# CS 38003 PYTHON PROGRAMMING

Ruby Tahboub

## BASICS

## PROGRAMMING LANGUAGES

- Programming Languages can be classified into:
  - High-level languages: designed to be used and understood by humans.
  - Low-level language: computer hardware can only understand a very low level language known as machine language.
- High-level languages need to be translated into machine language that the computer can execute.

## COMPILING VS. INTERPRETING

- Compilers convert programs written in a high-level language into the machine language of some computer.
- An interpreter analyzes and executes the source code instruction by instruction.
  - The source program is not translated into machine language all at once.

## COMPILING VS. INTERPRETING

- Compiling
  - Once program is compiled, it can be executed over and over without the source code or compiler.
  - Compiled programs generally run faster since the translation of the source code happens only once.
- Interpreting
  - the source code and interpreter are needed each time the program runs.
  - Programs developed in interpreted languages could be more flexible since they are developed and run interactively.

## PYTHON

- Python is a general purpose Language. Created by Guido Van Rossum in 1990.
- It is high-level, dynamic, object-oriented and multiplatform.
- It is one of the few scripting languages that has been used successfully in large projects.
- It offers flexibility, elegance and power.

## WHY PYTHON?

- Python programs are more compact than other languages because:
  - High-level data types allow complex operations in a single statement.
    - No variable declaration is necessary.
      - Rich, built-in collection types: Lists, Tuples, Dictionaries, etc.
- It is very useful in fast prototyping.
- It receives massive support by the community by providing various useful libraries.
- Makes programmers focus more on the application rather than coding.

## WORKING WITH PYTHON

- To open Python interpreter on a shell:

```
$ python
Python 3.6.9 (default, Jul 17 2020, 12:50:27)
[GCC 8.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

- To run a script via command line:

```
$ python filename.py
```

- To open Jupyter notebook

```
$ jupyter notebook
```

## ELEMENTS OF PROGRAMS

- Names (identifiers): Names are given to variables, functions, etc.
- Every identifier must begin with a letter or underscore ("_"), followed by any sequence of letters, digits, or underscores.
- Case sensitive.
- Reserved keywords should not be used as identifiers, e.g., `print, and, del, for, if, raise, ..., etc.`
- No data type is needed when declaring a variable.
- No return type is needed when declaring a function.

Operators: `+- * / % ** // << >> & | ^ ~ ..., etc.`

## LITERALS

- `456` integer
- `3.25` float
- `45j` Imaginary
- `'String'` String Literal
- `"String"` String Literal that can contain '
- `"""String"""` String Literal that can contain " and '

## SIMPLE PROGRAM

```python
# reading the temp from the user and converting it to a float
c = float(input('Enter temp: '))

# converting the temp from c to f
f = (9.0/5.0 * c) + 32

# printing out both temps
print('Temp in c = ', c, 'Temp in f = ', f)
```

```
Enter temp: 37.5

Temp in c = 37.5 Temp in f = 99.5
```

## DATA TYPES

## BASIC PYTHON DATA TYPES

- Integer : A whole number: `2, 10.`
  - Use `int(x)` to convert x to integer.

Float : Representing the decimals: `1.5, 10.3.`

  - Use `float(x)` to convert x to float.
  - String : Any sequence of characters enclosed by string quotes: `5', "abc".`
    - Use `str(x)` to convert x to string.

Boolean : Takes only two values: `True, False.`

  - Use `bool(x)` to convert x to boolean.
  - Containers:
    - That hold any data type: Lists `[1, 2, 3, 'a', 'xyz']`
- User defined data types: Classes

## BASIC PYTHON DATA TYPES

`input("prompt message")` is used to read data from user

```
>>> applicant = input("Enter the applicant's name: ")
Enter the applicant's name: John

>>> interviewer = input("Enter the interviewer's name: ")
Enter the interviewer's name: Emily

>>> time = input("Enter the appointment time: ")
Enter the appointment time: 2:00 PM

>>> print (interviewer, "will interview", applicant, "at", time)
Emily will interview John at 2:00 PM
```

The default data type of input data is String

## PRINTING

- Printing to console is done by using the function print
- The data type of `print` output is String

```
>>> print (123)
123

>>> print ('123')
123

>>> x = 10
>>> print ('x = ', x)
x =  10

>>> print (2+6)
8

>>> print (50)
50
```

## ARITHMETIC OPERATIONS

- Arithmetic operators inherit their definitions from the data types of operands e.g., `int`, `floating point`.
- Operations on `float` produce `float`.
- Operations on `int` produce `int` (except for /).
- What if one operand is `int` and the other is `float`?

## ARITHMETIC OPERATIONS

```
>>> 7.5/2
3.75
>>> 5/2
2.5
>>> 5.0/2
2.5
>>> 5.0//2
2.0
>>> 5//2
2
>>> 10%3
1
>>> 10.0%3
1.0
```

## DATA TYPE CONVERSION

```
>>> sname = input("Enter the student's name: ")
Enter the student's name: Ally
>>> print ('name: ', sname, type(sname))
name:  Ally <class 'str'>

>>> grade1 = input('Enter grade1: ')
Enter grade1: 10
>>> print ('grade1: ', grade1, type(grade1))
grade1:  10 <class 'str'>

>>> grade1_int = int(grade1)
>>> print ('grade1_int: ', grade1_int, type(grade1_int))
grade1_int:  10 <class 'int'>
```

## DATA TYPE CONVERSION

```
>>> grade2 = input('Enter grade2: ')
Enter grade2: 10.5

>>> grade2_float = float(grade2)
>>> print ('grade2_float: ', grade2_float) type(grade2_float))
grade2_float:  10.5 <class 'float'>

>>> grade2_int = int(grade2_float)
>>> print ('grade2_int: ', grade2_int, type(grade2_int))
grade2_int:  10 <class 'int'>

>>> total = grade1_int + grade2_float
print('total: ', total, type(total))
total:  20.5 <class 'float'>
```

## EXPRESSIONS

- An expression is a combination of one or more constants, variables, operators and functions.
- Standard mathematical precedence rules apply.

```
>>>  ((2**4) + 4) + 2
22
>>>  4 / 2 ** 2
1.0
```

## BOOLEANS

- Booleans are truth values.
- `False` an expression that is untrue.
- `True` an expression that is true.
- All `None` values are considered `False` by Python.
- `None` is the equivalent of 'null' in Java or C.
- Any other value (`Int`, `Float`, `Bool` or `String`) is `True`.

## EXAMPLES

```
>>> bool(0)
False
>>> bool(1)
True
>>> bool(123)
True
>>> bool(0.5)
True
>>> bool(-1)
True
```

```
>>> bool('a')
True
>>> bool('')
False
>>> bool([])
False
>>> bool({})
False
>>> bool(None)
False
```

## BOOLEAN EXPRESSIONS

- `==` equality
- `!=` inequality
- `>` greater than
- `>=` greater than or Equal
- `<` less than
- `<=` less than or

## COMBINING BOOLEAN EXPRESSIONS

- Python provides several logical operators for conditions.
  - `and` (all must be `True`)
  - `or` (either must be `True`)
  - `not` (must not be `True`)
- `xor` (ONLY one must be `True`)
- These operators allow combining different conditions together.
- Bit operations similar are similar to java' s:
  - `&` (and), | (or), ~(not)

# THANK YOU!