Preliminaries: 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 is scheduled for Wednesday, 10 February, 5-6pm, CF 125
Preliminaries: Announcements

Learning how to debug

Q: Where is there a syntax error in this code?

```python
for dayOfMonth in range(1,30):
    print("value of dayOfMonth is", dayOfMonth

myFancyNewVariable = False
```
Learning how to debug

Q: Where is there a syntax error in this code?

```python
for dayOfMonth in range(1,30):
    print("value of dayOfMonth is", dayOfMonth)
myFancyNewVariable = False
```

The `print` function is missing the closing parenthesis.

Q: When you run the program, where does the interpret tell you there is an error?
Preliminaries: Announcements

Learning how to debug

Q: Where is there a syntax error in this code?

```python
for dayOfMonth in range(1,30):
    print("value of dayOfMonth is", dayOfMonth

myFancyNewVariable = False
```

Lesson learned: Although the invalid syntax is on THIS line of code, that invalid syntax is caused by a previous line of code.
Q: What does the following code print to the screen?

```python
def tellThatSomethingTrueIsFalse(aStatement):
    print(aStatement, "is not true");
def tellThatSomethingFalseIsTrue(aStatement):
    print(aStatement, "is true");
claim = "Monkeys can fly"
if (True):
    tellThatSomethingTrueIsFalse(claim)
else:
    tellThatSomethingFalseIsTrue(claim)
```

A. is not true  
B. is true  
C. Monkeys can fly  
D. Monkeys can fly is not true  
E. Monkeys can fly is true
Q: What does the following code print to the screen?

```python
def tellThatSomethingTrueIsFalse(aStatement):
    print(aStatement, "is not true");
def tellThatSomethingFalseIsTrue(aStatement):
    print(aStatement, "is true");
claim = "Monkeys can fly"
if (True):
    tellThatSomethingTrueIsFalse(claim)
else:
    tellThatSomethingFalseIsTrue(claim)
```

A. is not true
B. is true
C. Monkeys can fly
D. Monkeys can fly is not true
E. Monkeys can fly is true

These are function declarations. We know this because the keyword `def` is used.
Q: What does the following code print to the screen?

```python
def tellThatSomethingTrueIsFalse(aStatement):
    print(aStatement, "is not true");
def tellThatSomethingFalseIsTrue(aStatement):
    print(aStatement, "is true");
claim = "Monkeys can fly"
if (True) :
    tellThatSomethingTrueIsFalse(claim)
else :
    tellThatSomethingFalseIsTrue(claim)
```

This is declaring a variable named `claim`, and assigning it the string literal "Monkeys can fly"

A. is not true
B. is true
C. Monkeys can fly
D. Monkeys can fly is not true
E. Monkeys can fly is true
Q: What does the following code print to the screen?

```
def tellThatSomethingTrueIsFalse(aStatement):
    print(aStatement, "is not true");
def tellThatSomethingFalseIsTrue(aStatement):
    print(aStatement, "is true");
claim = "Monkeys can fly"
if (True):
    tellThatSomethingTrueIsFalse(claim)
else:
    tellThatSomethingFalseIsTrue(claim)
```

This is an if, else selection statement.

The condition is True, which is ALWAYS True.

A. is not true
B. is true
C. Monkeys can fly
D. Monkeys can fly is not true
E. Monkeys can fly is true
Q: What does the following code print to the screen?

```python
def tellThatSomethingTrueIsFalse(aStatement):
    print(aStatement, "is not true");
def tellThatSomethingFalseIsTrue(aStatement):
    print(aStatement, "is true");
claim = "Monkeys can fly"
if True:
    tellThatSomethingTrueIsFalse(claim)
else:
    tellThatSomethingFalseIsTrue(claim)
```

Therefore the statement(s) in this code block are executed. This invokes a function, and passes as “input” the value of the variable claim

A. is not true
B. is true
C. Monkeys can fly
D. Monkeys can fly is not true
E. Monkeys can fly is true
From Last Time ... warmup

Q: What does the following code print to the screen?

```python
def tellThatSomethingTrueIsFalse(aStatement):
    print(aStatement, "is not true");
def tellThatSomethingFalseIsTrue(aStatement):
    print(aStatement, "is true");
claim = "Monkeys can fly"
if (True):
tellThatSomethingTrueIsFalse(claim)
else:
tellThatSomethingFalseIsTrue(claim)
```

The function “receives” the input and the value of that input is used in a print statement.

- A. is not true
- B. is true
- C. Monkeys can fly
- D. Monkeys can fly is not true
- E. Monkeys can fly is true

Correct answer: E. Monkeys can fly is true
Q: What is one advantage of thinking of functions as black boxes?
From Last Time

3,6 → add2Nums → “The sum is 9”

Q: What is one advantage of thinking of functions as black boxes?

```python
def add2Nums(number1, number2):
    summation = number1 + number2
    print("The sum is", summation)
```

Once you’ve written (and tested) the function, you can “forget” about the code that is in its interior.
From Last Time

Q: What is one advantage of thinking of functions as black boxes?

Once you’ve written (and tested) the function, you can “forget” about the code that is in its interior.

This is a skill that will permit you to write more and more complicated programs ... knowing ALL of the details becomes too daunting of a task, and sometimes it is good to think about the “big picture” only.
Take home exercise

Write a program that

- Declares a function that computes the fourth power of an integer
- Prompts the user three times for an integer input
- After each prompt, the program should invoke the custom function which should print to the screen the fourth power of the supplied number.

Q: How should you approach this task?
Take home exercise

Write a program that

- Declares a function that computes the fourth power of an integer
- Prompts the user three times for an integer input
- After each prompt, the program should invoke the custom function which should print to the screen the fourth power of the supplied number.

Q: What are three steps for declaring a function?  Q: How is a function invoked?
CSCI 141 Computer Programming I

Take home exercise

Write a program that

- Declares a function that computes the fourth power of an integer
- Prompts the user three times for an integer input
- After each prompt, the program should invoke the custom function which should print to the screen the fourth power of the supplied number.

Q: What are three steps for declaring a function?

- Name the function
- Decide on inputs (if any)
- Write the “body” of the function

Q: How is a function invoked?

You write the name of the function and provide all of the necessary “inputs”

Live demo
Take home exercise  Sample Solutions

```python
def calcFourthPower(aNumber):
    result = aNumber * aNumber * aNumber * aNumber
    print("The fourth power of", aNumber, "is", result)
```

Q: Does this function perform the desired task?
Q: Is the function named appropriately?
Take home exercise  

Sample Solutions

def calcFourthPower(aNumber):
    result = aNumber * aNumber * aNumber * aNumber
    print("The fourth power of", aNumber, "is", result)

def computeFourthPower(number):
    result = number ** 4
    print("The fourth power of", number, "is", result)

Q: Does this function perform the desired task?
Q: Is the function named appropriately?
Q: Can we write another version that performs the same task but is 2 lines of code?
Take home exercise

Sample Solutions

def calcFourthPower(aNumber):
    result = aNumber * aNumber * aNumber * aNumber
    print("The fourth power of", aNumber, "is", result)

def computeFourthPower(number):
    result = number ** 4
    print("The fourth power of", number, "is", result)

def calculate4thPower(num):
    print("The fourth power of", num, "is", num**4)

Q: Does this function perform the desired task?
Q: Is the function named appropriately?
All of these are good function names because they are descriptive AND they have a verb as part of the name. This is desired because functions “do” things.

A bad declaration:  
def aFunction(aNumber)
These (local) variables are also named nicely because their name helps us to identify what sort of data they represent.

A Not so good declaration:  
```python
def calculateResult(myVariable)
```

Take home exercise  Sample Solutions

```python
def calcFourthPower(aNumber):
    result = aNumber * aNumber * aNumber * aNumber
    print("The fourth power of", aNumber, "is", result)

def computeFourthPower(number):
    result = number ** 4
    print("The fourth power of", number, "is", result)

def calculate4thPower(num):
    print("The fourth power of", num, "is", num**4)
```

Q: How do we **invoke** these functions?
def calcFourthPower(aNumber):
    result = aNumber * aNumber * aNumber * aNumber
    print("The fourth power of", aNumber, "is", result)

def computeFourthPower(number):
    result = number ** 4
    print("The fourth power of", number, "is", result)

def calculate4thPower(num):
    print("The fourth power of", num, "is", num**4)

Q: How do we **invoke** these functions?

calcFourthPower(2)
computeFourthPower(3)
calculate4thPower(4)
Take home exercise  Sample Solutions

```python
def calcFourthPower(aNumber):
    result = aNumber * aNumber * aNumber * aNumber
    print("The fourth power of", aNumber, "is", result)

def computeFourthPower(number):
    result = number ** 4
    print("The fourth power of", number, "is", result)

def calculate4thPower(num):
    print("The fourth power of", num, "is", num**4)
```

Q: How do we **invoke** these functions?

```
calcFourthPower(2)  # The fourth power of 2 is 16
computeFourthPower(3)  # The fourth power of 3 is 81
calculate4thPower(4)  # The fourth power of 4 is 256
```
Function parameters
Functions that return values
Now that we are a bit more familiar with WHAT a function is, let’s describe a bit more about what REALLY happens … let us return to the code we’ve already seen …

```python
def tellThatSomethingTrueIsFalse(aStatement):
    print(aStatement, "is not true");
def tellThatSomethingFalseIsTrue(aStatement):
    print(aStatement, "is true");
claim = "Monkeys can fly"
if (True) :
    tellThatSomethingTrueIsFalse(claim)
else :
    tellThatSomethingFalseIsTrue(claim)
```

Q: What does it mean when we say that a functions receives input?
Function, objects, and parameters

```python
def tellThatSomethingTrueIsFalse(aStatement):
    print(aStatement, "is not true");
def tellThatSomethingFalseIsTrue(aStatement):
    print(aStatement, "is true");

claim = "Monkeys can fly"
if (True):
    tellThatSomethingTrueIsFalse(claim)
else:
    tellThatSomethingFalseIsTrue(claim)
```

This code declares two functions ...
Function, objects, and parameters

String
Monkeys can fly

```python
def tellThatSomethingTrueIsFalse(aStatement):
    print(aStatement, "is not true");
def tellThatSomethingFalseIsTrue(aStatement):
    print(aStatement, "is true");

claim = "Monkeys can fly"
if (True):
    tellThatSomethingTrueIsFalse(claim)
else:
    tellThatSomethingFalseIsTrue(claim)
```

... and creates a variable and binds it to an object of type String
Function, objects, and parameters

String

Monkeys can fly

def tellThatSomethingTrueIsFalse(aStatement):
    print(aStatement, "is not true");

def tellThatSomethingFalseIsTrue(aStatement):
    print(aStatement, "is true");

claim = "Monkeys can fly"
if (True):
    tellThatSomethingTrueIsFalse(claim)
else:
    tellThatSomethingFalseIsTrue(claim)

the conditional statement then executes one of the functions ....
String
Monkeys can fly

```python
def tellThatSomethingTrueIsFalse(aStatement):
    print(aStatement, "is not true");
def tellThatSomethingFalseIsTrue(aStatement):
    print(aStatement, "is true");

claim = "Monkeys can fly"
if (True):
    tellThatSomethingTrueIsFalse(claim)
else:
    tellThatSomethingFalseIsTrue(claim)
```

Notice that when we invoke the function, we are writing `claim`, but the function declaration uses the variable name `aStatement`.

**Q: How are these related?**
Monkeys can fly

```
def tellThatSomethingTrueIsFalse(aStatement):
    print(aStatement, "is not true");

def tellThatSomethingFalseIsTrue(aStatement):
    print(aStatement, "is true");

claim = "Monkeys can fly"
if (True):
    tellThatSomethingTrueIsFalse(claim)
else:
    tellThatSomethingFalseIsTrue(claim)
```

This technically means invoke the `tellThatSomethingTrueIsFalse` function, and passes as “input” the object that `claim` refers to.
Function, objects, and parameters

Monkeys can fly

```python
def tellThatSomethingTrueIsFalse(aStatement):
    print(aStatement, "is not true");

def tellThatSomethingFalseIsTrue(aStatement):
    print(aStatement, "is true");

claim = "Monkeys can fly"
if (True):
    tellThatSomethingTrueIsFalse(claim)
else:
    tellThatSomethingFalseIsTrue(claim)
```

This technically means invoke the `tellThatSomethingTrueIsFalse` function, and passes as “input” the object that `claim` refers to.

`tellThatSomethingTrueIsFalse` prints to the screen
Function, objects, and parameters

Monkeys can fly

```python
def tellThatSomethingTrueIsFalse(aStatement):
    print(aStatement, "is not true");
def tellThatSomethingFalseIsTrue(aStatement):
    print(aStatement, "is true");

claim = "Monkeys can fly"
if (True):
    tellThatSomethingTrueIsFalse(claim)
else:
    tellThatSomethingFalseIsTrue(claim)
```

If you go on in CS, you’ll learn the technical details of what it means to “pass” as input a value. It can be done via pass-by-value (you pass a primitive), or it can be done via pass-by-reference (you pass the address in memory where the data is located).

tellThatSomethingTrueIsFalse

prints to the screen
Function, objects, and parameters

Monkeys can fly

```python
def tellThatSomethingTrueIsFalse(aStatement):
    print(aStatement, "is not true");

def tellThatSomethingFalseIsTrue(aStatement):
    print(aStatement, "is true");
    claim = "Monkeys can fly"
    if True:
        tellThatSomethingTrueIsFalse(claim)
    else:
        tellThatSomethingFalseIsTrue(claim)
```

When the function `tellThatSomethingTrueIsFalse` executes, it creates a variable to refer to the input, and names that variable `aStatement`.

`tellThatSomethingTrueIsFalse` prints to the screen
When the function `tellThatSomethingTrueIsFalse` executes, it creates a variable to refer to the input, and names that variable `aStatement`.

There are now two variables, `aStatement` and `claim`, that refer to the same piece of data.
Monkeys can fly

```python
def tellThatSomethingTrueIsFalse(aStatement):
    print(aStatement, "is not true");

def tellThatSomethingFalseIsTrue(aStatement):
    print(aStatement, "is true");

claim = "Monkeys can fly"
if (True):
    tellThatSomethingTrueIsFalse(claim)
else:
    tellThatSomethingFalseIsTrue(claim)
```

When the function `tellThatSomethingTrueIsFalse` executes, it creates a variable to refer to the input, and names that variable `aStatement`.

Q: When the function `tellThatSomethingTrueIsFalse` is "using" the variable `aStatement`, what data is being referred to?
Function, objects, and parameters

When the body of the function `tellThatSomethingTrueIsFalse` uses the `aStatement` variable, it is referring to the String object with the value "Monkeys can fly".

Q: Can that variable `aStatement` be used OUTSIDE of the function `tellThatSomethingTrueIsFalse`?
Function, objects, and parameters

Monkeys can fly

```python
def tellThatSomethingTrueIsFalse(aStatement):
    print(aStatement, "is not true");
def tellThatSomethingFalseIsTrue(aStatement):
    print(aStatement, "is true");
claim = "Monkeys can fly"
if (True):
    tellThatSomethingTrueIsFalse(claim)
else:
    tellThatSomethingFalseIsTrue(claim)
```

**Variables and parameters defined in a function can ONLY be used in that function**

The technical term is that function variables and parameters have **local scope**.

Think of “scope” as “where the variable or parameter can be used”. Thus, local scope refers to the fact that a variable can only be used where it is defined.
Function, objects, and parameters

Each time that a function is invoked with has a locally scoped variable, the local scope variable is re-created at the start (when the function is invoked) and erased (made unavailable for use elsewhere) at the conclusion of the function.
Which line(s) of code contain an error(s)

```python
def printTheSecretWord(myInput):
    print("The secret word is", myInput)
    numOfCows = 32

aWord = "Banana"
printTheSecretWord(aWord)
print(numOfCows)
print(myInput)
```

A. 1  
B. 3  
C. 6  
D. 8  
E. 9
Both of the variables `numCows` and `myInput` are declared in the function `printTheSecretWord`, so those variables have scope only in THAT function.

- A. 1
- B. 3
- C. 6
- D. 8
- E. 9

On the board explanation
Function, objects, and parameters

Functions can also receive no, a single, or multiple arguments.
Function, objects, and parameters

Functions can also receive no, a single, or multiple arguments.

```python
def printTheWordHello():
    print("Hello")

def generatePersonalizedGreetings(aName, someNumber):
    for x in range(0, someNumber):
        print("Hello", aName)
```

How many functions are being declared?
How many inputs does the `printTheWordHello` function require?
How many inputs does the `generatePersonalizedGreetings` function require?
Function, objects, and parameters

Functions can also receive **no**, a single, or **multiple** arguments.

```python
def printTheWordHello():
    print("Hello")

def generatePersonalizedGreetings(aName, someNumber):
    for x in range(0, someNumber):
        print("Hello", aName)
```

**How are these functions invoked?**
Function, objects, and parameters

Functions can also receive no, a single, or multiple arguments.

```python
def printTheWordHello():
    print("Hello")

def generatePersonalizedGreetings(aName, someNumber):
    for x in range(0, someNumber):
        print("Hello", aName)

printTheWordHello()
generatePersonalizedGreetings("Filip", 4)
```
Function, objects, and parameters

Functions can also receive **no**, a single, or **multiple** arguments.

```python
def printTheWordHello():
    print("Hello")

def generatePersonalizedGreetings(aName, someNumber):
    for x in range(0, someNumber):
        print("Hello", aName)

printTheWordHello()
generatePersonalizedGreetings("Filip", 4)
```

Write the name of the function
Function, objects, and parameters

Functions can also receive **no**, a single, or **multiple** arguments.

```python
def printTheWordHello():
    print("Hello")

def generatePersonalizedGreetings(aName, someNumber):
    for x in range(0, someNumber):
        print("Hello", aName)

printTheWordHello()
generatePersonalizedGreetings("Filip", 4)
```

Use parentheses, and specify the correct number of inputs.
Function, objects, and parameters

Functions can also receive no, a single, or multiple arguments.

```python
def printTheWordHello():
    print("Hello")

def generatePersonalizedGreetings(aName, someNumber):
    for x in range(0, someNumber):
        print("Hello", aName)

printTheWordHello()
generatePersonalizedGreetings("Filip", 4)
```

Important: you must provide the inputs in the correct “order”
Function, objects, and parameters

Functions can also receive **no**, a single, or **multiple** arguments.

```python
def printTheWordHello():
    print("Hello")

def generatePersonalizedGreetings(aName, someNumber):
    for x in range(0, someNumber):
        print("Hello", aName)

printTheWordHello()
generatePersonalizedGreetings("Filip", 4)
```

What would happen if ...

```python
generatePersonalizedGreetings(4, "Filip")
```

(on the board explanation)
Functions can also return values. That means, that somewhere in the function’s code block, there is a `return` statement, that specifies “output” some value.
Functions can also return values. That means, that somewhere in the function’s code block, there is a `return` statement, that specifies “output” some value.

```python
def computeFourthPower(number):
    result = number ** 4
    return result
```

`return` is a keyword, which specifies, “output what is to the right of me”
Functions can also return values. That means, that somewhere in the function’s code block, there is a `return` statement, that specifies “output” some value.

```python
def computeFourthPower(number):
    result = number ** 4
    return result
```

Q: What does this function output?
Functions that return values

Functions can also return values. That means, that somewhere in the function’s code block, there is a `return` statement, that specifies “output” some value.

```python
def computeFourthPower(number):
    result = number ** 4
    return result
```

Q: How do you invoke and use a function that has a return value?
Functions can also return values. That means, that somewhere in the function’s code block, there is a `return` statement, that specifies “output” some value.

```python
def computeFourthPower(number):
    result = number ** 4
    return result

anOutput = computeFourthPower(23)
print("The result is", anOutput)
```

On the board explanation
Up next

More about functions