Types

Thomas Schwarz, SJ

Computer Organization

- Computers store and manipulate data
  - Just as we manipulate scribbles on paper if we calculate with paper and pencil
  - By experience:
    - Let the type of data determine how it is being stored
    - Example:
      - ASCII stores western language characters in 8 bits
      - Unicode stores symbols from all languages and usages in 32 bits
      - 16 b integers stored in 16 bits in early desktops

Types

- In order to manipulate data correctly, we need to know their *type*
- Fundamental types:
  - Integers: `int`
  - Floats: `float`
  - Strings: `str`
    - There is no special type for individual characters
- There are other types, such as `complex, bytes`
  - You can (and eventually should) construct your own types

Types

- If we have an expression, you can find its type by using the `type( )` function

```
>>> type(3141)
<class 'int'>
>>> type(3.141)
<class 'float'>
>>> type('3141')
<class 'str'>
>>> type(b'3141')
<class 'bytes'>
>>>
```

Type Conversion

- Certain abstract values can be stored in different forms
  - E.g. π is approximately 3.141592653589793.
  - The right side can be a floating point number or a string
  - To obtain π, we import the math module
    - `import math`
  - Then we use `math.pi`

Type Conversion

- Example continued

```
IDLE Shell 3.1
Python 3.10.1 (v3.10.1:2cd268a3a9, Dec  6 2
300.0.29.3)] on darwin
Type "help", "copyright", "credits" or "lic
>>> import math
>>> math.pi
3.141592653589793
>>>
```

Type Conversion

- Obviously, `math.pi` should be a float

```
>>> import math
>>> math.pi
3.141592653589793
>>> type(math.pi)
<class 'float'>
>>>
```

Type Conversion

- We can convert π to a string
  - Looks the same but is different under the hood

```
>>> type(str(math.pi))
<class 'str'>
>>> str(math.pi)
'3.141592653589793'
>>>
```

Type Conversion

- We can convert a floating point number to an int using the `int` function

```
>>> int(45.0)
45
>>> int(45.1)
45
>>> int(-45.1)
-45
>>>
```

- If the floating point does not correspond to an integer, than there is rounding to the nearest integer
  - down for positive and up for negative numbers

Type Conversion

- We can create a float out of a string

```
>>> float('3.145')
3.145
```

- As well as an integer

```
>>> int('-45')
-45
>>>
```

Type Conversion

- But the conversion can fail, resulting in a Value Error
  -

```
>>> int('fourty five')
Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module>
    int('fourty five')
ValueError: invalid literal for int() with base 10: 'fourty five'
```

Type Conversion

- Thus:
  - To convert to a floating point:
    - Use `float`
  - To convert to an integer:
    - Use `int`
  - To convert to a string:
    - Use `str`

Type Conversion

- Some type conversions are automatic
  - If we use the print function:
    - Argument(s) are converted automatically to strings

Type Conversion

- Resumen:
  - All data processed by a computer has a type
  - In Python, we can convert to different types