CSC101 – Introduction to ICT
Lecture 05
Mathematical Functions & Strings

Course Instructors

Dr. Majid Iqbal **Dr. Ashfaq Hussain Farooqi** **Ms. Sajida Kalsoom** **Mr. Rizwan Rashid**

Lecture Outline

- Common Python Functions
- Mathematical Functions
- Strings
- Formatting Numeric and String Data

Common Python Functions [1]

Python provides many useful functions for common programming tasks.

- A ***function*** is a group of statements that performs a specific task.
- Python, as well as other programming languages, provides a library of functions.
- You have already used the functions **eval**, **input**, **print**, and **int**.
  - These are built-in functions and they are always available in the Python interpreter.

Common Python Functions [2]

TABLE 3.1 Simple Python Built-in Functions

Function	*Description*	*Example*
`abs(x)`	Returns the absolute value for `x`.	`abs(-2)` is `2`
`max(x1, x2, ...)`	Returns the largest among `x1`, `x2`, ...	`max(1, 5, 2)` is `5`
`min(x1, x2, ...)`	Returns the smallest among `x1`, `x2`, ...	`min(1, 5, 2)` is `1`
`pow(a, b)`	Returns a^b. Same as `a ** b`.	`pow(2, 3)` is `8`

`round(x)`	Returns an integer nearest to `x`. If `x` is equally close to two integers, the even one is returned.	`round(5.4)` is `5`
		`round(5.5)` is `6`
		`round(4.5)` is `4`

`round(x, n)`	Returns the float value rounded to `n` digits after the decimal point.	`round(5.466, 2)` is `5.47` `round(5.463, 2)` is `5.46`

Common Python Functions [3]

```
>>> abs(-3) # Returns the absolute value
3
>>> abs(-3.5) # Returns the absolute value
3.5
>>> max(2, 3, 4, 6) # Returns the maximum number
6
>>> min(2, 3, 4) # Returns the minimum number
2
>>> pow(2, 3) # Same as 2 ** 3
8
>>> pow(2.5, 3.5) # Same as 2.5 ** 3.5
24.705294220065465
>>> round(3.51) # Rounds to its nearest integer
4
>>> round(3.1456, 3) # Rounds to 3 digits after the decimal point
3.146
```

Mathematical Functions

To use mathematical functions, import math module in your program
import math

Function	*Description*	*Example*
`fabs(x)`	Returns the absolute value for `x` as a float.	`fabs(-2)` is `2.0`
`ceil(x)`	Rounds `x` up to its nearest integer and returns that integer.	`ceil(2.1)` is `3` `ceil(-2.1)` is `-2`
`floor(x)`	Rounds `x` down to its nearest integer and returns that integer.	`floor(2.1)` is `2` `floor(-2.1)` is `-3`
`exp(x)`	Returns the exponential function of `x` (e^x).	`exp(1)` is `2.71828`
`log(x)`	Returns the natural logarithm of `x`.	`log(2.71828)` is `1.0`
`log(x, base)`	Returns the logarithm of `x` for the specified base.	`log(100, 10)` is `2.0`
`sqrt(x)`	Returns the square root of `x`.	`sqrt(4.0)` is `2`
`sin(x)`	Returns the sine of `x`. `x` represents an angle in radians.	`sin(3.14159 / 2)` is `1` `sin(3.14159)` is `0`

Mathematical Functions

Function	*Description*	*Example*
`asin(x)`	Returns the angle in radians for the inverse of sine.	`asin(1.0)` is `1.57` `asin(0.5)` is `0.523599`
`cos(x)`	Returns the cosine of `x`. `x` represents an angle in radians.	`cos(3.14159 / 2)` is `0` `cos(3.14159)` is `-1`
`acos(x)`	Returns the angle in radians for the inverse of cosine.	`acos(1.0)` is `0` `acos(0.5)` is `1.0472`
`tan(x)`	Returns the tangent of `x`. `x` represents an angle in radians.	`tan(3.14159 / 4)` is `1` `tan(0.0)` is `0`
`degrees(x)`	Converts angle `x` from radians to degrees.	`degrees(1.57)` is `90`
`radians(x)`	Converts angle `x` from degrees to radians.	`radians(90)` is `1.57`

Mathematical Functions - Example

The area of a pentagon can be computed using the following formula (s is the length of a side)

$$Area = \frac{5 \times s^2}{4 \times \tan\left(\frac{\pi}{5}\right)}$$

```
>>>import math
>>> s=5.5
>>> area=(5*math.pow(s,2))/(4 * math.tan(math.pi/5))
>>> area
52.04444136781625
```

Strings and Characters

A string is a sequence of characters. Python treats characters and strings the same way.

- *String* values must be enclosed in matching *single quotes* (') or *double quotes* (").
- Python does not have a data type for characters.
- A single-character string represents a character. For example,

```
letter = 'A' # Same as letter = "A"
numChar = '4' # Same as numChar = "4"
message = "Good morning" # Same as message = 'Good morning'
```

Character encoding – ASCII Code

- Mapping a character to its binary representation is called *character encoding*.
- There are different ways to encode a character.
  - The manner in which characters are encoded is defined by an encoding scheme.
- ASCII Code (American Standard Code for Information Interchange)
  - 7 bit Encoding Scheme

Character encoding: Unicode code

- Python also supports Unicode.
- Unicode is an encoding scheme for representing international characters.
- ASCII is a small subset of Unicode.
- A Unicode starts with **\u**, followed by four hexadecimal digits that run from **\u0000** to **\uFFFF**.

```
print('\u6B22\u8FCE')
```

→ 欢迎

The **ord()** and **`chr()`** Functions

- **ord(ch) function:** returns the ASCII code for the character **ch**
- **chr(code) function:** returns the character represented by the code.

```
>>> ch = 'a'
>>> ord(ch)
97
>>> chr(98)
'b'
>>> ord('A')
65
>>>
```

Escape Sequences for Special Characters

- A *backslash* (\) followed by a letter is called an *escape sequence*.

Output required as → He said, "John's program is easy to read"

Python Statement → print("He said, "John's program is easy to read" ")

Python Statement with escape characters

```
print("He said, \"John's program is easy to read\" ")
```

```
print("Name\t RegNo.")
```

```
Name        RegNo.
```

Escape Sequences for Special Characters

Character Escape Sequence	Name
\b	Backspace
\t	Tab
\n	Linefeed
\r	Carriage Return
\’	Single Quote
\”	Double Quote
\\	Backslash

The String Concatenation Operator (+)

- The **+** operator can be used to concatenate two strings.
- The augmented assignment += operator can also be used for string concatenation

Example-1

```
>>> message = "Welcome " + "to " + "Python"
>>> message
'Welcome to Python'
>>> chapterNo = 3
>>> s = "Chapter " + str(chapterNo)
>>> s
'Chapter 3'
```

Example-2

```
>>> message = "Welcome to Python"
>>> message
'Welcome to Python'
>>> message += " and Python is fun"
>>> message
'Welcome to Python and Python is fun'
>>>
```

Formatting Numbers and Strings

- Use the `format()` function
- Format function return **String**

Syntax:

format(item, format-specifier)

number or a string

string that specifies how the item is formatted

Example-1:

```
>>>format(1234.6789, '.2f')
'1234.68'
>>>
```

Interest_Example

```
>>> amount = 12618.98
>>> interestRate = 0.0013
>>> interest = amount * interestRate
>>> print("Interest is", interest)
Interest is 16.404674
```

```
print("Interest is",  format(interest,".2f"))
```

```
Interest is 16.40
```

Formatting Floating-Point Numbers

- If the item is a float value, you can use the specifier to give the **width** and **precision** of the format in the form of **width.precision**
  - **width** specifies the width of the resulting string
  - **precision** specifies the number of digits after the decimal point
  - **f** specifies that the data type of the number we are formatting(Conversion Code)

```
print(format(57.467657, "10.2f"))
print(format(12345678.923, "10.2f"))
print(format(57.4, "10.2f"))
print(format(57, "10.2f"))
```

square box (□) denotes a blank space

Formatting as a Percentage

- Use the conversion code % to format a number as a percentage

```
print(format(0.53457, "10.2%"))
print(format(0.0033923, "10.2%"))
print(format(7.4, "10.2%"))
print(format(57, "10.2%"))
```

```
|<------ 10 ------>|
□□□□ 53.46%
□□□□□ 0.34%
□□□ 740.00%
□□ 5700.00%
```

The format 10.2% causes the number to be multiplied by 100 and displayed with a % sign following it.

Justifying Format

- By default, the format of a number is right justified
- < for left justified
- > for right justified

```
print(format(57.467657, "10.2f"))
print(format(57.467657, "<10.2f"))
```

Formatting Integers

- Conversion codes **d, x, o, and b** can be used to format an integer in decimal, hexadecimal, octal, or binary

```
print(format(59832, "10d"))
print(format(59832, "<10d"))
print(format(59832, "10x"))
print(format(59832, "<10x"))
```

```
|←—— 10 ——→|
□□□□□ 59832
59832
□□□□□ e9b8
e9b8
```

Formatting Strings

- Conversion code **s** to format a string with a specified width

```
print(format("Welcome to Python", "20s"))
print(format("Welcome to Python", "<20s"))
print(format("Welcome to Python", ">20s"))
print(format("Welcome to Python and Java", ">20s"))
```

```
|←———— 20 ————→|
Welcome to Python
Welcome to Python
□□□ Welcome to Python
Welcome to Python and Java
```

Summary

- Python Built-in and Mathematical Functions
- String Encoding Scheme and functions
- Format() Function to format Strings and Numbers

Next Lecture

- Selection Structure
- Boolean Expression
- Different Types of IF-Statements