Introduction to Computer Programming

1.2 Variables

Variables

- We need a way to store and use values, e.g. numbers, within a program
- We can *assign* a value to a variable.
- The `print` function displays whatever is between the parentheses `(...)`

In [14]:

```
A = 2


print(A)
```

```
2
```

Variables can be created on separate or single lines

In [4]:

```
A = 1.0
B = 4.0
```

In [15]:

```
A, B = 1.0, 4.0
```

```
1.0
```

Data Types

Every variable has a type (`int` , `float` , `string` ....).

Basic Data Types (not exhaustive)

- `int` integer
- `float` floating point number (number with decimal point)

- `str` string: text data enclosed within quotation marks
  e.g. `'text'` or `"text"`
  (including number represented as text data)
- `bool` boolean: True or False

A type is automatically assigned when a variable is created.

Python's `type()` function returns the type of a variable within the parentheses `(...)`.

Example: Create some variables and display their type

In [19]:

```
A = 1
print(A, type(A))

B = 1.0
print(B, type(B))

C = '1'
print(C, type(C))

D = True
print(D, type(D))
```

```
1 <class 'int'>
1.0 <class 'float'>
1 <class 'str'>
True <class 'bool'>
```

Comments:
Comments are notes in the program that are not run as code.
The hash `#` symbol is used to:

- add a comment to a line of code to make a note about what it does.
- *comment* a line of code out to prevent it from running.

Either type a `#` at the start of the comment **or** to comment a whole line, select if and press `ctrl + 1` in Spyder (`ctrl + /` in many other softwares)

When you complete the in-class exercises today you can choose to:

- save your answers to each exercise as separate .py (Python) files
- *comment out* some of your code to allow you to store it all in one file, but only run certain (*uncommented*) blocks of code

Casting

The data type of a variable can be converted by *casting* (`int(variable_name)` , `float(variable_name)` ....)

Example: Convert from a floating point number to an integer

In [21]:

```
B = 1.0
B = int(E)
print(E, type(E))
```

```
1 <class 'int'>
```

Arithmetic Operators

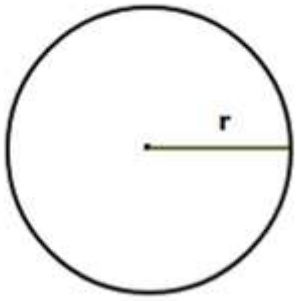
Python can be used like a calculator.

Arithmetic operators (+, -, /, * ....) are used with numeric values to perform common mathematical operations.

Operators are listed in order of operator precedence

**	Exponent
*	Multiplication
/	Division
//	Floor division (round down to the next integer)
%	Modulo `a % b = a - b * (a // b)` (remainder)
+	Addition
−	Subtraction

Example: Find the area of a circle with radius 2 metres



$$A = \pi r^2$$

In [10]:

```
pi = 3.142

r = 2

A = pi * r ** 2

print(A, 'm2')
```

```
12.568 m2
```

Arithmetic operators - a word of warning!

Question: What will the output of the following code be?

```
A = 2
B = '2'

print(A + A)

print(A + B)

print(B + B)
```

```
A = 2
B = '2'
```

Answer: Numbers represented as strings are not recognised as numerical values.
Arithmetic Operators behave differently on numerical and non-numerical values.

```
print(A + A)
```

> 2

```
print(A + B)
```

> Error.
> Cannot add numerical and non-numerical value

```
print(A + A)
```

> 22
> Adding string (text) data connect the two strings

In [24]:

```
A = 2

B = '2'

print (A + A)

# print (A + B) # generates an error

print (B + B) # strings are connected using +
```

```
4
22
```

Strings

Strings behave differently to numerical data.

We can return the Nth character(s) of a string with `string[N]`

Characters are *indexed* with integer values, starting from 0

In [31]:

```
x = 'Hello'

# print(x[0])   # first letter

# print(x[4])   # last letter
# print(x[-1])

# print(x[0:3]) # first 3 letters (excludes 'stop value')
# print(x[:3])

# print(x[2:5])  # last three letters
# print(x[2:])
# print(x[-3:])
```

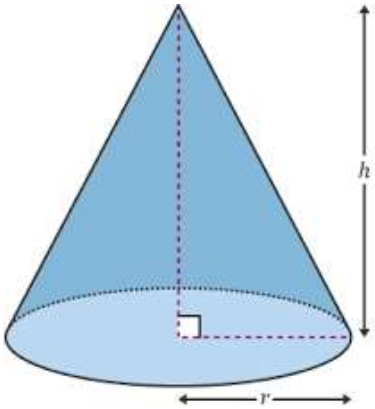
```
llo
llo
llo
```

In-class Demos

Example 1: Find the volume of a cone.

Base radius = 2 cm
Height = 10 cm

Volume of a cone: $V = \frac{Ah}{3}$
where A = base area



In [ ]:

In [ ]:

Example 2: Create a variabale `Name` and assign it a `string` value.

Use an arithmetc operator and `Name` to print the output: `My name is` followed by the value of `Name` . e.g.

```
My name is Hemma
```

In [ ]:

In [ ]: