Course 4+

POO

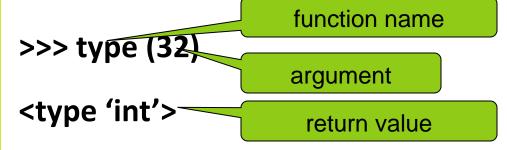
revision PROCEDURAL PROGRAMMING

CHAP. 3 FUNCTIONS -- CONTENTS

- Function calls
- Type conversion functions
- Maths functions
- Importing with from
- © Composition

Functions calls

- A function is a named sequence of statements that performs a computation.
- When you define a function, you specify:
 - the function name and
 - the sequence of statements.
- Later, you have to call the function by name in order to use it.



Type conversion functions

- Python provides built-in functions that convert values from one type to another.
- The int function takes any value and converts it to an integer, if it can, or complains otherwise:
- >>> int('32')
- 32
- >>> int('Hello')
- ValueError: invalid literal for int(): Hello
- >>> int(3.99999)
- 3
- >>> int(-2.3)
- <u>-2</u>

Type conversion functions

- Python provides built-in functions that convert values from one type to another.
- The float function converts integers and strings to floating-point numbers:
- >>> float(32)
- 32.0
- >>> float('3.14159')
- 3.14159

Type conversion functions

- Python provides built-in functions that convert values from one type to another.
- The str function converts its argument to a string:
- >>> str(32)
- '32'
- >>> str(3.14159)
- '3.14159'

Maths functions

- Python has a math module that provides most of the familiar mathematical functions.
- A module is a file that contains a collection of related functions.
- Before we can use the module, we have to import it:
- >>> import math
- This statement creates a module object named math. The module object contains the functions and variables defined in the module.
- To access one of the functions, you have to specify the name of the module and the name of the function, separated by a dot (also known as a period). This format is called **dot notation**.
- >>> ratio = signal_power / noise_power
- >>> decibels = 10 * math.log10(ratio)
- >>> radians = 0.7
- >>> height = math.sin(radians)

Composition

One of the most useful features of programming languages is their ability to take small building blocks and compose them.

```
>>> x = math.sin(degrees / 360.0 * 2 * math.pi)
>>> x = math.exp(math.log(x+1))
>>> minutes = hours * 60  # right
>>> hours * 60 = minutes  # wrong!
SyntaxError: can't assign to operator
```

Importing with from

Python provides two ways to import modules

>>> import math

>>> print math.pi

3.14159265329

>>> from math import pi

>>> print pi

3.14159265329

>>> from math import *

>>> print pi

 $>>> x = \exp(\log(2))$

- The advantage of importing everything from the math module is that your code can be more concise.
- The disadvantage is that there might be conflicts between names defined in different modules, or between a name from a module and one of your variables.