CIS 125 Principles of Programming Logic
Input – Process – Output

- Read Chapters 2, 3, and 4 of eBook.
- The basic purpose of most programs (and computers) is **IPO**: Input -> Process -> Output
- Program name examples: hw1-1.py, calc_pay.py, calc_tuition.py, output2.py, math3.py
  Can include spaces, but we won't. For homework, use: hw1-1.py, hw1-2.py, hw2-1.py, etc.
- Include section of **comments/remarks** in all your programs at the top (see examples below)
- **Variables**: store data; use short descriptive names, no spaces, no reserved words, not all caps, no special characters. Examples: firstName, salary, mph, tax_rate.
  - Data types: integer, float, string
- You output data and information to screen using **print** command
- **Concatenation**: joining/linking/combining two things together.

Introduction

Steps to creating a computer program:

Step	Description
1. Design the program	Create algorithm, determine, logic (steps). Use of flowcharts, hierarchy chart, and/or pseudocode. Blueprint. Like an outline to a paper. The need for increases as complexity of program/software increases. Often, you start with output first and work backwards.
2. Code the program	Chose language and begin programming. Use of interpreter, compiler, IDE, etc.
3. Test the program	Run the program and see if there are any syntax or semantic errors. If appropriate, try a variety of input options. The more complex the program/software, the more the testing, e.g. functionality testing, usability testing, load/stress testing, security testing, etc.
4. Debug program	Correct any errors and repeat test process.
5. Document and maintain	Create documentation/manual for program and provide ongoing maintenance, e.g. features, fixes, testing, updates, etc.

Source:
https://en.wikibooks.org/wiki/The_Computer_Revolution/Programming/Five_Steps_of_Programming

Pseudocode is more English like wording used to quickly write down and design a program without concerning oneself with language-specific syntax. One reason Python is such as great language is its syntax is very clean and simple, almost pseudocode like in some ways.

Variables and Output

Example: Simple **string** output

```
print("Hello, world!")
```

Output:

Hello, world!

Example: Example with **variables** and output: includes string, integer, and float **data types**

```
x = 45
y = 19.5
print("Hello, world!")
print("x =", x, "and y =", y)
```

Output:

Hello, world!
x = 45 and y = 19.5

Example: Program with **comments/remarks** at top and multiplying a string to create a line

```
# Python Multiplying a string to create a new line example program
# May 26, 2016
# Author: D Maier
print('-' * 40)
```

Output:

Example: Example with variables and output: Strings with **Concatenation**

```
message1 = "Hello"
message2 = "Goodbye"
print(message1,message2)
print(message1 + " " + message2)

print message1," ", message2)
print(message1 + message2)
```

Output:

Hello Goodbye
Hello Goodbye
Hello Goodbye
HelloGoodbye

Example: Example with Output: Formatting (\t is tab escape sequence)

```
################################################
# Python Example Program
# Examples of output and formatting
# Created: January 1, 2017
# Author: David Maier
################################################
EmpName1 = "John Smith"
salary = 45100.5
print("Salary: $", salary)
print("Salary: %.0f" % salary)
print("Salary: $%.2f" % salary)
print("Name:\t\tJohn Smith\nSalary:\t\t$%.2f" % salary)
print("William ", end="")
print("Susan ", end="")
print("Sally")
print("Employee 1 name: ", EmpName1[5:] + ", " + EmpName1[0:4])
```

Output:

Salary: $ 45100.5
Salary: 45100
Salary: $45100.50
Name: John Smith
Salary: $45100.50
William Susan Sally
Employee 1 name: Smith, John

Example: Tab Escape Sequence

```
print("First Quarter\t\t Second Quarter\t\t Third Quarter\t\t Fourth Quarter\t\t")
print("-------------\t\t --------------\t\t -------------\t\t --------------\t\t")
print("   $1,120,010 \t\t     $1,554,200\t\t    $1,234,321\t\t     $1,233,003\t\t")
```

Output:

First Quarter	Second Quarter	Third Quarter	Fourth Quarter
$1,120,010	$1,554,200	$1,234,321	$1,233,003

Example: Outputting multiple arguments

```
x, y, z = 15.5555, 20, 44.3
print("The variable values are %.2f %.2f %.2f" % (x, y, z) + ".")
```

Output:

The variable values are 15.56 20.00 44.30.

Example: New Line Sequence

```
print(" January \n February \n March")
```

Output:

January
February
March

Example: Formatting numeric output in Python (you do not place commas in large numbers when assigning the value; that is only done during output)

```
w = .25
x = 10
y = 15.3
z = 1000000

print("y =", y)
print("y = %.2f" % y)
print("y = %.2d" % y)
print("z =", z)
print("{:,}".format(z))
print("{:.2%}".format(w))
```

Output:

y = 15.3
y = 15.30
y = 15
z = 1000000
1,000,000
25.00%

Example: Suppressing a new line

```
print("William")
print("Susan")
print("Sally")
print("William ", end="")
print("Susan ", end="")
print("Sally ", end="")
```

Output:

William
Susan
Sally
William Susan Sally

Processing (via Math)

Example: Example of Processing: More use of Mod Math Operation

```
print(5 + 2)
```

Output:

7

Example: Storing math calculations in variables and then outputting then

```
x = 10 % 2
y = 6 % 4
z = 5 % 4
print(x)
print(y)
print(z)
print("z mod y = ", z % y)
```

Output:

0
2
1
z mod y = 1

Example: Order of Arithmetic

```
x = 3
y = 5
z = 6
result1 = y - x * z
result2 = (y - x) * z
print("Result1 = ", result1)
print("Result2 = ", result2)
```

Output:

Result1 = -13
Result2 = 12

Example: Math operation that divides and provides the integer/floor result

```
x = 9
y = 9 // 4
print(y)
```

Output:

2

Example: More math

```
a = 3
b = 5
c = 4.5
d = 2

print("a=", a, "b=", b, "c=", c, "d=", d)
print("---------------------------")
print("a + b =", a + b)
print("a + b * d =", a + b * d)
print("(a + b) * d =", (a + b) * d)
print("b mod a =", b % a)
print("Average of a, b, and c=", (a+b+c) / 3)
avg = (a+b+c) / 3
print("Average of a, b, and c= %.2f" % avg)
print("b to the power of d =", b ** d)
```

Output:

a= 3 b= 5 c= 4.5 d= 2
---
a + b = 8
a + b * d = 13
(a + b) * d = 16
b mod a = 2
Average of a, b, and c= 4.166666666666667
Average of a, b, and c= 4.17
b to the power of d = 25

Example: Math Library and functions/methods within in it

```
import math

total1 = math.pow(5,2)
total2 = math.sqrt(36)
total3 = math.pi * 2.5

print ("Total1 = ", total1)
print ("Total2 = ", total2)
print ("Total3 = ", total3)
```

Output (two runs of the program):

Total1 = 25.0
Total2 = 6.0
Total3 = 7.853981633974483

Example: Various Math Formulas (note use of // and int)

```
import math
x = 100
a = x / 7
b = int(x / 7)
c = math.ceil((x / 7))
d = -(-x // 7)

print(a)
print(b)
print(c)
print(d)
```

Output:

```
14.285714285714286
14
15
15
```

User Input

There are five main **data types** in Python: Numbers, **String**, List, Tuple, and Dictionary. This section will cover the first two. The last three will be covered later in the semester. With respect to numbers, we will cover the two most common types: **integer and float**. Python also supports long integers and complex numbers.

Example: Three variables with three different data types – integer, float, and string (no output)

```
x = 100
y = 10.5
fname = "Mary"
```

Example: User Input: String Data Type

```
first_name = input("What is your first name? ")
print("Hello", first_name)
```

Sample run:

```
What is your first name? John
Hello John
```

Example: User Input: Integer Data Type, Math, and Output

```
num = int(input("Enter a number: "))
result = num * 2
print("Result=", result)
```

Sample run:

Enter a number: 3
Result= 6

Sample run:

Enter a number: 3.3
Traceback (most recent call last):
  File "C:/input.py", line 1, in <module>
    num = int(input("Enter a number: "))
ValueError: invalid literal for int() with base 10: '3.3'

** In the second run of the program, a syntax error is produced and the program terminated. You cannot enter a float value when the user input is expecting an integer. User input must be an integer. Later in the semester we will learn how to validate the user input and prevent the program from crashing.

Example: User Input: Float Data Type, Math, and Output

```
num = float(input("Enter a number: "))
result = num * 2
print("Result=", result)
```

Sample run:

Enter a number: 2.5
Result= 5.0

** Note: you can enter an integer value as user input into a float. It converts the user input to float.

Example: User Input: Integer Data Type, Math, and Output (watch integer get converted to float)

```
num = float(input("Enter a number: "))
print(num)
```

Sample run:

Enter a number: 2
2.0

Example: Type casting (converting an integer into a float)

```
num = int(input("Enter a number: "))
num2 = float(num)
print(num2)
```

Sample run:

Enter a number: 2
2.0

Example: Input and Display Age

```
age = int(input("Enter your age: "))
print("You are", age, "years old.")
```

Example: Input and Display Age

```
num = input("Enter input: ")
result = num + 2
print(result)
```

Sample run:

Enter input: 2
Traceback (most recent call last):
  File "C:/input.py", line 2, in <module>
    result = num + 2
TypeError: Can't convert 'int' object to str implicitly

** Note a syntax error was produced because user input was taken in as a string and you cannot perform math on a string.

Example: User Input: String and Int Data Types

```
print("Welcome to this input example program.")
name = input("What is your name? ")
age = int(input("What is your age? "))
print("Welcome", name, ". You are", age, "years old.")
print("Welcome", name + ". You are", age, "years old.")
```

Sample run:

Welcome to this input example program.
What is your name? John Smith
What is your age? 23
Welcome John Smith . You are 23 years old.
Welcome John Smith. You are 23 years old.

Example: User Input (Float), Processing (Math), and Output

```
print ("This program will multiple two float numbers.")
x = float(input("Enter the first integer number:"))
y = float(input("Enter the second integer number:"))
total = x * y
print (x, " x ", y, " = ", total)
print (x, " x ", y, " = %.2f" % total)
```

Sample run:

This program will multiple two float numbers.
Enter the first integer number:2.5
Enter the second integer number:1.25
2.5 x 1.25 = 3.125
2.5 x 1.25 = 3.12

Example: Calculating a Baseball Batting Average

```
hits = int(input("Enter the player's number of hits: "))
bats = int(input("Enter the player's number of times at bat: "))
battingAverage = hits / bats
print("The player's batting average is ", battingAverage)
print("The player's batting average is %.3f" % battingAverage)
```

Sample run:

Enter the player's number of hits: 2
Enter the player's number of times at bat: 8
The player's batting average is 0.25
The player's batting average is 0.250

Example: User Input of Three Data Types (Input -> Process -> Output)

```
# Python Example Program
# January 21, 2018
# Author: D Maier
TAX_RATE = .25

# User input
firstName = input("Please enter your first name: ")
salary = float(input("Please enter your gross salary: "))
hireYear = int(input("Please enter the year you were hired: "))

# Calculations
netSalary = salary * (1 - TAX_RATE)

# Output
print("Hello,", firstName + ". You were hired in", hireYear)
print("Your net salary is", netSalary)
print("Your net salary is ${:,.2f}".format(netSalary))
```

Output:

```
Please enter your first name: William
Please enter your gross salary: 40000
Please enter the year you were hired: 2000
Hello, William. You were hired in 2000
Your net salary is 30000.0
Your net salary is $30,000.00
```

Example: Combining a print and input command onto one line

```
print(input('What is your name? '))
```

Sample run:

```
What is your name? William
William
```

Example: Use of a constant – STANDARD_WEEK

It is a standard in programming to use all capital letters when naming a **constant**, i.e. a value that should never change in the program. Constants are also placed together near the top of the program with other variable declarations.

```
STANDARD_WEEK = 40
rate = float(input("Please enter your hourly rate: "))
pay = STANDARD_WEEK * rate
print("Your pay is $%.2f" % pay)
```

Sample run:

```
Please enter your hourly rate: 10
Your pay is $400.00
```

Assess Your Understanding

1. What type of user input is this statement expecting?

```
val = input("Please enter user input: ")
```

- ☐ Integer
- ☐ Float
- ☐ String
- ☐ Tuple

2. What is wrong with this program?

```
rate = 15
total pay = 15 * 40
print("Your total pay is", total pay)
```

_____

3. The error in the program above is which of the following kinds?

- ☐ Syntax
- ☐ Semantic

4. What is the output of this program?

```
x = 2
y = ((2 + 3) * x) - 3
print(y)
```

_____

5. What is the output of this program?

```
import math
total = math.pow(2,3)
total = total % 3
print(total)
```

_____

6. How many lines of output will this program product?

```
print("Supervisors ", end="")
print("Managers ", end="")
print("Directors ", end="")
```

☐ 0

☐ 1

☐ 2

☐ 3

7. What is the output of this program?

```
num1 = 9
num2 = 4
total = num1 % num2
print("%.4f" % total)
```

_____

8. In this line of code, TAX_RATE is which of the following?

```
TAX_RATE = .25
```

☐ A variable

☐ A string

☐ A value

☐ A constant