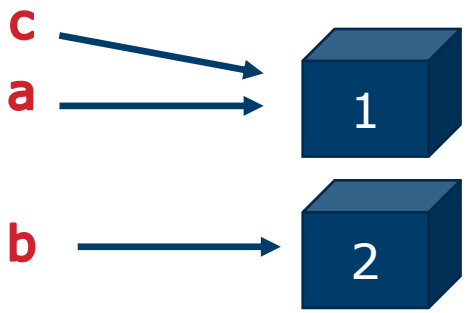
a = 1

b = 2




- To Swap we could try:

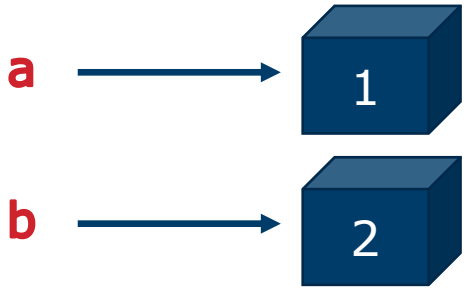
c = a



# An example – swapping two numbers

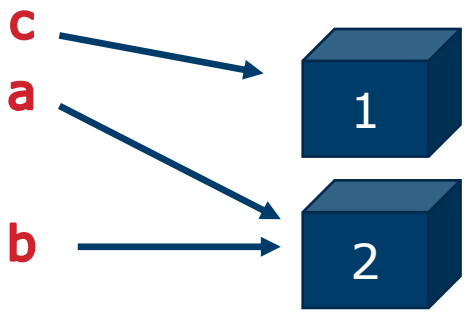
- Supposing we have two variables as follows:

```
a = 1
b = 2
```




- To Swap we could try:

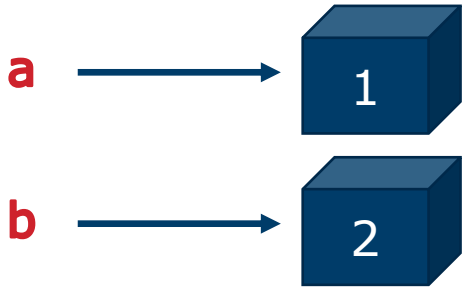
```
c = a
a = b
```



# An example – swapping two numbers

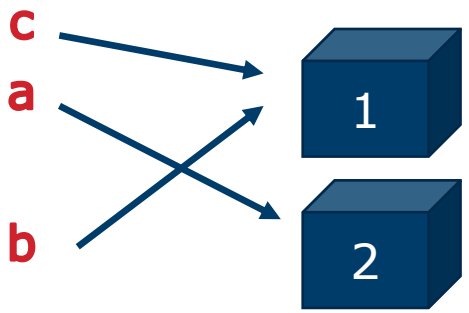
- Supposing we have two variables as follows:

```
a = 1
b = 2
```




- To Swap we could try:

```
c = a
a = b
b = c
```



# An example – swapping two numbers

- In fact, Python lets us do this:

```
a = 1
b = 2
a,b = b,a
```

- But not all languages support this approach!

- Remember:
  - Programming is made up of a plan (or problem solving) and writing code
  - Problem solving can be applied to any language
  - Then we just need to learn the syntax for a new language

# Reading Values from Users

age = input("Enter your age please:")

- In order to read a value from a user, we use the input function
  - A function is a piece of code that implements something that we will use many times – we write it once, store it in a library, and then use it in our code
  - When use use a function we are said to have called it
  - A function can be written so that we can pass it information by setting variables in the function – known as passing arguments
  - A function can return a result that we can store in a variable and then use in our code
- For input, we pass an argument that is presented to the user, and the function returns a value that is a string (an ordered list of characters)
- If we want to use the input value as a number, we must convert or cast the value.

# Writing to Output

```
print("Hello World!")
```

- Similarly, we can use a function to write information to the screen
- The function is called print and in its simplest form, takes a single argument which is the value written to the screen

# Writing to Output

- A more complex form of the print function lets us form a string that contains text combined with place holders for variable values.
- This is called string interpolation.

```
name = 'Tom'
age = 25
print("%s is %d years old", (name, age))
```

- The place holders begin with a % character followed by a letter to denote the type that will be displayed.

| Specifier | Meaning |
|---|---|
| d | Integer |
| o | Octal (base 8) value |
| x | Lowercase hexadecimal (base 16) |
| X | Uppercase hexadecimal (base 16) |
| e | Lowercase float exponential |
| E | Uppercase float exponential |
| F | Float |
| s | String |
| % | % character |

# Exercise!

- Wish me a happy birthday in cat years (true age multiplied by 7). I want a program that will prompt the user for their name and their age and will then print out a message as follows (assumes ‘Cathal’ is 20)

  ‘Happy Birthday Cathal you are 140’

# Integer Arithmetic

- Integers are whole numbers (no fraction part)

- a = 1 + 2  #answer 3
- b = 2 – 1  #answer 1
- c = 2 * 2  #answer 4
- d = 2 / 2  #answer 1.0
- e = 5 // 2  # answer 2
- f = 5 % 2  #answer 1

- Operator precedence:
  - The interpreter will do multiplication and division first, followed by addition and subtraction
  - If we want to change the order use brackets
  - g = (1 + 2) * 3  #answer is 9

# Floating Point Arithmetic

- Floating point numbers are numbers with a fraction.
- We can apply all of the same operators.
- Floats can lack precision – eventually (after 17 decimal places) the number will be rounded.
- Floats can overflow – eventually a floating point number will become too big and this can cause the interpreter to treat the number as infinite.

# Booleans

- Booleans are either True or False

- We can test conditions using Boolean operators. e.g.
  - a and b will be true if both a and b are true, otherwise false
  - a or b will be true if either a or b are true, otherwise false
  - a not 1 will be true if the value of a isn't 1

# Strings

- A string is a contiguous group of characters
  - e.g. “Cathal” or ‘Cathal’
- We can apply built in functions such as len
  - e.g. len(“Cathal”) will return 6
- We can apply operators to strings:
  - “cathal ” + “hoare” will return “cathal hoare”
  - “a” * 3 will return “aaa”

# Converting Variables

- We can convert between types:
  - int(“3”) will return 3
  - float(3) will return 3.0
- But we should be careful:
  - int(3.6) will return 3 - it simply slices off the fraction and causes a loss of precision
  - round(3.6) will return 4

# Letting Programs Choose

- When we write a Python program in its simplest form, the instructions run in sequence from the top line to bottom.
- This rarely gives enough expressivity to do something useful (it also makes for code that is hard to maintain or modify but that's a story for a different day…).
- Depending on circumstances, may want our code to make choices around what code is executed.
- We can define conditions (hence the name conditionals) that wrap blocks of code. If the condition is met (found to be true) then the block of code executes.

# if statements

```
#Version 1
score = 55
if score >= 40:
    print("Congrats you passed the exam")
```

declare a variable so we have something to check

If tells the program we are asking it to make a choice. Create a condition (score >= 40) – if this is true, the block runs

The block that executes – note how the block is indented - this separates the block controlled by the condition from other code in the program

# if statements

- If we need more complexity, we can chain conditions together. In this case, we use these to create categories of scores and depending on the category, we print an appropriate message.

```
#Version 2
score = 71
if score >= 70:
    print("Congrats you got a 1H")
elif score < 70 and score >= 55:
    print("Congrats you got a 2H")
elif score < 55 and score >= 40:
    print("Congrats you passed")
else:
    print("Oh dear")
```

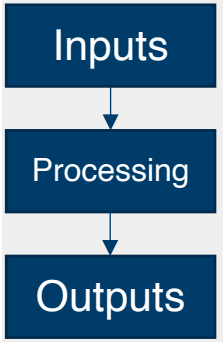
We will always have an if

elif allows us to add other conditions – we might or might not have these

else represents 'every other circumstance'

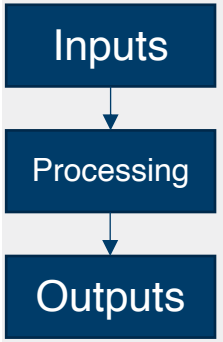
# Exercise!

BMI is a measure used by doctors and other health professionals to make decisions about health recommendations to patients or to decide on courses of treatment. It is calculated by dividing a person's mass by the square of their height. If the score is below 18.5 BMI is considered low. If the score is above 25 it is considered high. Write a program to allow a user to calculate their own BMI.



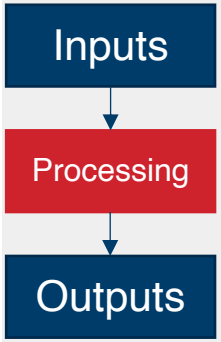
# Let's Write Some Code

BMI is a measure used by doctors and other health professionals to make decisions about health recommendations to patients or to decide on courses of treatment. It is calculated by dividing a person's mass by the square of their height. If the score is below 18.5 BMI is considered low. If the score is above 25 it is considered high. Write a program to allow a user to calculate their own BMI.



# Let's Write Some Code

BMI is a measure used by doctors and other health professionals to make decisions about health recommendations to patients or to decide on courses of treatment. It is calculated by dividing a person's mass by the square of their height. If the score is below 18.5 BMI is considered low. If the score is above 25 it is considered high. Write a program to allow a user to calculate their own BMI.

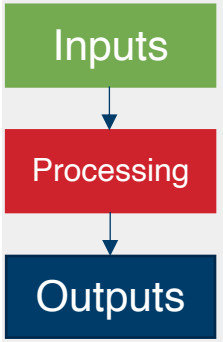



#Gives us a formula including a calculation and a set of variables

bmi = mass / (height * height)

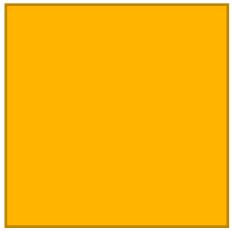
# Let's Write Some Code

BMI is a measure used by doctors and other health professionals to make decisions about health recommendations to patients or to decide on courses of treatment. It is calculated by dividing a person's mass by the square of their height. If the score is below 18.5 BMI is considered low. If the score is above 25 it is considered high. Write a program to allow a user to calculate their own BMI.



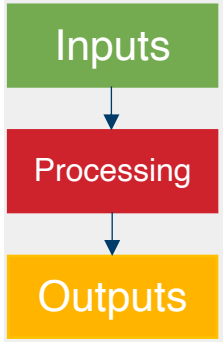

```
#We need to get some inputs. We will ask the user. We will also need to
#change these to numbers since we are going to do numeric operations

mass = int(input("Please enter your mass"))

height = int(input("Please enter your height"))

bmi = mass / (height * height)
```

# Let's Write Some Code



BMI is a measure used by doctors and other health professionals to make decisions about health recommendations to patients or to decide on courses of treatment. It is calculated by dividing a person's mass by the square of their height. If the score is below 18.5 BMI is considered low. If the score is above 25 it is considered high.

Write a program to allow a user to calculate their own BMI.




```
# We add some output. We use conditionals to decide on what message to print based on result
mass = int(input("Please enter your mass"))
height = int(input("Please enter your height"))
bmi = mass / (height * height)
if bmi < 18.5:
    print("BMI is low")
elif bmi > 25:
    print("BMI is high")
else:
    print("BMI is ok")
```


Next : going loopy