

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 [4]:

```
A = 1.0  
  
B = 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
- `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 [29]:

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

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

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

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

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

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.

To comment a whole line, select the line and press `ctrl + 1` in Spyder.

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 [30]:

```
B = 1.0
```

```
E = int(B)
```

```
print(B, E)
```

```
1.0 1
```

Arithmetic Operators

Python can be used like a calculator.

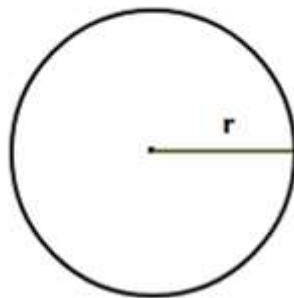
Arithmetic operators (+, -, /, *) are used with numeric values to perform common mathematical operations.

| | |
|----|---|
| ** | Exponent |
| * | Multiplication |
| / | Division |
| // | 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 |
| *, /, //, % | Multiplication, division, floor division, modulo (evaluated left to right) |
| +, - | Addition, subtraction (evaluated left to right) |

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



$$A = \pi r^2$$

In [32]:

```
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)
```

4

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

Error.
Cannot add numerical and non-numerical value

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

22

Adding string (text) data connect the two strings

In [36]:

```
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 [39]:

```
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

Notice the use of the `#` symbol to *comment* parts of the code

(i.e. change which parts of the code are run)

In-class Demos

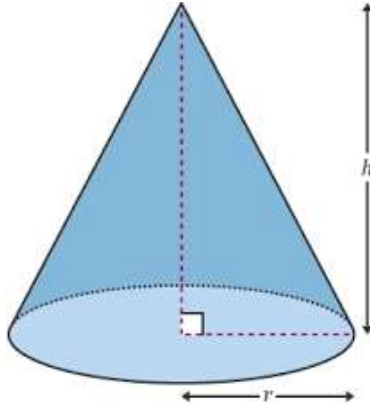
Example 1: Find the volume of a cone.

Base radius = 2 cm

Height = 10 cm

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

where A = base area, h = height of the cone



In []:

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

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

My name is Hemma

In [40]:

```
Name = 'Hemma'
output = 'My name is ' + Name
print(output)
```

My name is Hemma

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