ISE 101 – Introduction to Information Systems

- Lecture 1 Objectives:
  - General information on Python programming language
  - Identifier names
  - Printing information
  - Getting input from users

Programming Languages

- Programming Languages
  - C
  - C++
  - Java
  - Perl
  - C#
  - PHP
  - Python
  - Pascal
  - …

Each programming language has its own “syntax”

These languages are called high level computer languages

Hardware can only undestand and execute machine code

•Software development in machine code is really hard (Opcodes – assembly)
•Software in machine code is fast

Programming Languages

- Programs written with high level computer languages need to be translated into machine language
- Machine language depends on the CPU type
- Programs written in machine code for Intel CPU do not work with PowerPC CPU or any other.
- Software written in machine code is highly hardware dependent → No portability.
- However, they are fast.
- Each high level programming language can be
  - Compiled
  - Interpreted

  to translate into machine code

Compile

- A compiler is a complex computer program that takes another program written in a high-level language and translates it into machine code
- High level program is called “source code”
- The output of compiling is a machine code program that the computer can directly execute
- Not as fast as programs written directly in machine code, but can still be considered fast.
- Source code is portable but it has to be compiled for each hardware platform
- Source code is re-usable (as it is independent of hardware)

Interpreter

- An interpreter is a program that simulates a computer that understands a high-level language
- Instead of translating the source code into machine code, the interpreter analyzes and executes the source code instruction-by-instruction.
- Source code is relatively slow as each instruction has to go through the interpreter for execution
- Source code is very portable (the hardware depence is on the interpreter. There should be an interpreter for each hardware or OS)
- Highly re-usable

Compiling vs Interpretation

- Compiling is a one-shot translation into machine language.
- A source code is compiled once and an executable software is generated. This executable can be run over and over again
- Source code and compiler is not required anymore for execution.
- Interpreter and source code are required every time for program execution.
- Compiled programs are faster than interpreted programs
- Interpreted software is more flexible for development
- Interpreted software is also more portable for different hardware and OS platforms.

Other Languages

- There exits some languages (java, c#) that are compiled into machine language for a virtual CPU
- For each CPU and OS, a virtual machine software executes the developed program
- This is a hybrid mode with both compilation and interpretation
- Gets best of each → portability + speed
- More portable than compiled programs
- Faster than interpreted programs

Python

- In ISE 101, we are going to learn and use Python programming language
- Why Python instead of other programming languages?
  - Easy syntax
  - Easy debugging

    bug: errors in a program

    debug: process of finding and fixing bugs
  - Focus

    less on the programming language

    more on designing algorithms for solving problems

Python

- In ISE 101, we are going to learn and use Python programming language
  - Interpreted high-level programming language
  - Many libraries are available (scientific computation, visualization, games etc.)
  - Used in software industry for professional code development
- Python can be used
  - Interactive mode
  - scripts

Python versions

- Python versions 2.x and 3.x have minor changes in syntax
- For example:
  - Python 2.x

    print “Hello”

    or

    print(“Hello”)
  - Python 3.x

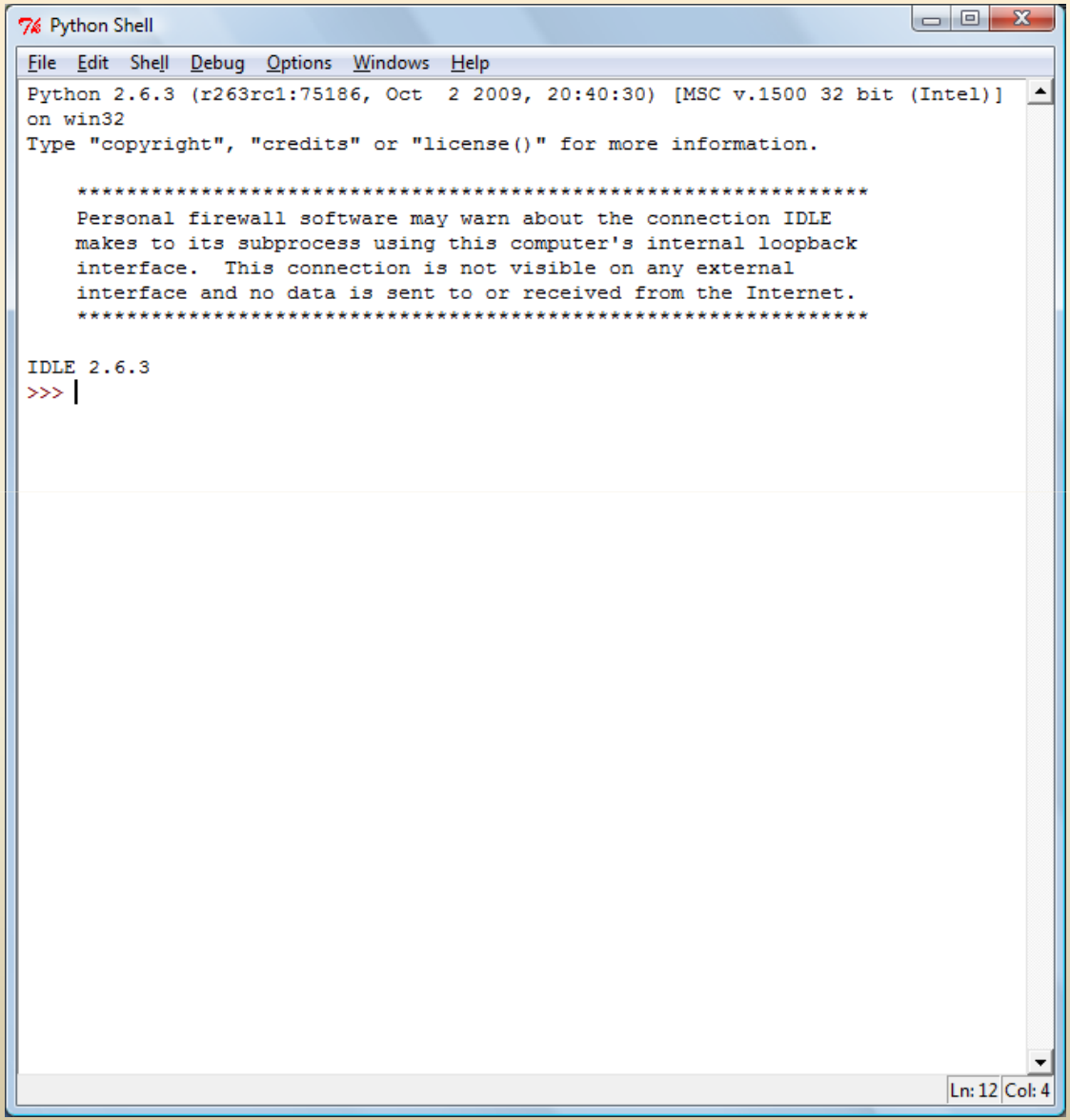
    print(“Hello”)
- Please use Python version 3.x (Latest version)
- Download from http://www.python.org

Integrated Development Environment (IDE)

- Larger software projects require complicated development tools that have
  - integrated editor (with syntax highlighting)
  - integrated python shell
  - integrated debugging tool
- Such development environment software is called "**I**ntegrated **D**evelopment **E**nvironment" (**IDE**)
- Wing101 will be used in this course (you can use other IDEs such as Eclipse)
- Download Wing101

http://wingware.com/downloads/wingide-101

Interactive Mode



Interactive Mode

- Python interpreter (called IDLE) can be started in interactive mode
- In this mode, “>>>” prompt indicates that the interpreter is ready for a command
- User writes a command in the proper syntax and after ENTER is pressed, the command is executed and the result of this single command is displayed on screen.
- This mode is good for trying out new things in Python
- For example:

```
>>> print 1+2
3
```

Script Mode

- All definitions (like functions) and variables are lost when we quit the interpreter
- This mode is not suitable for code development
- Code development is done by listing commands in file that is called a module or script
- Simple scripts can be written in any text processing program.
- However, medium or large sized software projects cannot be developed in “notepad”
- Integrated development environment (IDE) are used for these projects (Eclipse etc.)

Script Mode

- Once the scripts are written, they should be given a filename with ".py" extension
- These script files can be executed by double-clicking on the file

  or
- "python filename.py" from commandline or terminal

  or
- From the Python interpreter

  >>> import filename

Script Mode

- Sample script

```
# File: test.py
# A simple program for demonstration purpose

Def main():
    print “Demo program“
    x=input(“Enter a number between 0 and 1: “)
    for i in range(10):
        x=x*(1-x)
        print x

main()
```

Intermediate Python Files

- When a module is imported or executed first time, a file with ".pyc" extension is created
- This is an intermediate file for Python
- Python (as java and c#) uses a hybrid compiling / interpreting process
- Python source is compiled into more primitive instructions called *byte code*
- This makes the execution faster
- If you delete the byte code, Python will regenerate it again

First Python Script

- **Write a python script that computes the perimeter of a circle whose radius is 3.2 cm.**
- If you cannot solve a problem manually, you cannot design an algorithm and implement it.
- Perimeter of a circle = 2 * pi * radius

```
radius = 3.2
pi = 3.14

perimeter = 2 * pi * radius

print('Perimeter of the circle is ' \
    + str(perimeter))
```

Elements of Programs

- Names: Names are assigned to variables, functions etc.
- These names are called identifiers
- Python (and many of the other programming languages) has rules about how identifiers are formed
- Every identifier must begin with a letter or underscore "_" which may be followed by any sequence of letter, digits or underscores

Counter1	3cities
_CityName	big+city
name_surname	^new_variable
password4you	good@school

Elements of Programs

- An identifier cannot contain any spaces

  new constant

- Identifiers are case-sensitive

  art Art aRt arT ARt aRT ART (all different variables)

- Identifiers can be chosen freely. However it is really important to choose intelligent identifiers.

Good Choices for Identifiers

- Important aspects of coding
  - Readability:

    The code should be easily understood by others and by you

    After many years, you have to read your own code !

  - Reusability

Code Readability

- What is the difference between

```
radius = 3.2
pi = 3.14

perimeter = 2 * pi * radius

print('Perimeter of the circle is ' \
    + str(perimeter))
```

```
r= 3.2
r23 = 3.14

p = 2 * r * r23

print( str(perimeter))
```

Code Readability

- Even better → use comments as much as possible

```
# This code computes the perimeter of circle
# whose radius is 3.2 cm

# radius of the circle
radius = 3.2

# math constant pi
pi = 3.14

# perimeter is computed as 2 times pi times radius
perimeter = 2 * pi * radius

# print out the results
print('Perimeter of the circle is ' \
      + str(perimeter))
```

Naming Conventions for Identifiers

- There are many conventions available for naming conventions.
- Conventions are available on the Internet
- Choose a convention and stick to it throughout the code
- In this course we will use either
  - Camelcase
  - Snakecase
- You can use anyone.
- Do not use both convention within the same code

Naming Conventions

- **Camelcase** convention

  All white spaces and punctuations are removed

  First letter of each world is capitalized

  big city traffic → BigCityTraffic

  my brother's car → MyBrothersCar

  midterm #1 grade → Midterm1Grade

Naming Conventions

- **Snakecase** convention

  All punctuations are removed

  White spaces are replaced with underscore sign

  big city traffic → big_city_traffic

  my brother's car → my_brothers_car

  midterm #1 grade → midterm_1_grade

Identifiers

- Some of the names are reserved for Python statements
- These reserved words cannot be used for identifiers
- These reserved words are

and	**del**	**for**	**is**	**raise**
assert	**elif**	**from**	**lambda**	**return**
break	**else**	**global**	**not**	**try**
class	**except**	**if**	**or**	**while**
continue	**exec**	**import**	**pass**	**yield**
def	**finally**	**in**	**print**	

Expressions

- Fragments of code that produce or calculate new data values are called expressions
- Simplest kind of expressions is literal
- A literal is used to indicate a specific value

  >>> pi=3.14

  3.14 is a numeric literal
- More complex expressions can be constructed by combining expressions with operators
- Spaces within expressions are ignored. However, spaces should be used for easy reading.

Python Mathematical Operators

- Python operators are
  - Addition (+)
  - Subtraction (-)
  - multiplication (*)
  - division (/)
  - exponentiation (**)

Operator Precedence

- Some operators have precedence over others
- Order of precedence from high to low
  - Paranthesis
  - Exponential
  - Multiplication and division
  - Addition and subtraction

Example

- **Write a python code that converts fahrenheit to celcius**
- Fahrenheit = 9*Celcius/5 + 32

```
celcius=100

fahrenheit=9*celcius/5+32

print("celcius: " + str(celcius) +
    " = fahrenheit: " + str(fahrenheit))
```

Output Statements

- “print” command is used in Python to display information on screen
- print command can be used as

```
print(“something to display on screen”)
```

- Examples

```
print(“message part 1” + “message part 2”)
print(“2 * 5 =“ + str(10))
```

- + is used to combine strings

  “message ” + “combined” → “message combined”
- str(…) is used to convert another type (such as integer, float etc. → will be shown later) into a string

Assigning Input

- Purpose of an input statement is to get some information from the user and store it into a variable
- This is accomplished using an assignment statement combined with a special expression called “input”.

  <variable>=input(<prompt>)
- Prompt is an expression that serves to prompt the user for input. This is almost always a string literal

  >>> x = input(“Enter a value between 0 and 1”)
- Then it pauses and waits for user to type an expression and press <Enter> key.

Assinging Input

- After the user hits <Enter> key, the input is assigned to the variable
- For example

```
>>> temp = input("Enter the temperature: ")
Enter the temperature: 30
>>> print(temp)
23
```

- The input is assigned as a string
- To convert string into an integer use int(…)
- To convert string into a floating number use float(…)
- Integers: -2, 45, 237
- Floating point numbers: 3.45, 0.004, -4.53

Assignment Statements

- Basic assignment statement in Python is

  <variable> = <expr>
- This is an assignment NOT equality

  i = i + 1 (this equality is never correct)
- Left side expression is evaluated and its result is assigned to the variable on the right side.

  fahrenheit = 9.0 / 5.0 * celcius + 32
- A variable can be assigned a value many times. It retains the value of the latest assignment

Simultaneous Assignment

- Alternative to a single assignment , we can calculate and assign many values at the same time

  <var>, <var>, ..., <var> = <expr>, <expr>, ..., <expr>
- This is called simultanous assignment
- All of the expressions on the right hand side are evaluated and their results are assigned to the corresponding variable at the left-hand side

  >>> sum, diff = x+y, x-y
- This is equivalent to

  >>> sum = x+y

  >>> diff = x-y

Example

- **Write a Python script that computes the sum of squared numbers between 1 to 3.**

```
total=0

total=total+1**2
total=total+2**2
total=total+3**2

print("Total is "+str(total))
```

Example

- **Write a Python script that**

  1) gets the temperature from the user (in Celcius)

  2) converts it to Fahrenheit

  3) prints it on screen.

```
tmp=input("Enter temperature in Celcius: ")

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

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

Example

- **Write a Python script that**

  1) gets the 3 numbers from the user

  2) computes their average

  3) prints it on screen.

```
numbers=input('Enter 3 numbers: ')

num1,num2,num3=eval(numbers)

avg=1/3*(num1 + num2 + num3)
print("Average of " + str(num1) + " " +
    str(num2) + " " +
    str(num3) + " is " + str(avg))
```

Example

```
numbers=input('Enter 3 numbers: ')

num1,num2,num3=eval(numbers)

avg=1/3*(num1 + num2 + num3)
print("Average of " + str(num1) + " " +
    str(num2) + " " +
    str(num3) + " is " + str(avg))
```

- Values should be seperated using commas

```
Enter 3 numbers: 1,2,9
Average of 1 2 9 is 4.0
```

Example

- **Write a Python script that**

  1) gets the radius from the user

  2) computes the area of the circle

  3) prints it on screen.

```
radius=float(input("Enter the radius: "))

area=radius**2

print("radius: "+ str(radius)
                + "--> area: " + str(area))
```

Example

- **Write a program that converts US Dollars to a Turkish Lira:**
- **get the amount of USD to be converted from the user**
- **get the exchange rate (US/TL) from the user**
- **prints the TL amount**

```
radius=float(input("Enter the radius: "))

area=radius**2

print("radius: "+ str(radius)
                + "--> area: " + str(area))
```

Example

```
USD=float( input('Enter the amount of USD: ') )
exchange_rate=float( input('Enter exchange rate USD/TL: '))

TL = USD * exchange_rate

print(str(USD) + ' USD = ' + str(TL) + 'TL')
```

- Output

```
Enter the amount of USD: 10
Enter exchange rate USD/TL: 1.82
10.0 USD = 18.2TL
```