At the end of this lecture, students should be able to:

- import modules and use the functions defined in the module
- use integer division and modulus operators
- use comments. Include a docstring at the top of a program
- use self-documenting code to make the program easy to understand
- understand that an expression evaluates to one value
- understand the order of operations when an expression is evaluated
- understand how to develop a program in steps

### From lecture 2

- perform calculations using standard arithmetic operators
- use variables to store values
- describe differences between int and float types
- print numbers and strings to standard output

```python
amount_to_convert = 500
nz_dollars = amount_to_convert
nz_to_aus_rate = 0.95
```

### Literals and variables

**Literals** are the actual values which can be stored in the program memory, e.g.,
- 34
- -67.5
- "a particular string"

Variables can be assigned any literal value (or an expression). Variables are used to refer to (point to) a piece of information, e.g.,
- result = 567
- final_result = result + 45
- phrase = "a particular string"
- phrase = "Please tell me more"
- first_name = "Izzy"

**Expressions** are made up of literal values and variables. Expressions always evaluate to a single value. The right hand side of the assignment operator is an expression, e.g.,
- number = 3
- final_result = 567 + 16 ** number
- final_result = final_result + number * 5 / 78
Docstrings

A docstring is a special kind of string (text) used to provide documentation. A docstring:
- appears at the top of every COMPSCI 101 program,
- three double-quotes are used to surround the docstring,
- all programs should include a docstring at the beginning of the program,
- the docstring contains the author and a description of what the program does.

```python
###
Program which calculates the area of a circle.
Author: Adriana Ferraro
###
radius = 10
pi = 3.14159265359
area = pi * radius ** 2
print("Area of circle", area)
```

Use self documenting code

- Add comments sparingly to explain code that is difficult, or to tell other programmers something they need to know about the code. It is always important to use good variable names.
- The program below does the same job as the program on the previous slide but it uses very poor variable names which makes the program difficult to understand.

```python
###
Converts a length in inches to a length in centimetres.
Author: Damir Azhar
###
length_in_inches = 100
inches_to_cm = 2.54  # 1 inch = 2.54 cm
length_in_cm = length_in_inches * inches_to_cm
print("Length", length_in_cm)
```

Skeleton of a Python program

- In general the format of a Python program is:

```
###
Calculates the area of a rectangle.
Author: Damir Azhar
###
width = 3.56
height = 8.4
area = width * height
print("Area of rectangle", area)
```

Every Python program is stored in a file which has .py at the end of the file name, e.g., CalculateArea.py, CompoundInterest.py
Expressions – order of operations

Expressions containing numbers are evaluated in the same way as in mathematical expressions, i.e., BEDMAS applies:

- Brackets
- Exponents
- Division, Multiplication
- Addition, Subtraction

Note that the / operator always results in a float, e.g., 8 / 4 is 2.0.

What is the output?

```python
result1 = (25 - 7) * 3 + 12 / 3
result2 = 17 - 3 * 2 - 12 / 4 + 15
result3 = 32 / 4 ** (3 + 2 * 3 - 7) / 5
print(result1, result2, result3)
```

Remember to work from left to right when evaluating operators with the same priority.

More arithmetic operators

- So far, we have seen the following mathematical operators: +, -, *, /, **

- Two more mathematical operators:
  - Floor division (integer division) //
  - Modulus (remainder) %

- Floor division (integer division) performs the division and ignores the part after the decimal point, e.g.,
  - 16 // 5 gives 3
  - 17 // 5 gives 3
  - 34 // 5 gives 6

- Modulus performs the division and gives the remainder, e.g.,
  - 16 % 5 gives 1
  - 17 % 5 gives 2
  - 34 % 5 gives 4
  - 16 % 30 gives 16

Exercise

- Give the output

```python
result1 = 25 % 3
result2 = 20 % 3
result3 = 20 // 3
result4 = 5 // 7
result5 = (25 // 5) % 5
print(result1, result2, result3, result4, result5)
```

Exercise

- Order of operations

- Give the output when the following code is executed

```python
result1 = 25 / 4 // 3 + 4 * 10 % 3
result2 = 10 - 7 // 3 * 3 + 13 % 5 / 5 * 2
result3 = 17 % 3 * 2 - 3 ** 2 * 3 + 19 // 2
print(result1, result2, result3)
```
Python libraries

- Python has libraries of code which contain definitions and functions which perform useful tasks and calculations. The files in these libraries are called modules. The name of a module is the name of the file without the .py extension.
- The `math` module contains many useful math functions and constants, e.g., `math.sin()`, `math.cos()`, `math.pow()`, `math.sqrt()`, `math.floor()` ...
- In order to be able to use the functions of a module, we need to import the module. Importing a module means that we can then use all the functions defined inside that module, e.g.,

```python
### Calculates the radius of a circle, given the area.
Author: Damir Ashar
###
import math
area = 221.67
radius = math.sqrt(area / math.pi)
print("Radius of circle", radius)
```

```
Radius of circle 8.399985266079987
```

Finding the distance between two cities

- We want to find the distance between two cities given the latitude and longitude of the two cities.
- The formula (thank you Internet) for finding the distance between two points on the earth:

\[
distance = 2r \arcsin \left( \sqrt{\sin^2 \left( \frac{\phi_2 - \phi_1}{2} \right) + \cos(\phi_1) \cos(\phi_2) \sin^2 \left( \frac{\lambda_2 - \lambda_1}{2} \right)} \right)
\]

where:
- \( \phi \) is latitude
- \( \lambda \) is longitude
- \( R \) is the earth's radius (6371 km or 3959 miles)

Note that for this formula the angles passed to the trig functions need to be in radians.
Break the problem up into manageable sections

In the math module we can find all the functions we need:

\[
\text{math.sin()}, \quad \text{math.cos()}, \quad \text{math.asin()}, \quad \text{math.sqrt()}, \quad \text{math.radians()}
\]

\[
\text{distance} = 2r \arcsin \left( \sin^2 \left( \frac{\phi_2 - \phi_1}{2} \right) + \cos(\phi_1) \cos(\phi_2) \sin^2 \left( \frac{\lambda_2 - \lambda_1}{2} \right) \right)
\]

```
import math
miles_to_kms = 1.60934
theta_latitude_auckland = math.radians(-36.8404)
theta_latitude_sydney = math.radians(-33.8600)
lambda_longitude_auckland = math.radians(-174.7399)
lambda_longitude_sydney = math.radians(-151.2094)
earth_miles_radius = 3959 #6371 km

#break up the calculation into steps
print(distance_in_miles, "miles")
print(distance_in_kms, "kilometres")
```

Examples of Python features used in this lecture

- import modules and use the functions defined in the module
  ```python
  import math
  result = math.sqrt(345)
  ```

- use integer division and modulus operators
  ```python
  whole_number = 456 // 3
  left_overs = 456 % 12
  ```

- understand the order of operations when an expression is evaluated
  ```python
  result = 32 / 4 ** (3 + 2 * 3 - 7 % 4) / 5
  ```

Summary

- In a Python program we can:
  - import modules and use the functions defined in the imported module
  - use integer division and modulus operators
  - use comments. Every program contains a docstring at the top of the program
  - use self-documenting code to make the program easy to understand
  - understand that an expression evaluates to one value
  - understand the order of operations when an expression is evaluated
  - understand how to develop a program in steps