CSCI 141
Computer Programming I

Filip Jagodzinski
PRIOR to attending lab next week, you must activate your CS account

- The CS department has its own network and computer science labs
- Your username on the CS network is the same as your university computer username
- Activation and password setup for your CS account is at http://password.cs.wwu.edu
- You CANNOT activate your account and set the password from a CS lab computer. Do it BEFORE you attend the first lab.

Once labs and homework assignments become available (next week) ...

<table>
<thead>
<tr>
<th>Filip</th>
<th>Robert</th>
<th>Albert</th>
<th>ACM tutors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mon, Tue, Wed</td>
<td>Mon, Wed, Thur, Fri</td>
<td>Tuesday</td>
<td>Mon-Fri</td>
</tr>
<tr>
<td>11am-noon</td>
<td>4-5pm</td>
<td>2-4pm</td>
<td>4-7pm</td>
</tr>
<tr>
<td>CF 461</td>
<td>CF 405</td>
<td>CF 405</td>
<td>CF 162/164</td>
</tr>
</tbody>
</table>
Have you coded before?

If you can answer the following three questions more-or-less correctly without having to refer to a textbook, then please come see me after lecture and/or during my office hours.
Have you coded before?

1. True or False: The binary equivalent of the base 10 number 100 is 0100110011
Have you coded before?

1. True or False: The binary equivalent of the base 10 number 100 is 01001110011

2. What is the output of the python code in the below box?

```python
onMyMind = [4, "A", [4, "B"], [ ], False]
print(onMyMind[2:-3])
```
Have you coded before?

1. True or False: The binary equivalent of the base 10 number 100 is 01001110011

2. What is the output of the python code in the below box?

   ```python
   onMyMind = [4, "A", [4, "B"], [ ], False]
   print(onMyMind[2:-3])
   ```

3. Write a function `excise` with two parameters; both arguments are strings. The function should return a string that is the second argument with all occurrences of the first argument removed. For example

   ```python
   print(excise ("W", "WWU CS"))
   ```

   would output

   `U CS`
Today

- Programming ... large and small ... is an art and a science
- Computers are Hardware and Software
- A program is ....
- Elements of a program
- IDLE
- Pseudocode
- Writing an Algorithm
- Errors
Programming ... large and small ... is an art and a science

Q: These all depict programs ... what do they have in common? How are they different?

A nice game of thermonuclear war

“things”
Cascading down

Programs (apps) on your smartphone

A simple python program
Programming ... large and small ... is an art and a science

Art

Programs are carefully crafted
There is a “logic” to how they work
User’s experience (colors, fonts, etc.)
Accessibility, and ease of use

VS
Programming ... large and small ... is an art and a science

Art

- Programs are carefully crafted
- There is a “logic” to how they work
- User’s experience (colors, fonts, etc.)
- Accessibility, and ease of use

VS
Programming ... large and small ... is an art and a science

Art
- Programs are carefully crafted
- There is a “logic” to how they work
- User’s experience (colors, fonts, etc.)
- Accessibility, and ease of use

Science
- Follow strict syntax rules
- Issues of portability
- Debugging
- Backward compatibility
Programming ... large and small ... is an art and a science

Art
- Programs are carefully crafted
- There is a “logic” to how they work
- User’s experience (colors, fonts, etc.)
- Accessibility, and ease of use

Science
- Follow strict syntax rules
- Issues of portability
- Debugging
- Backward compatibility
Programming ... large and small ... is an art and a science

Art
- Programs are carefully crafted
- There is a “logic” to how they work
- User’s experience (colors, fonts, etc.)
- Accessibility, and ease of use

Science
- Follow strict syntax rules
- Issues of portability
- Debugging
- Backward compatibility

Although in this course you won’t learn about effective UI (User Interface) design, nor about portability issues ... nor will you write huge programs ... instead ...

Goal of this course : proceed from problem -> program (implementation of solution)
Programming ... large and small ... is an art and a science

Real-world scenario .... Write a “program” to keep track of employee work hours
Goal of this course : proceed from problem -> program (implementation of solution)

Q: What do I mean by “solution” to the above scenario?
Programming ... large and small ... is an art and a science

Real-world scenario .... Write a “program” to keep track of employee work hours
Goal of this course : proceed from problem --> program (implementation of solution)

Solutions:
(as perceived by the end-user)
Programming ... large and small ... is an art and a science

Real-world scenario .... Write a “program” to keep track of employee work hours
Goal of this course : proceed from problem -> program (implementation of solution)

Solutions:
(as perceived by the end-user)

Programs:
(implementation of a program)
Programming ... large and small ... is an art and a science

Real-world scenario .... Write a “program” to keep track of employee work hours
Goal of this course : proceed from problem -> program (implementation of solution)

Solutions: (as perceived by the end-user)

Programs: (implementation of a program)

There may be (usually is) more than one “solution“ to the problem
Computers are Hardware and Software

Input devices
(supply data)
Computers are Hardware and Software

Processing and Storage
(do “something” with the data)

CPU

Main memory

Secondary Storage Devices
Computers are Hardware and Software

Processing and Storage
(do “something” with the data)

CPU
Central Processing Unit

Main memory
Short term
(goes away when computer or program is turned off)

Secondary Storage Devices
Long term memory
CPU – The “brain” of the computer

CPU
Central Processing Unit

Q: What does a CPU “do”?  
Q: How does a CPU work?
CPU – The “brain” of the computer

Q: What does a CPU “do”?  
Q: How does a CPU work?

The hardware/circuitry component of “how” a computer does “stuff” you’ll learn about when you take CSCI247 … for the time being, just appreciate that there’s lots of “stuff” going on
CPU – The “brain” of the computer

CPU
Central Processing Unit

A sample program:
*Multiply 3 by 4 and add 2*
*Print to screen*

Q: How does a computer perform this task?
CPU – The “brain” of the computer

Step 1: Fetch first instruction: “Multiply 3 and 4”

Computers execute instructions in a program sequentially.

A sample program:

Multiply 3 by 4 and add 2
Print to screen

Q: How does a computer perform this task?
CPU – The “brain” of the computer

Step 1: Fetch first instruction: “Multiply 3 and 4”

Step 2: Decode instruction: convert to computer “instructions”

“decoding” means taking the “code” in a program and moving data around from memory, to registers, ALUs, etc.

(remember, you’ll learn about this in CSCI247, so for the time being … just appreciate that there’s quite a bit going on)

Sample a sample program

**Multiply 3 by 4** and add 2

Print to screen

Q: How does a computer perform this task?
CPU – The “brain” of the computer

CPU
Central Processing Unit

A sample program:

\textbf{Multiply 3 by 4} and add 2
\textit{Print to screen}

Q: How does a computer perform this task?

Step 1: Fetch first instruction: “\textbf{Multiply 3 and 4}”

Step 2: Decode instruction: convert to computer “instructions”

→ Step 3: Execute: calculate product of 3 and 4; save into memory

The basic components of a computer that do “work” are called Arithmetic Logical Units (ALUs) ... they are circuits that perform simple mathematical tasks

(remember, you’ll learn about this in CSCI247, so for the time being ... just appreciate that there’s quite a bit going on)
CPU – The “brain” of the computer

CPU
Central Processing Unit

Step 1: Fetch first instruction: “Multiply 3 and 4”
Step 2: Decode instruction: convert to computer “instructions”
Step 3: Execute: calculate product of 3 and 4; save into memory
Step 4: Fetch second: add 2 to result in memory

A sample program:
Multiply 3 by 4 and add 2
Print to screen

Q: How does a computer perform this task?
CPU – The “brain” of the computer

CPU
Central Processing Unit

A sample program:
*Multiply 3 by 4 and add 2*

*Print to screen*

Q: How does a computer perform this task?

Step 1: Fetch first instruction: “Multiply 3 and 4”
Step 2: Decode instruction: convert to computer “instructions”
Step 3: Execute: calculate product of 3 and 4; save into memory
Step 4: Fetch second: add 2 to result in memory
Step 5: Decode instruction: convert to computer “instructions”
CPU – The “brain” of the computer

A sample program:

Multiply 3 by 4 and add 2
Print to screen

Q: How does a computer perform this task?

Step 1: Fetch first instruction: “Multiply 3 and 4”
Step 2: Decode instruction: convert to computer “instructions”
Step 3: Execute: calculate product of 3 and 4; save into memory
Step 4: Fetch second: add 2 to result in memory
Step 5: Decode instruction: convert to computer “instructions”
Step 6: Execute: add 2 and 12; save to memory
CPU – The “brain” of the computer

**Central Processing Unit**

**A sample program:**

*Multiply 3 by 4 and add 2*

Q: How does a computer perform this task?

Step 1: Fetch first instruction: “Multiply 3 and 4”

Step 2: Decode instruction: convert to computer “instructions”

Step 3: Execute: calculate product of 3 and 4; save into memory

Step 4: Fetch second: add 2 to result in memory

Step 5: Decode instruction: convert to computer “instructions”

Step 6: Execute: add 2 and 12; save to memory

Step 7: Fetch instruction: **Print to screen**
CPU – The “brain” of the computer

CPU
Central Processing Unit

A sample program:
*Multiply 3 by 4 and add 2*

Print to screen

Q: How does a computer perform this task?

Step 1: Fetch first instruction: “Multiply 3 and 4”
Step 2: Decode instruction: convert to computer “instructions”
Step 3: Execute: calculate product of 3 and 4; save into memory
Step 4: Fetch second: add 2 to result in memory
Step 5: Decode instruction: convert to computer “instructions”
Step 6: Execute: add 2 and 12; save to memory
Step 7: Fetch instruction: *Print to screen*
Step 8: Decode instruction: convert to computer “instructions”
CPU – The “brain” of the computer

A sample program:

*Multiply 3 by 4 and add 2*

- **Step 1:** Fetch first instruction: “Multiply 3 and 4”
- **Step 2:** Decode instruction: convert to computer “instructions”
- **Step 3:** Execute: calculate product of 3 and 4; save into memory
- **Step 4:** Fetch second: add 2 to result in memory
- **Step 5:** Decode instruction: convert to computer “instructions”
- **Step 6:** Execute: add 2 and 12; save to memory
- **Step 7:** Fetch instruction: **Print to screen**
- **Step 8:** Decode instruction: convert to computer “instructions”
- **Step 9:** Execute: print to screen “14”

Q: How does a computer perform this task?
A sample program:

Multiply 3 by 4 and add 2
Print to screen

Q: How does a computer perform this task?

Q: What is the python “code” that “solves” this task?

Take-home message
(or, “what’s all this got to do with CSCI 141?”)

• Python (as are most languages in use today) is a High Level Language (HLL), which means that you do NOT have to worry about how data is moved around in the CPU, how ALUs work, etc. ... all that you are concerned about is converting a “task” ...
A sample program:

Multiply 3 by 4 and add 2
Print to screen

Q: How does a computer perform this task?

Python (as are most languages in use today) is a High Level Language (HLL), which means that you do NOT have to worry about how data is moved around in the CPU, how ALUs work, etc. ... all that you are concerned about is converting a “task” ... into a sequence of syntactically valid python “instructions” that the computer can understand and execute as you intended them to be performed.

Take-home message
(or, “what’s all this got to do with CSCI 141?”)
In this course, you’ll use **IDLE**, which is an **Integrated Development Environment (IDE)** for Python

An IDE is an easy-to-use software application for writing computer programs.
In this course, you’ll use **IDLE**, which is an Integrated Development Environment (IDE) for Python.

You use an IDE to write code ... but the code has to follow the syntax (rules) of the programming language.
In this course, you’ll use **IDLE**, which is an **Integrated Development Environment (IDE)** for Python.

You use an IDE to write code ... but the code has to follow the syntax (rules) of the programming language.

Modern-day IDEs are helpful and easy to use because they use color to help make the code more readable ... here, the “print” command is colored purple to specify that print is a valid python method.

There are several ways that a **print** routine in python can be invoked. One of them is to include in parentheses what you want printed. In this case, inside of the parentheses there is a numerical calculation, which python will perform first, and THEN print the result to the screen.
In this course, you’ll use **IDLE**, which is an **Integrated Development Environment** (IDE) for Python. You use an IDE to write code ... but the code has to follow the syntax (rules) of the programming language. Modern-day IDEs are helpful and easy to use because they use color to help make the code more readable ... here, the “print” command is colored purple to specify that print is a valid python *method*. To “run” a python program in IDLE, select *run module* from the Run option, or press F5 on the keyboard.
In this course, you’ll use **IDLE**, which is an **Integrated Development Environment** (IDE) for Python.

You use an IDE to write code … but the code has to follow the syntax (rules) of the programming language.

Modern-day IDEs are helpful and easy to use because they use color to help make the code more readable … here, the “print” command is colored purple to specify that print is a valid python *method*.

To “run” a python program in IDLE, select *run module* from the Run option, or press F5 on the keyboard.

Q: What would happen if you had typed “printt”?
In this course, you’ll use **IDLE**, which is an **Integrated Development Environment** (IDE) for Python.

You use an IDE to write code ... but the code has to follow the syntax (rules) of the programming language.

Modern-day IDEs are helpful and easy to use because they use color to help make the code more readable ... here, the “print” command is colored purple to specify that print is a valid python *method*.

To “run” a python program in IDLE, select *run module* from the Run option, or press F5 on the keyboard.

The word is not colored purple in IDLE ...
In this course, you’ll use **IDLE**, which is an Integrated Development Environment (IDE) for Python.

You use an IDE to write code ... but the code has to follow the syntax (rules) of the programming language.

Modern-day IDEs are helpful and easy to use because they use color to help make the code more readable ... here, the “print” command is colored purple to specify that print is a valid python *method*.

To “run” a python program in IDLE, select *run module* from the Run option, or press F5 on the keyboard.

The word is not colored purple in IDLE ... and when run the program produces an error.
In this course, you’ll use **IDLE**, which is an Integrated Development Environment (IDE) for Python. You use an IDE to write code ... but the code has to follow the syntax (rules) of the programming language. Modern-day IDEs are helpful and easy to use because they use color to help make the code more readable ... here, the “print” command is colored purple to specify that print is a valid python method. To “run” a python program in IDLE, select *run module* from the Run option, or press F5 on the keyboard.

Learning how to code means becoming familiar with what routines are available, and reading the errors and using them to inform you where the code needs to be fixed.
Pseudocode; writing an algorithm

What is an algorithm?
This is often the most difficult part for beginning students ... *Where/how do you start?*

---

**Goal:**

- Problem that needs to be solved
- Write a program
Pseudocode; writing an algorithm

This is often the most difficult part for beginning students ... **Where/how do you start?**

```
Problem
Goal: that needs to be solved

“I need a tool that adds three numbers”
```

Write a program
Pseudocode; writing an algorithm

This is often the most difficult part for beginning students ... **Where/how do you start?**

Q: **How should you proceed?**

A. Sit down and type random characters and cross your fingers and hope that you’ll miraculous write code that accomplishes the task
B. Sit down and start writing code
C. Brainstorm, create a rough outline of what you want to do, and THEN write code

---

Goal: Problem that needs to be solved

Write a program

“I need a tool that adds three numbers”
Pseudocode; writing an algorithm

This is often the most difficult part for beginning students ... **Where/how do you start?**

**Goal:** Problem that needs to be solved

- “I need a tool that adds three numbers”
- “I need to sort 3,000,000 social security numbers”

**Write a program**

**Q: How should you proceed?**

**A.** Sit down and type random characters and cross your fingers and hope that you’ll miraculous write code that accomplishes the task

**B.** Sit down and start writing code

**C.** Brainstorm, create a rough outline of what you want to do, and THEN write code

**Western**

**CSCS 141**

**Computer Programming I**
Pseudocode; writing an algorithm

Q: If you know the syntax of Python, can you write good code?
Pseudocode; writing an algorithm

Q: If you know the syntax of Python, can you write good code?

No ... Just because you “know” the words doesn’t mean that you can use them effectively

Q: So how will you learn to write good code?
Q: If you know the syntax of Python, can you write good code?

No ... Just because you “know” the words doesn’t mean that you can use them effectively

Q: So how will you learn to write good code? Practice, Practice, Practice
Pseudocode; writing an algorithm

Q: If you know the syntax of Python, can you write good code?

No ... Just because you “know” the words doesn’t mean that you can use them effectively

Q: So how will you learn to write good code? Practice, Practice, Practice

There’s a commonly used approach to writing code ... **pseudocode**

Q: What does this mean?
Pseudocode; writing an algorithm

Q: If you know the syntax of Python, can you write good code?

No ... Just because you “know” the words doesn’t mean that you can use them effectively

Q: So how will you learn to write good code? Practice, Practice, Practice

There’s a commonly used approach to writing code ... pseudocode

An analogy: painting

- A first draft
- Not too detailed, but not too abstract
- Can be refined several times and each revision results in a more detailed “draft”
Pseudocode; writing an algorithm

Q: If you know the syntax of Python, can you write good code?

No ... Just because you “know” the words doesn’t mean that you can use them effectively

Q: So how will you learn to write good code? Practice, Practice, Practice

There’s a commonly used approach to writing code ... pseudocode

The “idea”
The “inspiration”

• A first draft
• Not too detailed, but not too abstract
• Can be refined several times and each revision results in a more detailed “draft”
Pseudocode; writing an algorithm

Q: If you know the syntax of Python, can you write good code?

No ... Just because you “know” the words doesn’t mean that you can use them effectively

Q: So how will you learn to write good code? Practice, Practice, Practice

There’s a commonly used approach to writing code ... pseudocode

A first draft
• A first draft
• Not too detailed, but not too abstract
• Can be refined several times and each revision results in a more detailed “draft”
Pseudocode; writing an algorithm

Q: If you know the syntax of Python, can you write good code?

No ... Just because you “know” the words doesn’t mean that you can use them effectively

Q: So how will you learn to write good code? Practice, Practice, Practice

There’s a commonly used approach to writing code ... **pseudocode**

- A first draft
- Not too detailed, but not too abstract
- Can be refined several times and each revision results in a more detailed “draft”
Q: If you know the syntax of Python, can you write good code?

No ... Just because you “know” the words doesn’t mean that you can use them effectively

Q: So how will you learn to write good code? Practice, Practice, Practice

There’s a commonly used approach to writing code ... **pseudocode**

- A first draft
- Not too detailed, but not too abstract
- Can be refined several times and each revision results in a more detailed “draft”
Pseudocode ... an example

Task: Write a program that prompts a user for non-negative integers, sums them, and outputs their sum.

As “humans” this is an “easy” task ... but remember that a computer program is executed on a computer that follows instructions EXACTLY as they are supplied to the CPU.

Q: What is the sum of 2, 6, 8, 19, and 11?
Q: What is the sum of 2, 32, 44, 19, 11?
Q: What is the sum of ....
Pseudocode ... an example

Task: Write a program that prompts a user for non-negative integers, sums them, and outputs their sum

As “humans” this is an “easy” task ... but remember that a computer program is executed on a computer that follows instructions EXACTLY as they are supplied to the CPU

Pseudocode on-the-board exercise
Task: Write a program that prompts a user for non-negative integers, sums them, and outputs their sum

As “humans” this is an “easy” task ... but remember that a computer program is executed on a computer that follows instructions EXACTLY as they are supplied to the CPU

Possible pseudocode

1. Prompt user to enter first number
2. Save the first number into “memory”
3. Set sum to be the first number
4. Prompt user to enter another number
5. Save the number into “memory”
6. Increment the sum by the number in memory
7. Repeat Steps 4-6 until user enters -1
8. Print the sum
Pseudocode ... an example

Task: Write a program that prompts a user for non-negative integers, sums them, and outputs their sum

As “humans” this is an “easy” task ... but remember that a computer program is executed on a computer that follows instructions EXACTLY as they are supplied to the CPU

Possible pseudocode

1. Prompt user to enter first number
2. Save the first number into “memory”
3. Set sum to be the first number
4. Prompt user to enter another number
5. Save the number into “memory”
6. Increment the sum by the number in memory
7. Repeat Steps 4-6 until user enters -1
8. Print the sum

Once you have a rough idea of the “logic” of your program, you rely on python syntax to convert the pseudocode to “real” code
Pseudocode ... an example

Task: Write a program that prompts a user for non-negative integers, sums them, and outputs their sum

As “humans” this is an “easy” task ... but remember that a computer program is executed on a computer that follows instructions EXACTLY as they are supplied to the CPU

Possible pseudocode

1. Prompt user to enter first number
2. Save the first number into “memory”
3. Set sum to be the first number
4. Prompt user to enter another number
5. Save the number into “memory”
6. Increment the sum by the number in memory
7. Repeat Steps 4-6 until user enters -1
8. Print the sum
Pseudocode ... an example

Task: Write a program that prompts a user for non-negative integers, sums them, and outputs their sum

As “humans” this is an “easy” task ... but remember that a computer program is executed on a computer that follows instructions EXACTLY as they are supplied to the CPU

Possible pseudocode

1. Prompt user to enter first number
2. Save the first number into “memory”
3. Set sum to be the first number
4. Prompt user to enter another number
5. Save the number into “memory”
6. Increment the sum by the number in memory
7. Repeat Steps 4-6 until user enters -1
8. Print the sum
Pseudocode ... an example

Task: Write a program that prompts a user for non-negative integers, sums them, and outputs their sum.

As "humans" this is an "easy" task ... but remember that a computer program is executed on a computer that follows instructions EXACTLY as they are supplied to the CPU.

Possible pseudocode:

1. Prompt user to enter first number
2. Save the first number into "memory"
3. Set sum to be the first number
4. Prompt user to enter another number
5. Save the number into "memory"
6. Increment the sum by the number in memory
7. Repeat Steps 4-6 until user enters -1
8. Print the sum

```
num1 = int(input("Please enter a number "))
mySum = num1
while (num1 > -1):
    num1 = int(input("Please enter another number "))
    mySum = mySum + num1
print(mySum)
```
Q: If there are no syntax errors, will the code always run without producing an error?
Errors

Q: If there are no syntax errors, will the code always run without producing an error?

```
num1 = int(input("Please enter a number "))
mySum = num1
while (num1 > -1):
    num1 = int(input("Please enter another number "))
    mySum = mySum + num1
print(mySum)
```

Task: “read” the code

Q: What could go wrong when this program is run?

Task: demo using online interpreter
Up next

Labs (Tuesday and Wednesday)
Variables
Data types
Etc.