Arithmetic Expressions

Using IDLE

Arithmetic Expressions

- Numeric Literals
  - We have two different types of numerical values
    - Integers (positive or negative)
    - Floating point values
      - Written in decimal or scientific notation
      - `5.32e-4`
      - `0.000532`

Arithmetic Expressions

- Python expressions have a “type”
  - Necessary because internal representation differs
  - Most important
    - **int** -- an integer
    - **floa**t -- a floating point number
    - complex -- a complex floating point number
    - **str** -- a string
    - bytes -- a “raw” string of bytes

Arithmetic Expressions

- The keyword type will reveal the type of an expression

```
type(3.14)
<class 'float'>
```

```
type(-12)
<class 'int'>
```

```
type('hello')
<class 'str'>
```

```
type(b'hello')
<class 'bytes'>
```

Arithmetic Expressions

- Arithmetic expressions use operators
  - unitary: - for minus
  - binary: +, -, *,
  - Division: / yields always a floating point
    - `6/2` gives `3.0`
  - “Floor” division: // yields always an integer
    - If necessary, rounded down

```
6//2
3
```

```
-3//4
-1
```

```
3//4
0
```

Arithmetic Expressions

- Arithmetic expressions use operators
  - Exponentiation uses two asterisks
    - Example: Square-root of 0.75
      - `0.75**0.5`
      - `0.8660254037844386`
  - Can yield complex numbers
    - `(-1.25)**0.5`
    - `(6.845983728302534e-17+1.1180339887498`
`95j)`

Arithmetic Expressions

- Binary operations
  - Python knows certain operations on the binary representation of numbers
    - | bit-wise or
    - ^ bit-wise xor
    - & bit-wise and
- Can be very useful, but we will not look at them in this class
- Just be aware of their existence

Arithmetic Expressions

- To combine expressions use
  - Parentheses (...)
  - Precedence rules
    - BODMAS
      - Brackets, Order, Division/Multiplication, Addition/ Subtraction
- Left associativity:
  - `a*b/c` becomes `(a*b)/c`
  - But `a**b**c` is `a**(b**c)`

Expressions with Strings

- A string is a piece of text
- Indicated with quotation marks
- Python allows single, double quotation marks as long as the string does not contain newlines
- Example:
  - `'hello world'`
  - `"Python is cool"`

Expressions with Strings

- We can concatenate strings

```
>>> "hello" + "world"
'helloworld'
```

- And we can multiply a string with a positive integer

```
>>> 'JA'*20
'JAJAJAJAJAJAJAJAJAJAJAJAJAJAJAJAJAJA'
```

Expressions with Strings

- This is interesting:
  - The interpretation of the asterisk `*` depends on the *operands*
- But we cannot multiply a string with a string

```
>>> 'hello'*'world'
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    'hello'*'world'
TypeError: can't multiply sequence by non-int of type 'str'
```

- This results in an error called a “type error”