CSCI 141
Computer Programming I

Filip Jagodzinski
Announcements

Midterm Exam

- 12 February (Friday)
- In-class
- 50 minutes
- Closed Notes and Closed Book
- In addition to homework and labs, I recommend that you practice using the questions throughout the text AND exercises at the end of each chapter

- Extra (optional) review, 10 February, 5-6pm, CF125
Take home exercise

Write python code that will print the first 10 positive multiples of the number 34 whose ones digit is NOT 0 nor 5.

Hint: the digits are 34, 68, 102, 136, 204, 238, 272, 306, 374, 408

(Live demo)
From Last Time

```
for x in range (0, 100):
    if (x % 2 == 0):
        print (x)
```
From Last Time

These are all keywords, which specify important capabilities for your use when writing your python programs

for allows you to repeat a statement(s) multiple times
in allows you to assign a value, one-by-one, to a variable
if is the conditional functionality in python

Although the keywords may be specific to python (in is an example), the functionality of a for loop and conditional (if) statements are ubiquitous for all programming languages
From Last Time

```python
for x in range(0, 100):
    if (x % 2 == 0):
        print(x)
```

Here the `x` is a variable.

Once it is assigned a value, that value can be accessed over and over again.
range and print are functions (much more on this today)
From Last Time

```python
for x in range(0, 100):
    if (x % 2 == 0):
        print(x)
```

**range** and **print** are functions (much more on this today)

You can (should) think of them as black boxes, which may receive input, and which may produce or generate an output or perform some sort of functionality.

- **range** takes a list of numbers (0, 100) as input and outputs a list of even numbers.
- **print** takes a number (x) as input and outputs "text" to the screen.
From Last Time

Functions are easy to identify because the parentheses specify input argument(s)

In this case the range function receives as input 2 arguments (we’ve now seen that range can also receive three arguments)

The print function receives one input argument

Outputs “text” to the screen
From Last Time

These are **operators** (you CAN think of them as functions, too, because they receive input, the **operands**, and produce an “answer”, in this case == results in **True** or **False** and % results in a numerical value.

```python
for x in range (0, 100):
    if (x % 2 == 0):
        print (x)
```
From Last Time

Indentation is how you indicate in python which statement(s) constitutes the code block for a for loop, or a code block for an if statement, etc.

```python
for x in range (0, 100):
    if (x % 2 == 0):
        print(x)
```
From Last Time

```python
for x in range (0, 100):
    if (x % 2 == 0):
        print (x)
```

Step-by-step execution
For the code snippet:

```python
for x in range(0, 100):
    if (x % 2 == 0):
        print(x)
```

The range function generates a sequence of numbers from 0 to 100, inclusive. The `if` statement checks if the number is even (i.e., divisible by 2) and prints it if true.
From Last Time

```
for x in range (0, 100):
    if (x % 2 == 0):
        print (x)
```

Each time that \( x \) is assigned a value, the code block for the for loop is executed.
From Last Time

```python
for x in range(0, 100):
    if (x % 2 == 0):
        print(x)
```

Is the value of $x$ modulo 2 equal to zero?
From Last Time

```
for x in range (0, 100):
    if (x % 2 == 0):
        print (x)
```

Is the value of $x$ modulo 2 equal to zero?

Yes, therefore execute the body of the `if` statement ...

And print the value of $x$

Output

0
From Last Time

```
for x in range (0, 100):
    if (x % 2 == 0):
        print (x)
```

The "next" value of \(x\) is 1

Output

0
From Last Time

```
for x in range(0, 100):
    if (x % 2 == 0):
        print(x)
```

The “next” value of $x$ is 1

Is the value of $x$ modulo 2 equal to zero?

Output

0
From Last Time

```python
for x in range(0, 100):
    if (x % 2 == 0):
        print(x)
```

The “next” value of \( x \) is 1

Is the value of \( x \) modulo 2 equal to zero?

No, therefore do NOT execute the body of the `if` statement ...

Output

0
From Last Time

```python
for x in range(0, 100):
    if (x % 2 == 0):
        print(x)
```

The “next” value of \( x \) is 2

\[ 0, 1, 2, \ldots, 98, 99 \]

**Output**

0
From Last Time

```python
for x in range(0, 100):
    if (x % 2 == 0):
        print(x)
```

The "next" value of \( x \) is 2

Is the value of \( x \) modulo 2 equal to zero?

Output

0
From Last Time

```python
for x in range (0, 100):
    if (x % 2 == 0):
        print (x)
```

The “next” value of `x` is 2

Is the value of `x` modulo 2 equal to zero?

Yes, therefore execute the body of the `if` statement ...

And print the value of `x`
From Last Time

```
for x in range (0, 100):
    if (x % 2 == 0):
        print (x)
```

The “next” value of $x$ is 3

Output

0
2
From Last Time

```python
for x in range(0, 100):
    if (x % 2 == 0):
        print(x)
```

The “next” value of \( x \) is 3

Is the value of \( x \) modulo 2 equal to zero?

Output
0
2
From Last Time

```python
for x in range(0, 100):
    if (x % 2 == 0):
        print(x)
```

The “next” value of $x$ is 3

Is the value of $x$ modulo 2 equal to zero?

No, therefore do NOT execute the body of the if statement ...

This process will be repeated until ...

Output

0

2
From Last Time

```
for x in range (0, 100):
    if (x % 2 == 0):
        print (x)
```

The “next” value of `x` is 98

Output

```
0
2
4
...
96
```
From Last Time

```python
for x in range(0, 100):
    if (x % 2 == 0):
        print(x)
```

The “next” value of $x$ is 98

Is the value of $x$ modulo 2 equal to zero?

Output

0
2
4
...
96
From Last Time

```python
for x in range (0, 100):
    if (x % 2 == 0):
        print (x)
```

The “next” value of \( x \) is 98

Is the value of \( x \) modulo 2 equal to zero?

Yes, therefore execute the body of the if statement ...

And print the value of \( x \)

Output

0
2
4
...
96
98
From Last Time

```python
for x in range (0, 100):
    if (x % 2 == 0):
        print (x)
```

Output

```
0
2
4
...
96
98
```

Task: Explain in “words” what happens next
Today

while loops
for loop iteration with Lists
Functions
For loops ... iterating over a list of items

```
for aNum in range(1, 5) :
    # Code block
```

This for loop iterates as many times as there are entries in the list output by the `range` function.
For loops ... iterating over a list of items

```
for aNum in range(1, 5):
    # Code block
```

This for loop iterates as many times as there are entries in the list output by the `range` function.

The `range` function outputs a “list” of integer values.

Q: Can we create a for loop that relies on a list of values that are NOT integers?
For loops ... iterating over a list of items

```python
for aNum in range(1, 5) :
    # Code block
```

This for loop iterates as many times as there are entries in the list output by the `range` function.

The `range` function outputs a “list” of integer values.

Q: Can we create a for loop that relies on a list of values that are NOT integers?

... there is another format for the python for loop.

It can be used to iterate over a list of items, which can be **Strings**.
For loops ... iterating over a list of items

```
for lectureDays in [ "Mon", "Wed", "Fri" ]:
    print ("Lectures are on", lectureDays)
```

Learning by example
For loops ... iterating over a list of items

```python
for lectureDays in ["Mon", "Wed", "Fri"]:
    print("Lectures are on", lectureDays)
```

Several things to notice ... this looks very similar to a for loop which

- Uses the keywords `for` and `in`
For loops ... iterating over a list of items

```python
for lectureDays in [ "Mon", "Wed", "Fri" ]:
    print ("Lectures are on", lectureDays)
```

Several things to notice ... this looks very similar to a for loop which

- Uses the keywords `for` and `in`
- A **colon** that specifies the “end” of the for loop declaration
For loops ... iterating over a list of items

```python
for lectureDays in [ "Mon", "Wed", "Fri" ] :
    print ("Lectures are on", lectureDays)
```

Several things to notice ... this looks very similar to a for loop which

- **Uses the keywords** `for` and `in`
- **A colon** that specifies the “end” of the for loop declaration
- **Code block for the “body”** of the for loop which is indented
For loops ... iterating over a list of items

```python
for lectureDays in [ "Mon", "Wed", "Fri" ]:
    print ("Lectures are on", lectureDays)
```

Several things to notice ... this looks very similar to a for loop which

- Uses the keywords `for` and `in`
- A colon that specifies the “end” of the for loop declaration
- Code block for the “body” of the for loop which is indented
- A variable `lectureDays` that is the iterator variable

**Q: What is different in this for loop compared to what you’ve already seen?**
For loops ... iterating over a list of items

```python
for lectureDays in ["Mon", "Wed", "Fri"] :
    print ("Lectures are on", lectureDays)
```

What is different is that the range function is not being used in this case. Instead ...
For loops ... iterating over a list of items

```python
for lectureDays in [ "Mon", "Wed", "Fri" ]:
    print ("Lectures are on", lectureDays)
```

What is different is that the range function is not being used in this case. Instead ...

- Use of square brackets that specify a list
For loops ... iterating over a list of items

```python
for lectureDays in ["Mon", "Wed", "Fri"]:
    print("Lectures are on", lectureDays)
```

What is different is that the range function is not being used in this case. Instead ...

- Use of square brackets that specify a list
- Items in the list are separated by commas
For loops ... iterating over a list of items

```python
for lectureDays in ["Mon", "Wed", "Fri"]:
    print ("Lectures are on", lectureDays)
```

What is different is that the range function is not being used in this case. Instead ...

- Use of square brackets that specify a list
- Items in the list are separated by commas
- In this case, the list items are Strings
For loops ... iterating over a list of items

```
for lectureDays in [ "Mon", "Wed", "Fri" ]:
    print ("Lectures are on", lectureDays)
```

The variable `lectureDays` will be assigned a value, one-turn-at-a-time for each item in the list. For each item in the list, the body, or code block, for the for loop, will be executed.

**Q: What is the output of these lines of Python code?**
For loops ... iterating over a list of items

```python
for lectureDays in ["Mon", "Wed", "Fri"]:
    print("Lectures are on", lectureDays)
```

The variable `lectureDays` will be assigned a value, one-turn-at-a-time for each item in the list. For each item in the list, the body, or code block, for the for loop, will be executed.

**Q: What is the output of these lines of Python code?**

To answer this question, first determine how many items there are in the list. Then execute the for loop by executing the code block that is the body of the for loop.
For loops ... iterating over a list of items

```python
for lectureDays in ["Mon", "Wed", "Fri"]:
    print ("Lectures are on", lectureDays)
```

When `lectureDays` has the value “Mon”, the following is printed:

Lectures are on Mon
For loops … iterating over a list of items

```python
for lectureDays in ["Mon", "Wed", "Fri"]:
    print("Lectures are on ", lectureDays)
```

When `lectureDays` has the value “Wed”, the following is printed:

Lectures are on Wed
For loops ... iterating over a list of items

```python
for lectureDays in ["Mon", "Wed", "Fri"]:
    print("Lectures are on", lectureDays)
```

When `lectureDays` has the value “Fri”, the following is printed:

```
Lectures are on Fri
```

Fri
For loops ... iterating over a list of items

```
for lectureDays in [ "Mon", "Wed", "Fri" ] :
    print ("Lectures are on", lectureDays)
```

Q: What is the output of these lines of Python code?
For loops ... iterating over a list of items

```python
for lectureDays in [ "Mon", "Wed", "Fri" ]:
    print ("Lectures are on", lectureDays)
```

Q: What is the output of these lines of Python code?

```
Lectures are on Mon
Lectures are on Wed
Lectures are on Fri
```
Iterating ... another approach

```
for lectureDays in [ "Mon", "Wed", "Fri" ]:
    print ("Lectures are on", lectureDays)
```

```
for x in range (0, 100):
    if (x % 2 == 0):
        print (x)
```

Task: Draw the flow diagram for a for loop
Iterating ... another approach

```python
for lectureDays in ["Mon", "Wed", "Fri"]:
    print("Lectures are on", lectureDays)

for x in range(0, 100):
    if (x % 2 == 0):
        print(x)
```

Regardless of whether a for loop uses a sequence of numbers, or a list of strings, as the sequence of items for which it iterates, there is an initial test step, followed by either the for loop's termination or another iteration of the loop's body.
Iterating ... another approach

Regardless of whether a for loop uses a sequence of numbers, or a list of strings, as the sequence of items for which it iterates, there is an initial test step, followed by either the for loops termination or another iteration of the loop’s body.

Task: Use the python keyword `for` to create a for loop that iterates indefinitely.

```python
for lectureDays in ["Mon", "Wed", "Fri"]:
    print ("Lectures are on", lectureDays)
```

```python
for x in range(0, 100):
    if (x % 2 == 0):
        print (x)
```
Iterating ... another approach

```python
for lectureDays in ["Mon", "Wed", "Fri"]:
    print("Lectures are on", lectureDays)
```

```python
for x in range(0, 100):
    if (x % 2 == 0):
        print(x)
```

Regardless of whether a for loop uses a sequence of numbers, or a list of strings, as the sequence of items for which it iterates, there is an initial test step, followed by either the for loops termination or another iteration of the loop’s body.

**Task:** Use the python keyword `for` to create a for loop that iterates indefinitely.

Unlike many other programming languages, you cannot (easily) make a for loop iterate indefinitely.
Task: Write python code so that the word “Still going” is printed over and over again, indefinitely.
Iterating ... another approach

Task: Write python code so that the word “Still going” is printed over and over again, indefinitely

```python
goOn = True
while (goOn == True):
    print ("Still going")
```
Task: Write Python code so that the word “Still going” is printed over and over again, indefinitely.

```python
goOn = True
while (goOn == True):
    print("Still going")
```

The Python keyword `while`.
Iterating ... another approach

Task: Write Python code so that the word “Still going” is printed over and over again, indefinitely.

```python
goOn = True
while (goOn == True):
    print("Still going")
```

The Python keyword `while`
Iterating ... another approach

Task: Write python code so that the word “Still going” is printed over and over again, indefinitely

```python
goOn = True
while (goOn == True):
    print("Still going")
```

The python keyword `while`

code block

a Boolean value, or variable of type Boolean, or Boolean expression
Iterating ... another approach

**Task:** Write python code so that the word “Still going” is printed over and over again, indefinitely

```python
goOn = True
while (goOn == True):
    print("Still going")
```

The python keyword **while**

a Boolean value, or variable of type Boolean, or Boolean expression

Only if the Boolean expression (or value) is True is the code block (body) of the while loop executed
Iterating ... another approach

Task: Write python code so that the word “Still going” is printed over and over again, indefinitely

```python
goOn = True
while (goOn == True):
    print ("Still going")
```

Let’s be a bit more elegant and more like “real” programmers ...

```python
goOn = True
while (goOn):
    print ("Still going")
```

Q: What is the output of either of these programs?
Iterating ... another approach

Task: Write python code so that the word “Still going” is printed over and over again, indefinitely

```python
goOn = True
while (goOn == True):
    print("Still going")
```

Let’s be a bit more elegant and more like “real” programmers ...

```python
goOn = True
while (goOn):
    print("Still going")
```

Q: What is the output of either of these programs?
Iterating ... another approach

Task: Write python code so that the word “Still going” is printed over and over again, indefinitely

```python
# Let's be a bit more elegant and more like "real" programmers...
goOn = True
while (goOn == True):
    print ("Still going")
```

Regardless of how you write/use the Boolean expression/value/variable that determines if the body of the while loop should be executed again...

Be sure that you can write the control flow diagram for a while loop
Iterating ... another approach

Task: Write python code so that the word “Still going” is printed over and over again, indefinitely

```python
goOn = True
while (goOn == True):
    print ("Still going")
```

Let’s be a bit more elegant and more like “real” programmers ...

```python
goOn = True
while (goOn):
    print ("Still going")
```

Q: Why do we have both for loops and while loop?

Task: Give an example of a program that is better suited for using while instead of for
Iterating ... another approach

Answer: A program that has a loop that continues based on a user’s input.
Iterating ... another approach

Answer: A program that has a loop that continues based on a user’s input.

Task: write a program that prompts a user to input the number of quarters that have been found. The program should then ask, “How many additional quarters have you found?” If the user enters 0, the program ends. If the user enters a positive integer, the running total of quarters should be displayed, and the question asked again.

This is the “prose” description of a program.

Decompose it to write pseudocode and then a python program.
Answer: A program that has a loop that continues based on a user’s input.

Task: write a program that prompts a user to input the number of quarters that have been found. The program should then ask, “How many additional quarters have you found?” If the user enters 0, the program ends. If the user enters a positive integer, the running total of quarters should be displayed, and the question asked again.

This is the “prose” description of a program.

Decompose it to write pseudocode and then a python program.

Look closely at the words in this verbal description of a program:

- Prompt
- Input
- Number
- Ask
- If 0
- If > 0
- Running Total
- Asked again
Iterating ... another approach

Answer: A program that has a loop that continues based on a user’s input.

Task: write a program that prompts a user to input the number of quarters that have been found. The program should then ask, “How many additional quarters have you found?” If the user enters 0, the program ends. If the user enters a positive integer, the running total of quarters should be displayed, and the question asked again.

This is the “prose” description of a program.

Decompose it to write pseudocode and then a python program.

Look closely at the words in this verbal description of a program:

- Prompt
- Input
- Number
- Ask
- If 0
- If > 0
- Running Total
- Asked again

You could try and code this using a for loop (is it possible?), but because the cycle of asking might be indefinite, a while loop is better.

Knowing whether you should use for or while is a skill that you’ll gain from experience.
Iterating ... another approach

Answer: A program that has a loop that continues based on a user’s input.

Task: write a program that prompts a user to input the number of quarters that have been found. The program should then ask, “How many additional quarters have you found?” If the user enters 0, the program ends. If the user enters a positive integer, the running total of quarters should be displayed, and the question asked again.

This is the “prose” description of a program. Decompose it to write pseudocode and then a Python program.

Q: How might you code this portion of the program?
Iterating ... another approach

Answer: A program that has a loop that continues based on a user’s input.

This is the “prose” description of a program.

Decompose it to write pseudocode and then a Python program.

Task: write a program that prompts a user to input the number of quarters that have been found. The program should then ask, “How many additional quarters have you found?” If the user enters 0, the program ends. If the user enters a positive integer, the running total of quarters should be displayed, and the question asked again.

```
qFound = int(input("How many quarters "))
```

Decide on a variable name Use of input, int and the assignment operator (=)
Answer: A program that has a loop that continues based on a user’s input.

Task: write a program that prompts a user to input the number of quarters that have been found. The program should then ask, “How many additional quarters have you found?” If the user enters 0, the program ends. If the user enters a positive integer, the running total of quarters should be displayed, and the question asked again.

Q: How might you code this part of the program?

This is the “prose” description of a program.

Decompose it to write pseudocode and then a python program:

```python
qFound = int(input("How many quarters 
"))
```
Answer: A program that has a loop that continues based on a user’s input.

Task: write a program that prompts a user to input the number of quarters that have been found. The program should then ask, “How many additional quarters have you found?” If the user enters 0, the program ends. If the user enters a positive integer, the running total of quarters should be displayed, and the question asked again.

```
qFound = int(input("How many quarters "))
```

```
qFound = int(input("How many additional quarters "))
```

This is the “prose” description of a program.

Decompose it to write pseudocode and then a python program.

This is that part of programming that constitutes the “art” of writing code. Through practice, you’ll learn how to structure all of the pieces together ...
Answer: A program that has a loop that continues based on a user’s input.

Task: Write a program that prompts a user to input the number of quarters that have been found. The program should then ask, “How many additional quarters have you found?” If the user enters 0, the program ends. If the user enters a positive integer, the running total of quarters should be displayed, and the question asked again.

This is the “prose” description of a program.

Decompose it to write pseudocode and then a Python program:

```python
qFound = int(input("How many quarters "))
```

```python
qFound = int(input("How many additional quarters "))
```

Q: How should you code this portion of the program?
Iterating ... another approach

Answer : A program that has a loop that continues based on a user’s input

Task : write a program that prompts a user to input the number of quarters that have been found. The program should then ask, “How many additional quarters have you found?” If the user enters 0, the program ends. If the user enters a positive integer, the running total of quarters should be displayed, and the question asked again.

This is the “prose” description of a program.

Decompose it to write pseudocode and then a python program

```
qFound = int(input("How many quarters "))

qFound = int(input("How many additional quarters "))
```

Q: How should you code this portion of the program

You might be tempted to use an if statement ... but why might using an if statement not be the best choice in this case?
Answer: A program that has a loop that continues based on a user’s input.

Task: write a program that prompts a user to input the number of quarters that have been found. The program should then ask, “How many additional quarters have you found?” If the user enters 0, the program ends. If the user enters a positive integer, the running total of quarters should be displayed, and the question asked again.

This is the “prose” description of a program. Decompose it to write pseudocode and then a Python program.

Eventually the program will prompt again, perhaps indefinitely many times, and we’ve just seen that using `for` cannot be made to loop indefinitely.
Answer: A program that has a loop that continues based on a user’s input.

Task: write a program that prompts a user to input the number of quarters that have been found. The program should then ask, “How many additional quarters have you found?” If the user enters 0, the program ends. If the user enters a positive integer, the running total of quarters should be displayed, and the question asked again.

This is the “prose” description of a program.

Decompose it to write pseudocode and then a python program.

Q: How should you code this portion of the program?

```python
qFound = int(input("How many quarters "))
while (qFound > 0):
    qFound = int(input("How many additional quarters "))
```
Iterating ... another approach

Answer: A program that has a loop that continues based on a user’s input.

Task: write a program that prompts a user to input the number of quarters that have been found. The program should then ask, “How many additional quarters have you found?” If the user enters 0, the program ends. If the user enters a positive integer, the running total of quarters should be displayed, and the question asked again.

This is the “prose” description of a program.

Decompose it to write pseudocode and then a python program:

```python
qFound = int(input("How many quarters "))
while (qFound > 0):
    qFound = int(input("How many additional quarters "))
```

Q: How should you code this portion of the program?
Iterating ... another approach

Answer: A program that has a loop that continues based on a user’s input.

Task: Write a program that prompts a user to input the number of quarters that have been found. The program should then ask, “How many additional quarters have you found?” If the user enters 0, the program ends. If the user enters a positive integer, the running total of quarters should be displayed, and the question asked again.

This is the “prose” description of a program. Decompose it to write pseudocode and then a python program.

```python
qFound = int(input("How many quarters "))
qTotal = qFound
while (qFound > 0):
    print("Woot. You have", qTotal, "quarters")
    qFound = int(input("How many additional quarters "))
    qTotal = qTotal + qFound
```

Q: How should you code this portion of the program?

You decide on the “sum” variable, which is updated each time that the body of the while loop is executed, and you keep track of the running total.
Up until this point, we’ve seen several **functions** (I’ve used that word repeatedly, and by now you know that you should think of a function as a “black box” which his code that performs some sort of task)

- `range(0, 100)` → A list of numbers
- `print(x)` → Outputs “text” to the screen
Up until this point, we’ve see several **functions** (I’ve used that word repeatedly, and by now you know that you should think of a function as a “black box” which his code that performs some sort of task)

range(1, 5)
print("Hello")
randrange(3,50)
int("43")
type(16)

Q: What do these functions “do”?
Up until this point, we’ve see several **functions** (I’ve used that word repeatedly, and by now you know that you should think of a function as a “black box” which his code that performs some sort of task)

Python doesn’t contain ALL of the possible functions that you might want to use in a program.

For example

- A function that adds three numbers and then subtracts 5 from the sum
- A function that prints “Happy birthday” four times

**Task : Draw “black box” diagrams for these functions**

(on the board exercise)
Functions

Up until this point, we’ve see several **functions** (I’ve used that word repeatedly, and by now you know that you should think of a function as a “black box” which his code that performs some sort of task)

Looking at the “black” boxes on the board ...

Just as the **range** and **print** functions have a name, and take a specified number of “inputs” and produce an “output” or perform a task ... if you want to write your own function to perform a sequence of calculations, you must also:

- Give your function a name
- Specify its parameters (input(s))
- Designate what the function “does”

```python
def name( parameters ):
    statements
```
Define a function name `printUserWord`, which prompts the user to input a word, and also prompts the user to specify how many times that word should be printed.

The function should then print that word to the screen as many times as the user has indicated.
Functions that return values