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 [25]:

```
b = 3

print(b)
```

```
3
```

Variables can be created on separate or single lines

In [1]:

```
a = 1.0
b = 4.0

print(a, b)
```

```
1.0 4.0
```

In [4]:

```
a, b = 1.0, 4.0

print(a, b)
```

```
1.0 4.0
```

Data Types

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

Basic Data Types (not exhaustive)

- `int` **integer**: (e.g. 3, 88)
- `float` **floating point number**: number with decimal point (e.g. 1.5, 99.9626)

- `str` **string**: text data enclosed within quotation marks (e.g. `'hello'` or `"12"`)
  (includes numbers represented as text data)
- `bool` **Boolean**: `True` or `False`
  (first letter capitalised)

A type is automatically assigned when a variable is created.

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

Example: What is the **type** of each of these variables?

In [3]:

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

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

In [1]:

```
b = 1.0
print(b, type(b)) # print variable
```

```
1.0 <class 'float'>
```

In [2]:

```
c = '1'
print(c, type(c))
```

```
1 <class 'str'>
```

In [ ]:

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

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 [ ]:

```

```

Arithmetic Operators

Arithmetic Operators

Python can be used like a calculator.

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

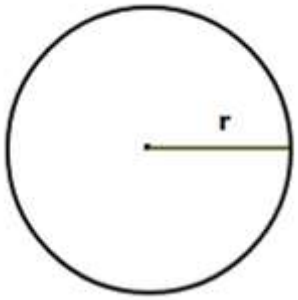
$**$	Exponent
/	Division
$*$	Multiplication
//	Floor division (round down to the next integer)
$\%$	Modulo `a % b = a - (b * a // b)` (remainder)
$+$	Addition
$-$	Subtraction

Operators in order of operator precedence

$**$	**Exponent**
/, $*$, //, $\%$	**Division, multiplication, floor division, modulo** (evaluated left to right in code)
$+$, $-$	**Addition, subtraction** (evaluated left to right in code)

Example:

Find the area of a circle with radius 2 metres



$A = \pi r^2$
where $\pi = 3.142$

In [ ]:

Comments:
A line of code that begins with the hash # symbol is a **comment**.

Comments are used for two things:

1. A line in the code that we don't want to run.
2. A human-readable annotation in the code.

```
# Calculate the area of the circle

#r = 1 # Circle radius

r = 2 # Circle radius
```

Variable reassignment

We can reassign an existing variable a new value.

In [2]:

```
a = 1
a = 2
print(a)
```

2

In programming, `a = a + 1` is a valid expression.

If this were a mathematical equation, a value of `a` does not exist that satisfies the equality.

But in programming, an expression of this form is used to *reassign* the value of `a`.

The new value of `a` is the original value of `a` when modified as in the expression on the right hand side.

In [6]:

```
a = 1
a = a + 1
print(a)
```

2

Assignment Operators

The simplest assignment operator is `=` which assigns the value following this operator to a variable.

Assignment operators that compound the `=` operator with any arithmetic operator (e.g. `+=`, `-=`) can be used to re-assign the value of a variable.

In [13]:

```
a = 1
a += 1     # Equivalent to a = a + 1
print(a)
```

2

What will be displayed if we add `print(a)` at the end of this code?:

In [15]:

```
a = 3
a -= 1
```

Strings

Strings behave differently to numerical data.

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

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

In [4]:

```
x = 'Hello'
```

In [5]:

```
print(x[0])   # First letter
```

```
H
```

In [ ]:

```
print(x[4])   # Last letter
print(x[-1])
```

In [ ]:

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

In [ ]:

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

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'


print(a + a)
```

Answer:

> 4

```
a = 2
b = '2'


print(a + b)
```

Answer:

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

Numbers represented as strings are not recognised as numerical values.

```
a = 2
b = '2'


print(b + b)
```

Answer:

> 22

Adding string (text) data *connects* the two strings

Arithmetic Operators behave differently on numerical and non-numerical values.

Most arithmetic operators cannot be used on non-numerical values.

Best Practises - Readability and Tidiness

Alongside the Python syntax, we will be teaching you best practises for programming.

These best practises are taken from a document called PEP 8 (or PEP8/PEP-8), written in 2001, that provides guidelines on how to write Python code.

https://peps.python.org/pep-0008/ (https://peps.python.org/pep-0008/)

The content of PEP 8 focuses on improving the readability and consistency of Python code and is widely adopted by users of Python.

While these practises and conventions are not needed to ensure your code runs, they will help you to write well-presented code.

The readability and tidiness of your code is important for:

- remembering what your code does when you refer back to it
- communicating your code to others (e.g. working as a team, preparing an assignment)
- demonstrating professionalism

You will be marked on your use of these best practises in the assignment for this unit.

Best Practises - Variable Naming

A single naming convention should be used consistently for all variable names in a program.

Several naming conventions exist:

`camelCase`	Each word in the variabale name, except the first, starts with a capital letter
`PascalCase`	Each word in the variable name starts with a capital letter
`snake_case`	Each word in the variable name is lower case and is separated by an underscore character

The most widely used naming convention for writing code in Python is `snake_case` .

We will use `snake_case` for the code examples thoughout these notes and build up our knowledge of naming conventions over the unit.

Demos

Example

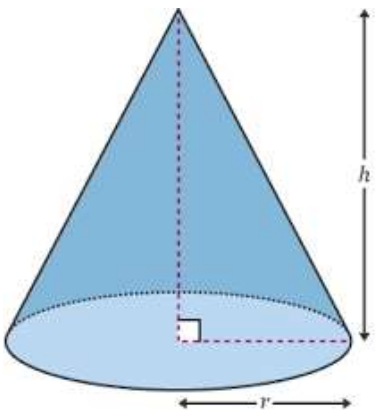
Volume of a cone: $V = \frac{Ah}{3}$

$A =$ area of base

$h =$ height of cone

Find the volume of the cone:

Base radius, r = 2 cm
Height, h = 10 cm
$\pi = 3.142$



In [ ]:

```
```

Example

Create a variable `a` and assign it a `string` value with two or more characters.

Create a variable `b` and assign it a `string` value with three or more characters.

Use an arithmetic operator to connect the **first two** letters of `a` and the **last three** letters of `b` to make a new string.

In [21]:

```
```

Out[21]:

```
'twhon'
```

Tips for running python files when completing the in-class exercises.

To run the code for one exercise at a time, you can either:

- save your answers to each exercise as separate .py (Python) files

or

- use comments `#` to *comment out* the code you do not want to run, leaving only the code for the exercise you are working on uncommented.