

Python programs

- Program (or script) is a sequence of **definitions** and **commands**
 - Definitions evaluated and commands executed by Python interpreter in a **shell**
 - Can be typed directly into a shell, or stored in a file that is read into the shell and evaluated
- **Command** (or **statement**) instructs interpreter to do something

Objects

- At heart, programs will manipulate data objects
- Each object has a **type** that defines the kinds of things programs can do to it
- Objects are:
 - **Scalar** (i.e. cannot be subdivided), or
 - **Non-scalar** (i.e. have internal structure that can be accessed)

Scalar objects

- `int` – used to represent integers, e.g., 5 or 10082
- `float` – used to represent real numbers, e.g., 3.14 or 27.0
- `bool` – used to represent Boolean values `True` and `False`
- The built in Python function `type` returns the type of an object

```
>>> type(3)
```

```
<type 'int'>
```

```
>>> type(3.0)
```

```
<type 'float'>
```

Expressions

- Objects and operators can be combined to form **expressions**, each of which denotes an object of some type
- The syntax for most simple expressions is:
 - <object> <operator> <object>

Operators on `ints` and `floats`

- `i + j` – sum – if both are `ints`, result is `int`, if either is `float`, result is `float`
- `i - j` – difference
- `i * j` – product
- `i / j` – division – if both are `ints`, result is `int`, representing quotient without remainder
- `i % j` – remainder
- `i ** j` – `i` raised to the power of `j`

Some simple examples

```
>>> 3 + 5
```

8

```
>>> 3.14 * 20
```

62.8

```
>>> (2 + 3)*4
```

20

```
>>> 2 + 3*4
```

14

Performing simple operations

- Parentheses define sub-computations – complete these to get values before evaluating larger expression
 - $(2+3)*4$
 - $5*4$
 - 20
- Operator precedence:
 - In the absence of parentheses (within which expressions are first reduced), operators are executed left to right, first using $**$, then $*$ and $/$, and then $+$ and $-$

Comparison operators on `ints` and `floats`

- `i > j` – returns `True` if strictly greater than
- `i >= j` – returns `True` if greater than or equal
- `i < j`
- `i <= j`
- `i == j` – returns `True` if equal
- `i != j` – returns `True` if not equal

Operators on bools

- a and b is True if both are True
- a or b is True if at least one is True
- not a is True if a is False; it is False if a is True

Type conversions (type casting)

- We can often convert an object of one type to another, by using the name of the type as a function
 - `float(3)` has the value of `3.0`
 - `int(3.9)` truncates to `3`