Built-In Functions

Introduction

- Predefined functions provided by Python.
- Good example of a well known built-in function, that everyone is familar with: **print**
- Why is it called built-in?
  - Can use them directly, without the need to **import**
  - What is import? will be covered later
- All these built-in functions are available in the **builtins** module
  - automatically loaded in the background
  - use: **print(dir(__builtins__))** to see all the commands

Type Conversion Functions

- To convert from one type to another
- int("5")   # convert string 5 to an integer
- float("3.14")   # convert float 3.14 to floating point number
- bool(0) # convert integer 0 to boolean False

Inspection Functions

- To check the type, identity, attributes of variables.
- type(10)   # check the type. output: <class 'int'>
- id("abc")   # get the memory address
- isinstance(10, int) # check if 10 is an integer. output: True
- dir(str)   # list the attributes/methods of an object
- help(str)   # get or display the docstring/help

Math Functions and More

- abs(-4)   # find the absolute value, output: 4
- sum([1, 2, 3]) # find the sum of the elements in the input list. output 6
- min(3, 1, 4)   # find the minimum value. output: 1
- max(3, 1, 4)   # find the maximum value. output: 4
- round(3.14159, 2) # find the rounded value upto 2 decimal places. output: 3.14
- pow(2, 3)   # computes the 2 raised to 3. output: 8
- divmod(10, 3) # computes the integer division and modulus at the same time. output: (3, 1)
- len("hello")   # find the length of the string, output: 5
- range(5)   # create a list from 0 to 4

Any time you enclose a set of numbers or characters, seperated by comma, and enclosed in square braces, you are creating a list. More about Lists in Module 3.

Execution Functions

- eval()
  - Evaluates a Python **expression** string
  - Returns the result.
- exec()
  - Executes dynamic statements (**code blocks**), including assignments, loops, function definitions, etc
  - Does not return anything.
- Dangerous to use with untrusted input.

```
x = 10
result = eval("x + 5")
# output: 15
```

```
code = """
a = 5
b = 10
print("Sum is:", a + b)
"""
exec(code)
```