

# **TCSS 142 – Introduction to Programming**

**Autumn 2014  
Day 01**

# Day 1 Overview

- Introductions
  - Students
  - Instructor
  - Course
  - Programming
- Computer Science
  - Computer
  - Algorithm
  - Control structures
  - Python

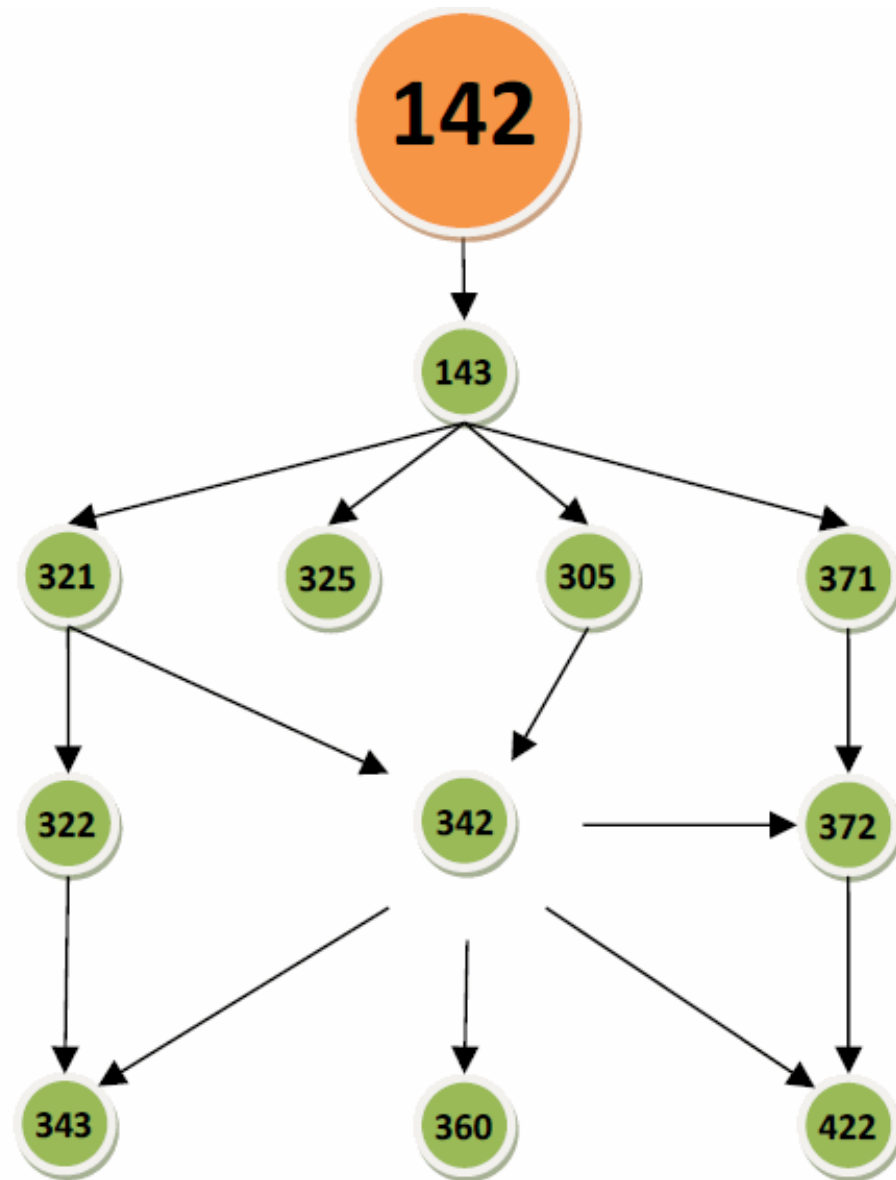
# Ice-breaker

- You have 10 minutes to complete the following
- Form groups of 3-4 with your classmates sitting next to you.
- Designate one person in your group to be the leader for the discussion and one person to be the note taker - you will turn in your notes at the end of the exercise
- Collect the following information about each group member:
  - Name, net id, major, level in college, how you want to be addressed
  - Any programming background – if so, what languages
  - Why are you taking this course?
  - An interesting and unique fact about each person
- Turn in the notes

# Syllabus Questions

- You have 20 minutes to complete the following.
- Log onto computers using your UW netid and password, log onto Canvas (<https://canvas.uw.edu>), find this course, and answer the following questions (help one another, if needed):
  - Who is your instructor and what is the best way to reach her
  - What is the grading scheme in this course
  - What are the rules regarding the lab
  - What are the rules regarding all other assignments
  - What are the other course documents located on Canvas
  - Do you any questions regarding the course that the syllabus does not cover

# Course Flowchart



# Degree Programs

- Computer and Software Systems (CSS)
  - The focus is on solving problems through the development of software
- Computer Engineering and Systems (CES)
  - The focus is on solving problems through the development of hardware
- Information Technology and Systems (ITS)
  - The focus is on solving problems through the use of existing and emerging technologies
- Learn more at: <http://www.tacoma.uw.edu/institute-technology>

# What is computer science?

- “The body of knowledge of computing is frequently described as the systematic study of algorithmic processes that describe and transform information: their theory, analysis, design, efficiency, implementation, and application. The fundamental question underlying all of computing is, What can be (efficiently) automated?” -- Peter Denning

(“Computer Science the Discipline,” Encyclopedia of Computer Science, ed. E. Reilly et al. Groves Dictionaries, Inc. 2000)

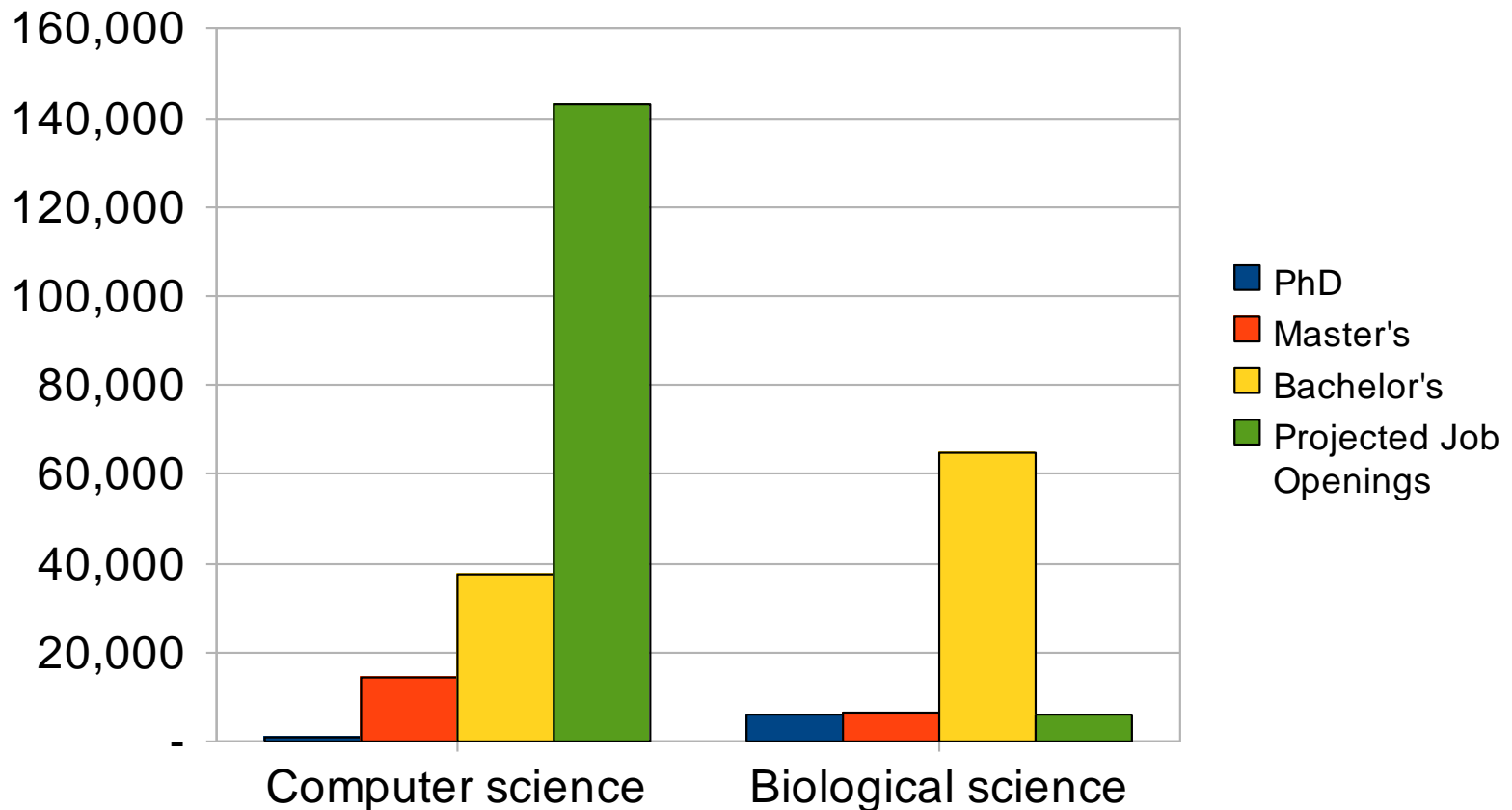
- Computation is the creative use of a machine to solve problems applicable to any domain in an efficient and automated manner.

# What is computer science?

- Computer science subfields ??



# The CS job market



SOURCES: Tabulated by National Science Foundation/Division of Science Resources Statistics; data from Department of Education/National Center for Education Statistics: Integrated Postsecondary Education Data System Completions Survey; and NSF/SRS: Sur

# What is Programming Like?

# Computer - Definition

- “Computers are incredibly fast, accurate, and stupid. Human beings are incredibly slow, inaccurate, and brilliant. Together they are powerful beyond imagination.”
- Computer – general-purpose(?) programmable device capable of:
  - receiving, processing, and storing input,
  - producing, displaying, and storing output,  
*all according to a series of stored instructions*
- Computer program - an algorithm written in a programming language.

# Algorithm

- It is a finite set of unambiguous instructions (written as a sequence) that solves a stated problem in a finite amount of time, terminating under its own control.
- Algorithms encompass much more than computer programs.
- Computers – fast and flexible tools for implementing algorithms
  - not all algorithms can be implemented as computer programs

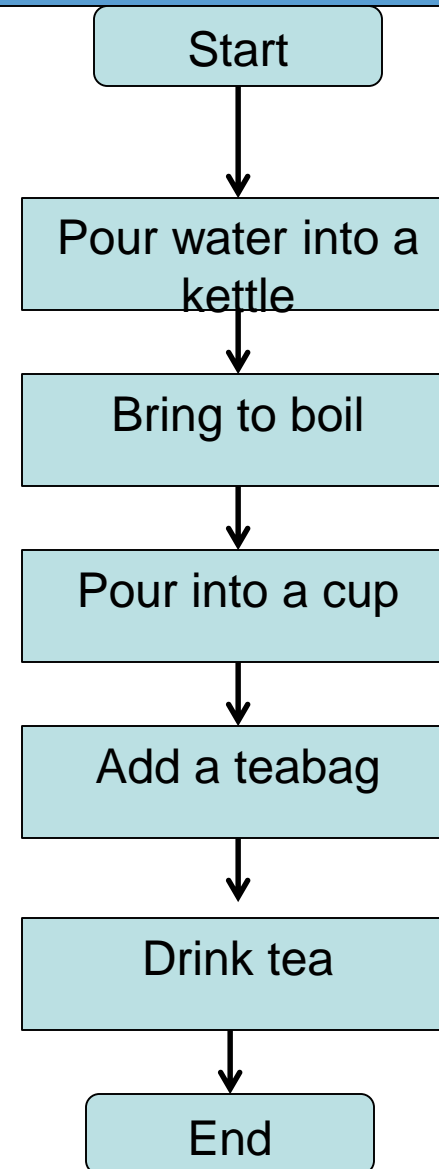
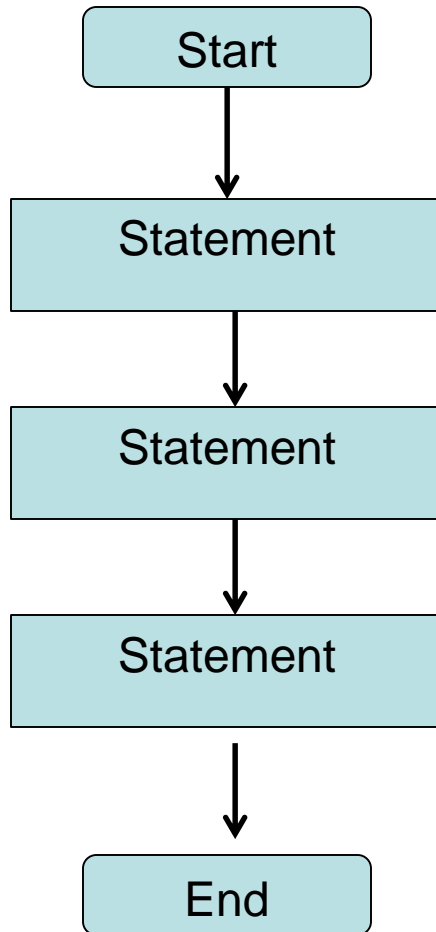
# Algorithm

- More than one solution to a given problem.
- Algorithms follow these general guidelines:
  - input
  - calculations (manipulate known information) – arithmetic and logical operations, control structures
  - output

# Basic Control Structures

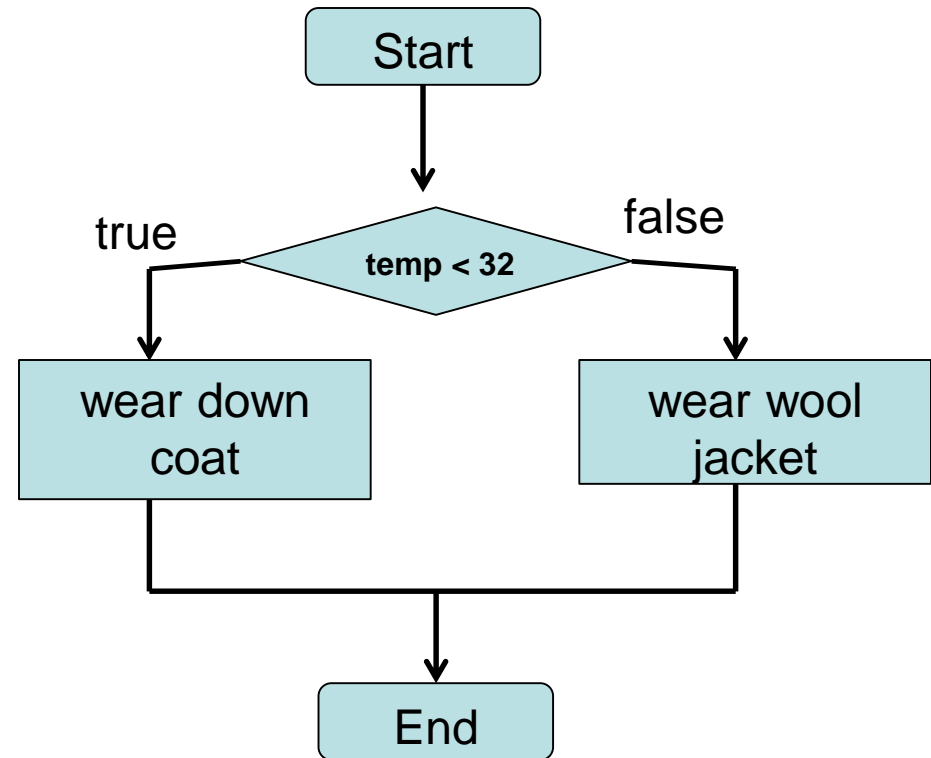
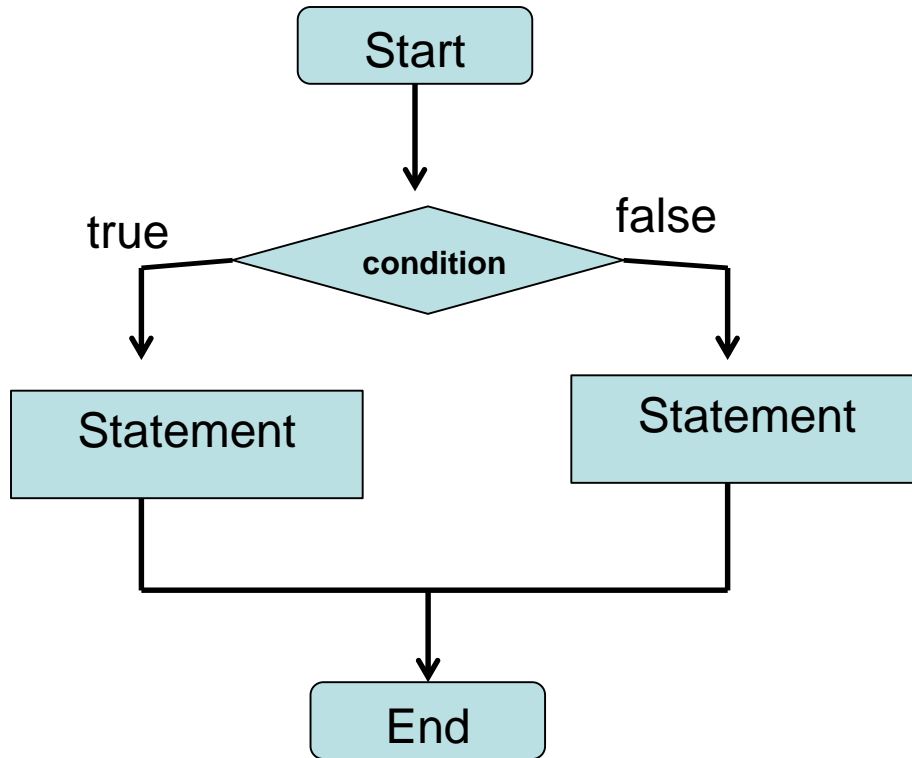
- a sequence is a series of statements that execute one after another
- selection (branch) is used to execute different statements depending on certain conditions
- looping (repetition) is used to repeat statements while certain conditions are met
- a subprogram is used to break the program into smaller units

# Sequence



# Selection (Branch)

IF Condition THEN Statement1 ELSE Statement2

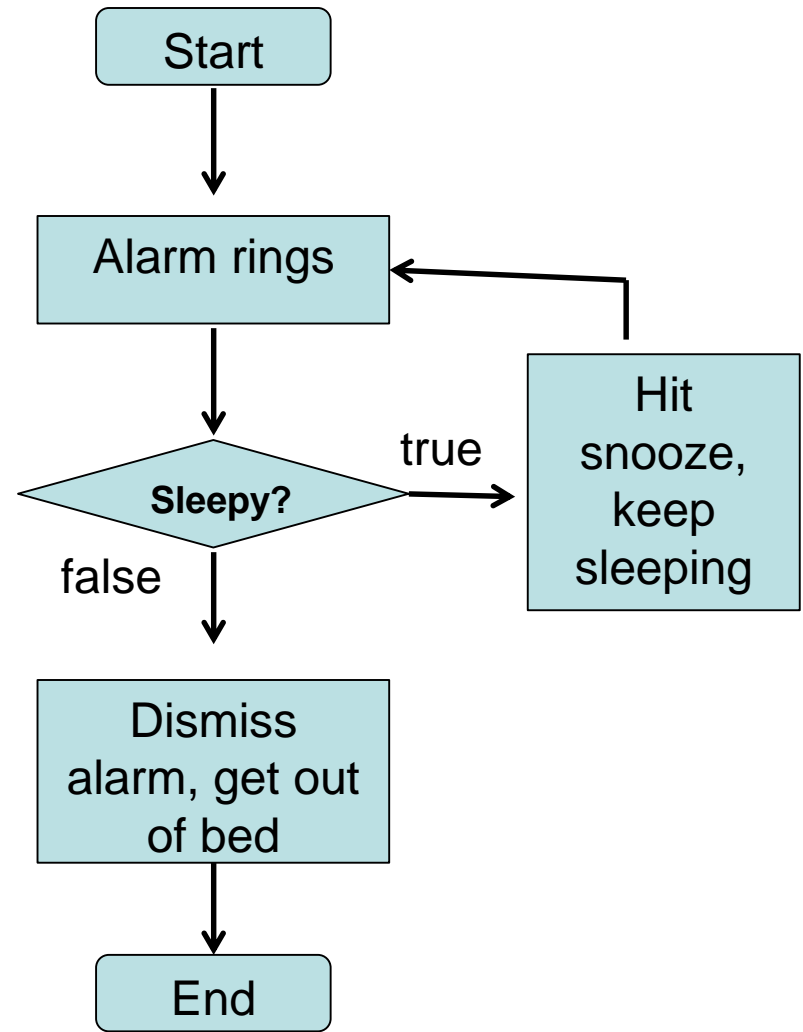
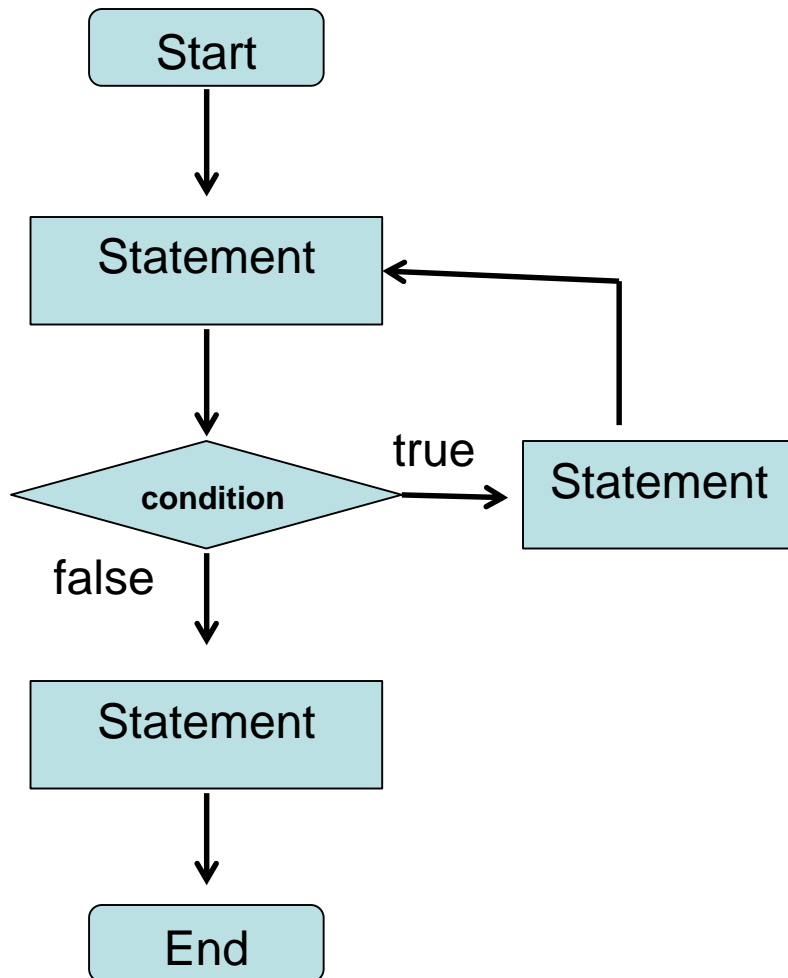


[http://www.rff.com/order\\_processing.htm](http://www.rff.com/order_processing.htm)

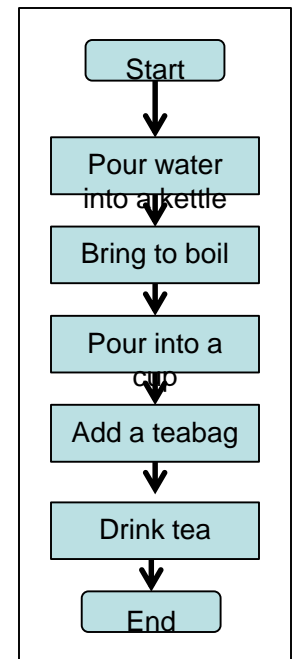
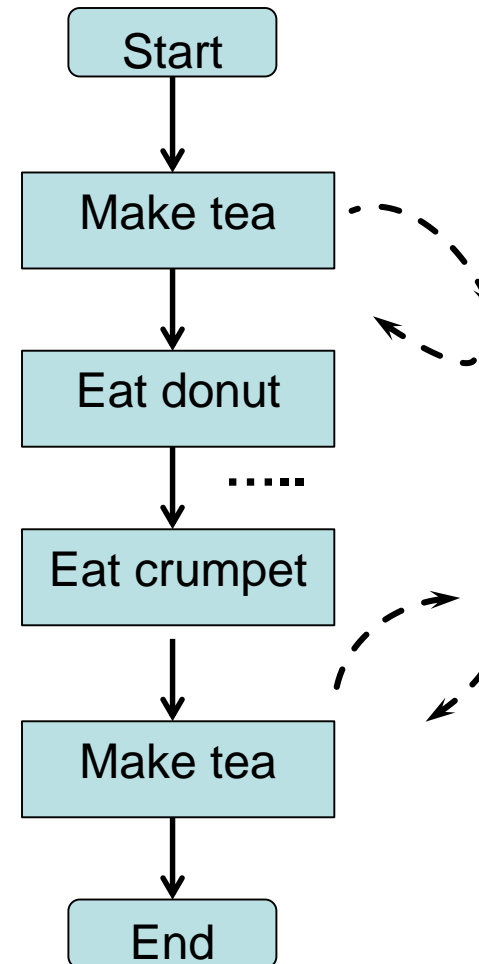
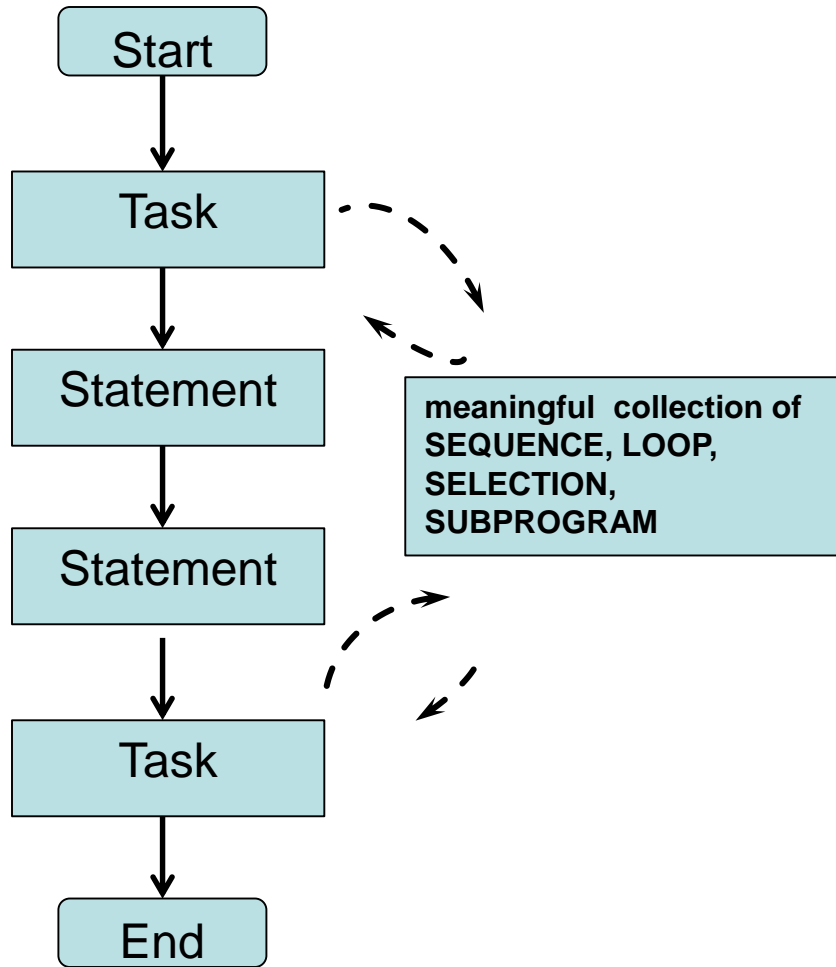


# Loop (Repetition)

WHILE Condition THEN Statement/s

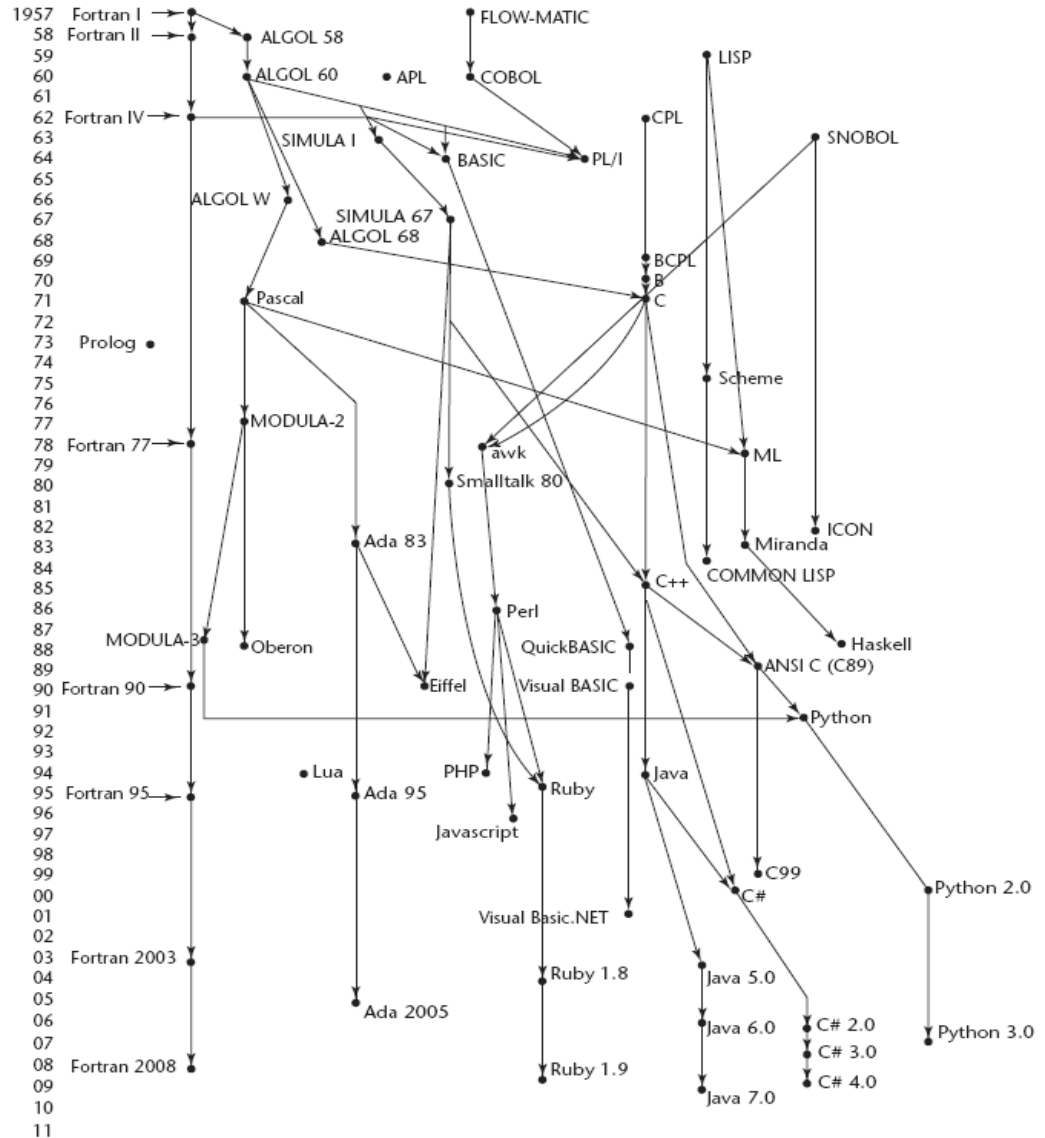


# Subprogram (Method)



# What is a

- It is a language with strict grammar rules, symbols, and special words used to construct a computer program
  - Low-level (machine, assembly)
  - High-level

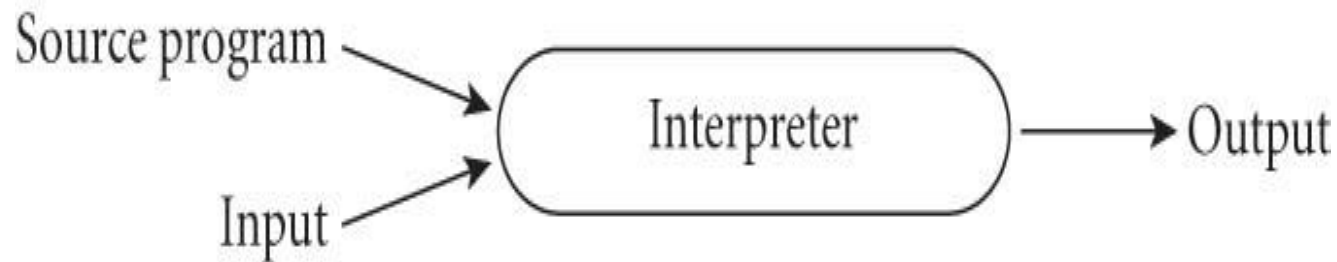


# High Level Languages

- are portable
- user writes program in language similar to natural language
- examples -- FORTRAN, COBOL, Pascal, Ada, Modula-2, C++, Java, **Python**
- most are standardized to provide an official description of the language
- must be translated into low-level languages
  - compilers
  - interpreters

# Interpreter

- Python is interpreted
  - the piece of software that translates Python code to machine code



# Some Python History

- 1991: Guido van Rossum invents Python
- Named after the 1970s BBC comedy series *Monty Python's Flying Circus*
- The language and many supporting tools are free; it runs on any operating system
- [www.python.org](http://www.python.org)
- Python is easy, powerful, and fun

# Using Python

- Python must be installed prior to its use
  - One of the items installed is the Python interpreter
- Python interpreter can be used in two modes:
  - Interactive mode: enter statements on keyboard
  - Script mode: save statements in a Python script
- Start the interactive mode
  - in the lab: GenDev folder -> Python -> IDLE (Python GUI)
  - anywhere on Windows: Start -> type python into search box -> IDLE (Python GUI)

# Interactive Mode

- When you start Python in interactive mode, you will see a prompt
  - Indicates the interpreter is waiting for a Python statement to be typed
  - Prompt reappears after previous statement is executed
  - Error message displayed, if you incorrectly type a statement
  - Good way to learn new parts of Python
- Use as a calculator
  - find the sum of the numbers 10, 2, 4 and then divide by a 0
  - find the product of the same numbers



# Objects

- Everything in Python is an object
- Objects could be of different types
- Number types:
  - Integers
  - Floating points
  - Complex numbers (A complex number is a number consisting of a *real* and *imaginary* part. It can be written in the form  $a + bi$ , where  $a$  and  $b$  are real numbers, and  $i$  is the standard imaginary unit with the property  $i^2 = -1$ )
- An expression is a combination of operators and operands.  
 $+, -, /, //, *, **, \%$

# Values

- Integers:

- Digits and a sign only
- No other symbols, i.e. no commas, units, etc.

~~1234.~~


~~\$1200~~


~~1,200,999~~

- Floating point numbers

- Digits, sign, period
- Scientific notation
- No other symbols

# Scientific Notation

**2.7e+04** means  $2.7 \times 10^4 =$   
 $2.7000 =$   
  
**27000.0**

**2.7e-04** means  $2.7 \times 10^{-4} =$   
 $0002.7 =$   
  
**0.00027**

# Integer Division Operator

- division by 0 produces ??
- two division operators: integer and floating point
- if one or both operands has a floating point type, the result is a floating point; otherwise, the result is an integer type

- Examples

11 / 4                      has value ?

11 // 4                    has value ?

11.0 // 4.0                has value ?

11 / -4                    has value ?

11 // -4.0                has value ?

# Exercise

- Find the average age of five people around you using integer division and floating point division
- Compute  $2^{10}$
- Compute  $2.0^{10}$
- Compute the factorial of 5
- Compute the number of seconds in a year

# Limits of Computation

- Integer – the size of the integer in Python determined by the size of RAM
- Floating point numbers
  - Enter the following commands:

```
>>> import sys
>>> print(sys.float_info)
```
  - What is max exponent?
  - What is min exponent?
- Not all real numbers can be represented – representational error ( $\pi$ ,  $1/3$ ,  $0.1$ )

# Operator Precedence

- Precedence determines which operator is applied first in an expression having several operators
- All operations inside of ( ) are evaluated first
- \*\* is evaluated next
- \*, /, // and % are at the same level of precedence and are evaluated next (left to right)
- + and – have the same level of precedence and are evaluated last (left to right)

# Exercise: with a partner

- First evaluate on a piece of paper, then run in Python

$2 + 3 * 4 - 6$

$(2 + 3) * 4 - 6$

$4.0 / 2 * 9 / 2$

$4.0 / (2 * 9) / 2$

$7 \% 2$

$12 \% 100$

$813 \% 100$

$813 \% 100 + 2$



# What is a Variable?

- Python allows us to name objects so that we can refer to them later.
- A variable name is a location in memory which we can refer to by an identifier, and in which a data value that can be changed is stored.

# Identifiers

- an identifier must start with a letter or underscore, and be followed by zero or more letters (A-Z, a-z), digits (0-9), or underscores

- **VALID**

age\_of\_dog                      taxRateY2K  
PrintHeading    ageOfHorse

- **NOT VALID (Why?)**

age#                              2000TaxRate  
Age-Of-Cat

# Identifiers

- Identifiers are case sensitive, e.g. age is not the same as Age
- Reserved words (keywords) have special meaning in Python and cannot be used as programmer-defined identifiers
  - Table 1-2, p. 17
- Variable name should reflect its use
- Variables can reference different values while program is running

# Assignment Operator Syntax

Variable = Expression

First, Expression on the right is evaluated.

Then the resulting value is stored in the memory location of the variable on the left, which we can reference later on our program

# Example

```
>>> pi = 3.14
>>> pi
>>> radius = 8.0
>>> height = 16
>>> baseArea = pi * radius **2
# comment
>>> baseArea
>>> cylinderVolume = baseArea * height
>>> cylinderVolume
```

# Exercise

value of	a	b	c	d	idx
>>> a = 10					
>>> b = 20					
>>> a = b					
-----					
>>> a = 10					
>>> b = 20					
>>> c = a * b					
>>> d = a + b					
-----					
>>> idx = 0					
>>> idx = idx + 1					
>>> idx = idx + 2					

# Script Mode

- Statements entered in interactive mode are not saved as a program
- To have a program use script mode
  - Save a set of Python statements in a file
  - The filename should have the .py extension
  - To run the file, or script, type  
`python filename`  
at the operating system command line or open the file and select Run -> Run Module

# Demo

- Open Python shell

- File -> New File

- Enter

```
pi = 3.14
```

```
radius = 8.0
```

```
baseArea = pi * radius **2
```

```
print(baseArea)
```

- File -> Save As

- Navigate to your H: drive

- Name the file myFirstProg.py

- Run -> Run Module

- Program output is written to ???

- Help -> Python docs



# What is a string?

- Everything in Python is an object; objects could be of different types
- A string is simply a sequence of characters (any characters)
  - In Python an object is a string, if it is surrounded by single or double quotes

```
>>> "hello"
??
>>> 'hello'
??
>>> howdy = "Howdy TCCS 142"
>>> howdy
??
>>> howdy2 = "You rock!"
>>> howdy2
??
```

# Concatenation

- To append strings together, use +

```
>>> howdy + howdy2
```

```
??
```

```
>>> message = howdy + ' . ' + howdy2
```

```
>>> message
```

```
??
```

# Output

- Function: piece of prewritten code that performs an operation
- `print` function: displays output on the screen
- Argument: data given to a function
  - Example: data that is printed to screen

```
print(baseArea)
print(13)
print(2 * 3 + 4)
print("baseArea")
print('2 * 3 + 4')
```

# Exercise

- First evaluate on a piece of paper, then run in Python

```
print('hello', 'tcss', '142')
print('hello', 'tcss', '142', sep = ',')
print('hello', 'tcss', '142', sep = '\t')
print("hello 'tcss' 142")
print("hello "tcss" 142")
print("hello \"tcss\" 142")
print("hello tcss " + 142)
```

# Example

- What would be the output of the following program?

**# first verse**

```
print("Bawitdaba", end = ' ' )  
print("da bang a dang diggy diggy diggy")  
print()
```

**# second verse**

```
print("said the boogy")  
print("said up jump the boogy")
```

# Exercise

- Write and save a program `output.py` to your H:drive. The program prints the following message – line-by-line

The `""` represents an empty string.

Backslash character `\` causes an "escape" from the 'normal' way characters are interpreted by the compiler.

And `#` represents comments.

# Receipt Question

Download the receipt program from Canvas and improve it using variables.

**# Calculate total owed, assuming 8% tax / 15% tip**

```
print("Subtotal:", end=' ');  
print(38 + 40 + 30);
```

```
print("Tax:", end=' ');  
print((38 + 40 + 30) * .08);
```

```
print("Tip:", end=' ');  
print((38 + 40 + 30) * .15);
```

```
print("Total:" , end=' ');  
print(38 + 40 + 30 + (38 + 40 + 30) * .15 + (38 + 40 +  
30) * .08);
```

# Receipt Answer

```
subtotal = 38 + 40 + 30
tax = subtotal * .08
tip = subtotal * .15
total = subtotal + tax + tip

print("Subtotal:", end = ' ')
print(subtotal)

print("Tax: ", tax)

print("Tip: ", tip)

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



# Last Slide 😊

- There is reading / quiz to be completed before the next class meeting
  - Read chapter 2 from your textbook and take the associated quiz
- There is reading / quiz to be completed within the next 2 weeks
  - Read chapter 1 from your textbook and take the associated quiz
- Class ends at 17:10

# Number and field formatting

- Using built-in `format` function
  - Two arguments:
    - Numeric value to be formatted
    - Format specifier
  - Returns string containing formatted number
  - Format specifier typically includes precision and data type
- As a lab exercise
- Formatting does NOT change variable value – formats for display purposes only!!!

# Input

- Most programs need to read input from the user
- Built-in `input` function reads input from keyboard
  - Returns the data as a string
  - Format: `variable = input(prompt)`
- In order to interpret the value as an int or as a float, the string value has to be converted
  - `int(item)` converts *item* to an int
  - `float(item)` converts *item* to a float

# Example

- Open `myFirstProg.py` and save as `areaIO.py`
- Let's change the program so that we prompt for input, read it, and then calculate the area of a circle
- Let's extend the program by prompting for a height as well and calculating the volume of a cylinder
- What happens when a wrong value is entered?

- Storage of floats and chars
- Accessing h drive from home
- Algorithms: input – process - output