ISE 101 – Introduction to Information Systems

• Lecture 2 Objectives:
  – Variable types
  – Boolean algebra
  – Branches
Python Variable Types

- Numeric data types
  - Integers
  - Floating point numbers (floats)
  - Complex numbers
Python Variable Types

• Integers
  Represents negative and positive integers without fractional parts
  -2   24   -5    -22

• Floating point numbers (floats)
  Represents negative and positive numbers with fractional parts
  2.45  -4.47  4.0  8e3  4e9  -3e-4
Python Variable Types

• Even if the fractional part is zero, the number is a floating point number
  3.0 \rightarrow float
  -12.0 \rightarrow float

• Floating point numbers can be represented by scientific notation
  \( a \times 10^b \rightarrow aE_b \)

  23,000 can be represented as 23E3 or 2.3E4 or 0.23E5 ...
  -0.0001 can be represented as -1E-4 or -10E-5 ...
Python Variable Types

• Complex numbers have real and imaginary components

\[ x + yj \]

-3+5.4j

• In math imaginary part is denoted with “i”
• In engineering, “j” is used (as i typically represent current)

• Complex number will be covered in your Calculus courses
• We will not deal with complex numbers in this course
Python Variable Types

• Strings
  Array of characters used with single quotes or double quotes
  “This is a string”
  ‘This is also a string’

• Triple quotes can be used for multiple line strings
  """ This is a multiple line string example and it can extend to many lines""
  """
Python Variable Types

• Boolean (bool)
  Represents variables that can only take 2 values
  - True
  - False

• Other variable types that we will use later
  – Byte
  – List
  – Tuple
  – Dictionary
Python Variable Types

• In order to learn the type of a variable use

```
>>> type('test input')
<class 'str'>
>>> type(3.4)
<class 'float'>
>>> type(3)
<class 'int'>
>>> type(-3.2E3+4j)
<class 'complex'>
>>> var=True
>>> type(var)
<class 'bool'>
```
Python Variable Types

• In other programming languages, a variable has to be declared before using
• For example in C,
  – type of variables are required to be declared before using it
  – once declared, the type of a variable cannot be changed
• No need to declare variable type in Python
• Variable types are assigned when you assign a value to a variable
• Types of variables can change if you assign a value of another type
Python Variable Types

• Variable types are assigned when you assign a value to a variable

```python
>>> x=3.2
>>> type(x)
<class 'float'>
```

• Types of variables can change if you assign a value of another type

```python
>>> x=3.2
>>> type(x)
<class 'float'>
>>> x=True
>>> type(x)
<class 'bool'>
```
Boolean Algebra

- Boolean algebra is defined on set {True, False}
- Three logical operators are defined on this set
  - and
  - or
  - not
- Operator precedence from high to low is
  Parenthesis $\rightarrow$ not $\rightarrow$ and $\rightarrow$ or
- Result of any logical expression is again either True or False
NOT

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>not</td>
<td>False</td>
<td>= True</td>
</tr>
<tr>
<td>Not</td>
<td>True</td>
<td>= False</td>
</tr>
</tbody>
</table>

• Complement is denoted by (..)’ sign
• For example
  not (3 < 5) = False

not (3.4 < 2) = True
**AND**

<table>
<thead>
<tr>
<th></th>
<th>and</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>and</td>
<td>False</td>
<td>=</td>
</tr>
<tr>
<td>False</td>
<td>and</td>
<td>True</td>
<td>=</td>
</tr>
<tr>
<td>True</td>
<td>and</td>
<td>False</td>
<td>=</td>
</tr>
<tr>
<td>True</td>
<td>and</td>
<td>True</td>
<td>=</td>
</tr>
</tbody>
</table>

- For example
  \((3 < 5) \text{ and } (4 > 6) = \text{ False}\)

  \((3 < 5) \text{ and } (4 < 6) = \text{ True}\)
OR

<table>
<thead>
<tr>
<th></th>
<th>or</th>
<th></th>
<th>=</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td></td>
<td>False</td>
<td></td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>or</td>
<td>True</td>
<td>=</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td></td>
<td>False</td>
<td>=</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>or</td>
<td>True</td>
<td>=</td>
<td>True</td>
</tr>
</tbody>
</table>

- For example
  
  \((3 < 5) \text{ or } (4 > 6) = True\)
  
  \((3 < 5) \text{ or } (4 < 6) = True\)
De Morgan Theorem

• De Morgan theorem is used to complement expressions
• To complement an expression using De Morgan theorem
  – Complement all variables in the function
  – Replace all and operators with or
  – Replace all or operators with and
  – Replace all True statements with False
  – Replace all False statements with True
• Examples:
  \[ f(x,y) = (x' \text{ and } y) \text{ or } (y' \text{ and } x) \text{ then } \]
  \[ f'(x,y) = (x \text{ or } y') \text{ and } (y \text{ or } x') \]
De Morgan Theorem

• Examples:
  \[ f(x, y) = (x' \text{ and } y) \text{ or True} \]
  then
  \[ f'(x, y) = (x \text{ or } y') \text{ and False} \]

• Example
  \[ f(a, b, c) = (a \text{ and } b \text{ and } c) \text{ or } (a' \text{ and } b) \]
  then
  \[ f'(a, b, c) = (a' \text{ or } b' \text{ or } c') \text{ and } (a \text{ or } b') \]
Expression Readability

• not $3 < 8$ and $1 > 7$ or $44 < 12.4$ = ?  
  False
• This expression is not readable
• Use more parenthesis and spaces to make it readable

  \[
  ( (not (3 < 8)) \text{ and } (1 > 7) ) \text{ \ or \ } (44 < 12.4) \]
  
  False

  not( ( (3 < 8) \text{ and } (1 > 7) ) \text{ \ or \ } (44 < 12.4) )
  
  True
Value Comparisons in Python

- In Python there are 6 ways of comparing numbers

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;</td>
<td>greater than</td>
</tr>
<tr>
<td>&lt;</td>
<td>smaller than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>greater than or equal to</td>
</tr>
<tr>
<td>&lt;=</td>
<td>less than or equal to</td>
</tr>
<tr>
<td>==</td>
<td>equal to</td>
</tr>
<tr>
<td>!=</td>
<td>not equal to</td>
</tr>
</tbody>
</table>
Why Use "=="?

• Remember that = means assign in Python (and in many other programming languages)
  
i = i + 1

• For checking equality of two numbers or expressions, "==" is used. This generates a Boolean output (True or False)

• Examples
  
3 == 5  False
3 == 3.0  True
3+0j == 3  True
Why Use “==“?

- “==“ can be used to check equality of Boolean and string type

Example

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>‘aaa’ == ‘aaa’</td>
<td>True</td>
</tr>
<tr>
<td>False == False</td>
<td>True</td>
</tr>
<tr>
<td>False == (5&lt;3)</td>
<td>True</td>
</tr>
<tr>
<td>True == (5!=3)</td>
<td>True</td>
</tr>
</tbody>
</table>
\textbf{Branches}

- Until now, all Python scripts flows from top to bottom without any branching

```python
tmp = input("Enter temperature in Celcius: ")
celcius = float(tmp)
fahrenheit = 9*celcius/5 + 32
print("Celcius: " + str(celcius) + "= Fahrenheit: " + str(fahrenheit))
```

- Sometimes we have perform different actions depending on the situation
Branches

• Write a Python script that
  – gets the temperature from the user,
  – converts it to fahrenheit
  – Displays the result
  – Displays a warning “too hot” if the temperature is greater than 100 fahrenheit
Flow Diagram

Get temperature from user

Convert the temperature to fahrenheit

Display the temperature in fahrenheit

Temperature > 100 ?

True

Display “too hot”

False

Display the temperature in fahrenheit
tmp = input("Enter temperature in Celcius: ")

celcius = float(tmp)
fahrenheit = 9*celcius/5 + 32

print("Celcius: " + str(celcius) + " = Fahrenheit: " + str(fahrenheit))

if (fahrenheit > 100):
    print('too hot')
If Statement

- Sometimes the script should perform different actions when a logical expression is correct and incorrect
- Write a Python script that
  - Gets midterm and final grades from the user
  - Computes the weighted average (40% midterm, 60% final)
  - Display the weighted average and
  - Display
    - ‘Passed’ if the average is greater or equal to 50
    - ‘Failed’ if the average is less than 50
Flow Diagram

1. Get midterm & final grades from user
2. Compute the weighted average
3. Display the average
4. Check if the average is greater than or equal to 50
   - True: Display "passed"
   - False: Display "failed"
Branches

```python
midterm_grade = float(input("Enter midterm grade: "))
final_grade = float(input("Enter final grade: "))

average_grade = 0.40 * midterm_grade + 0.60 * final_grade

print("Average: " + str(average_grade))

if (average_grade >= 50):
    print('Passed')
else:
    print('Failed')
```
IF ... ELSE Statement

• “if” is used to generate branches in the flow of the script
• Structure

```python
if (logical_statement):
    ...
    ...
else:
    ...
    ...
```

A logical statement is required after the if statement
Colon (:) is required after the if and else statements

These commands will be executed if the logical_statement produces True
These commands will be executed if the logical_statement produces False
IF ... ELSE Statement

• The scope in Python is determined using Tab (NOT white space)
• In C/C++, paranthesis are used for this purpose
Scope

- If the logical expression after the “if” statement produces True, all instructions within the scope of “if” will be sequentially executed.

- If the logical expression after the “if” statement produces False, all instructions within the scope of “else” will be sequentially executed.
Example

• Write a Python script that
  – generates a random integer number between 1 and 10
  – gets a guess from the user
  – if the guess is equal to the random number display ‘correct’
  – if the guess is greater than the random number display ‘My number is smaller than your guess’
  – if the guess is smaller than the random number display ‘My number is greater than your guess’
import random

my_number = random.randint(1, 10)

user_guess = int(input('Guess my number: '))

if (user_guess == my_number):
    print('Correct')

if (user_guess > my_number):
    print('My number is greater than your guess')
else:
    print('My number is smaller than your guess')
Flow Diagram

- Generate random integer
- Get a guess from the user
- If number = guess?
  - Display "correct"
  - If number > guess
    - Display "greater"
  - Display "smaller"
Example

```python
import random

my_number = random.randint(1, 10)

user_guess = int(input('Guess my number: '))

if (user_guess == my_number):
    print('Correct')
elif (user_guess > my_number):
    print('My number is greater than your guess')
else:
    print('My number is smaller than your guess')
```
Example

- Write a Python script that
  - gets 3 floating point numbers from the user
  - finds the maximum of the 3 numbers
  - display it

- This script can be written in many different ways
- Prefer the simplest method
Example

```python
number1 = float(input('Enter first number: '))
number2 = float(input('Enter second number: '))
number3 = float(input('Enter third number: '))

# compare the first two numbers and assign the larger one to a variable max_number
if number1 >= number2:
    max_number = number1
else:
    max_number = number2

# compare the max_number with number3
# is number3 is greater than max_number
# assign number3 to max_number
if number3 > max_number:
    max_number = number3

# at this point, max_number is the largest of the three numbers
print('Largest number: ' + str(max_number))
```
Example

• Write a Python script that
  – gets 2 numbers from the user
  – Prints the following menu choice
    (1) Addition'
    (2) Subtraction'
    (3) Multiplication'
    (4) Division'
  – Gets the user’s choice
  – Perform the corresponding operation and display its result
  – If the user enters a wrong choice then warn the user
Example

```python
number1 = float(input('Enter first number: '))
number2 = float(input('Enter second number: '))

print('(1) Addition')
print('(2) Subtraction')
print('(3) Multiplication')
print('(4) Division')
choice = int(input('Choose the operation: '))

if choice == 1:
    print('Addition ' + str(number1 + number2))
elif choice == 2:
    print('Subtraction ' + str(number1 - number2))
elif choice == 3:
    print('Multiplication ' + str(number1 * number2))
elif choice == 4:
    print('Division ' + str(number1 / number2))
else:
    print('Wrong choice')
```