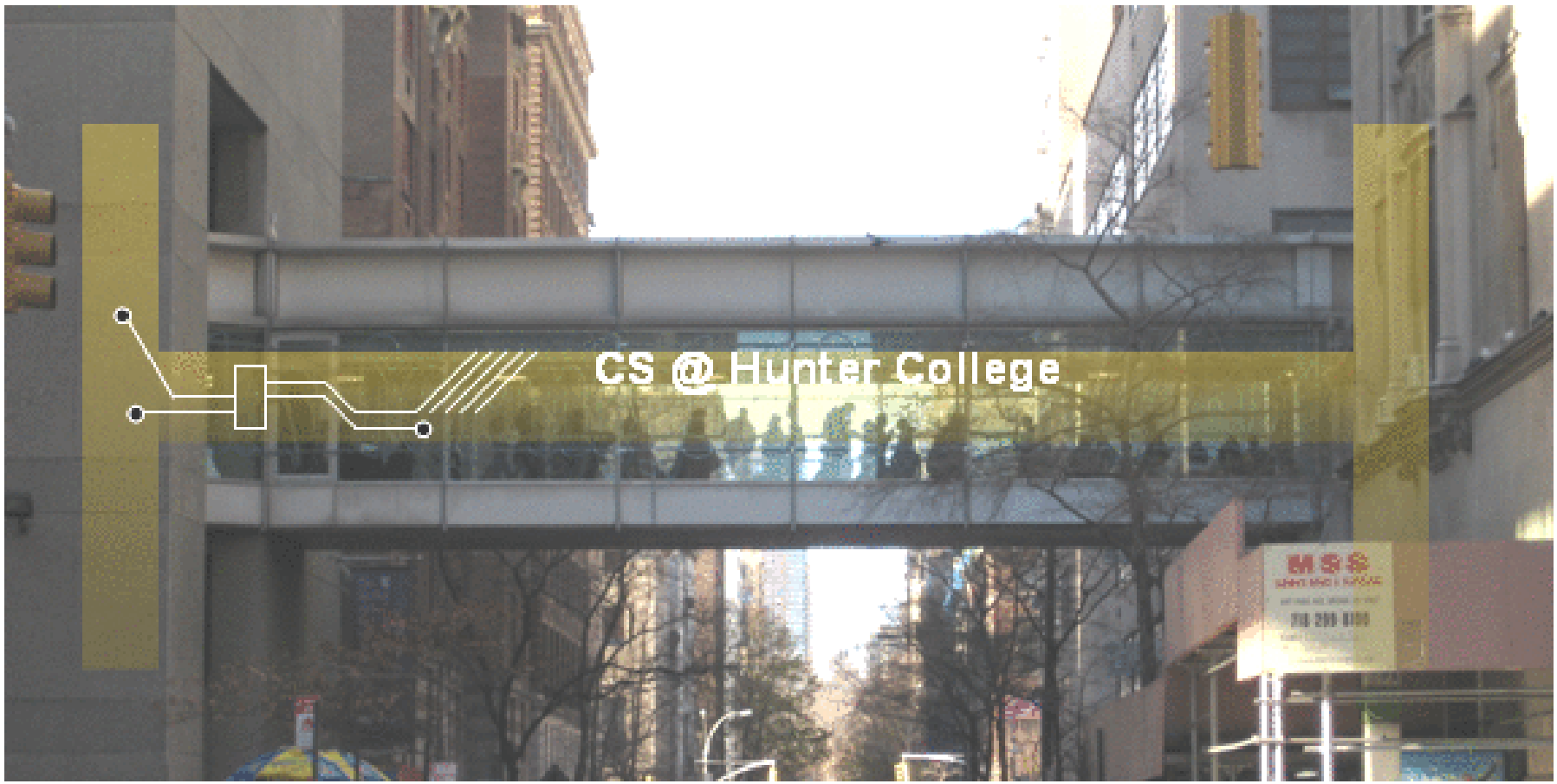
CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

Announcements

- Each lecture includes a survey of computing research and tech in NYC.

  Today: Prof. Jia Xu (machine translation)

Announcements

- Each lecture includes a survey of computing research and tech in NYC.

  Today: Prof. Jia Xu (machine translation)

- We've passed the halfway mark of the semester (Lecture 8 of 14).

Announcements

- Each lecture includes a survey of computing research and tech in NYC.

  Today: Prof. Jia Xu (machine translation)

- We've passed the halfway mark of the semester (Lecture 8 of 14).
  - Excellent job and keep up the good work!

Announcements

- Each lecture includes a survey of computing research and tech in NYC.

  Today: Prof. Jia Xu (machine translation)

- We've passed the halfway mark of the semester (Lecture 8 of 14).
  - Excellent job and keep up the good work!
  - Upcoming: More variety in lecturers as more CSci 127 Teaching Staff will be covering class segments.

Frequently Asked Questions

From lecture slips & recitation sections.

Frequently Asked Questions

From lecture slips & recitation sections.

- Why all the fuss about women in computer science? There's lots of women.

Frequently Asked Questions

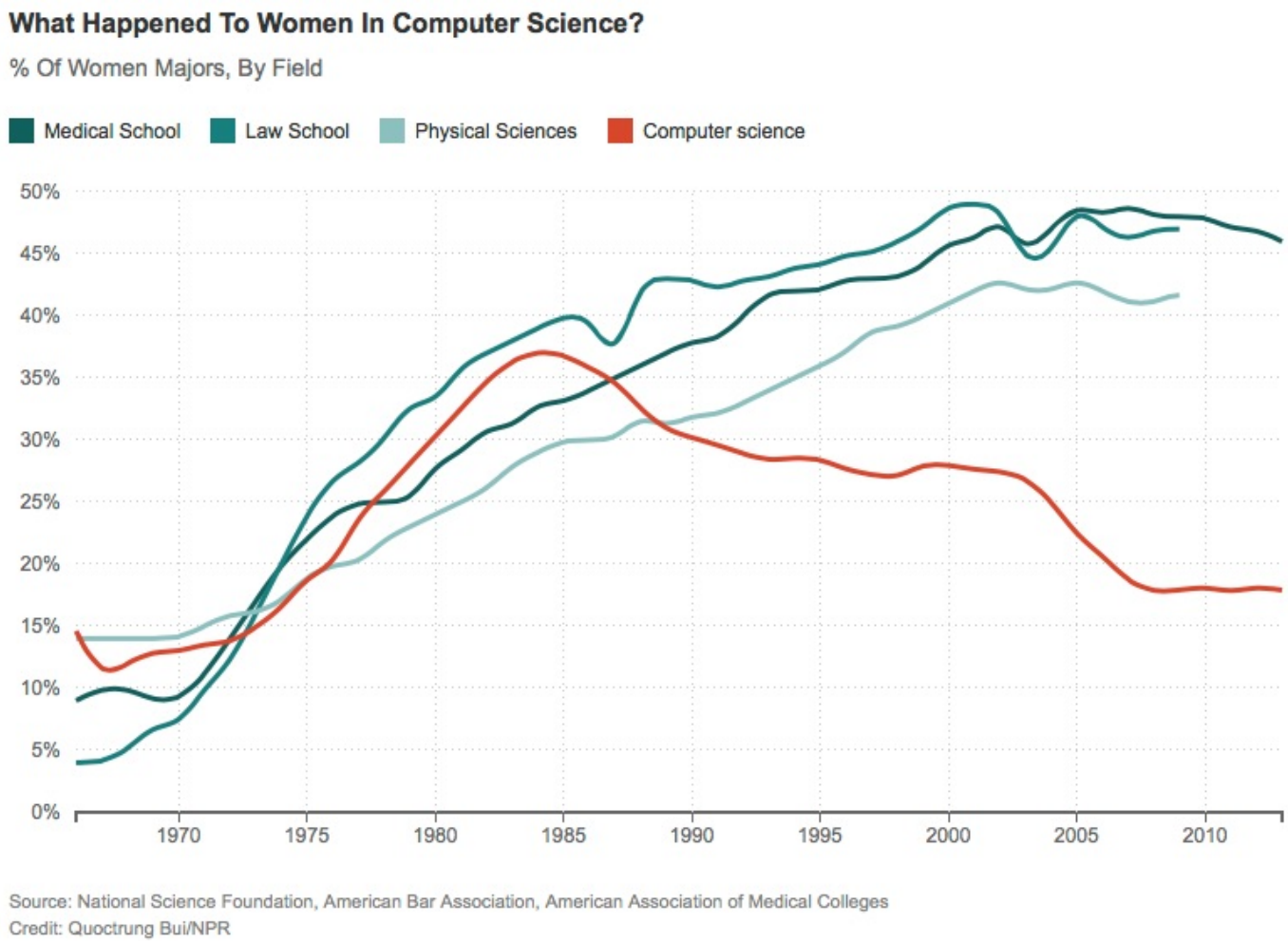
From lecture slips & recitation sections.

- Why all the fuss about women in computer science? There's lots of women. *Well, actually, there's not.*

Frequently Asked Questions

From lecture slips & recitation sections.

- Why all the fuss about women in computer science? There's lots of women. *Well, actually, there's not.*



Frequently Asked Questions

From lecture slips & recitation sections.

- Why all the fuss about women in computer science? There's lots of women. *Well, actually, there's not.*

Frequently Asked Questions

From lecture slips & recitation sections.

- Why all the fuss about women in computer science? There's lots of women. *Well, actually, there's not.*
- Yes, but I'm a guy. How does this help me?

Frequently Asked Questions

From lecture slips & recitation sections.

- Why all the fuss about women in computer science? There's lots of women.
  Well, actually, there's not.
- Yes, but I'm a guy. How does this help me?
  And it's not just women. Across CUNY there were < 1000 *students graduating with a tech degree (out of 24,000 graduates).*

Frequently Asked Questions

From lecture slips & recitation sections.

- Why all the fuss about women in computer science? There's lots of women.
  Well, actually, there's not.
- Yes, but I'm a guy. How does this help me?
  And it's not just women. Across CUNY there were < 1000 *students graduating with a tech degree (out of 24,000 graduates).*
  Versus Stanford where 90% take a computer science course & half major in computer science.

Frequently Asked Questions

From lecture slips & recitation sections.

- Why all the fuss about women in computer science? There's lots of women.
  Well, actually, there's not.
- Yes, but I'm a guy. How does this help me?
  And it's not just women. Across CUNY there were < 1000 *students graduating with a tech degree (out of 24,000 graduates).*
  Versus Stanford where 90% take a computer science course & half major in computer science.
- And how does this help me?

Frequently Asked Questions

From lecture slips & recitation sections.

- Why all the fuss about women in computer science? There's lots of women.
  Well, actually, there's not.
- Yes, but I'm a guy. How does this help me?
  And it's not just women. Across CUNY there were < 1000 *students graduating with a tech degree (out of 24,000 graduates).*
  Versus Stanford where 90% take a computer science course & half major in computer science.
- And how does this help me?
  Redesigned CSci 127:

Frequently Asked Questions

From lecture slips & recitation sections.

- Why all the fuss about women in computer science? There's lots of women.
  Well, actually, there's not.
- Yes, but I'm a guy. How does this help me?
  And it's not just women. Across CUNY there were < 1000 *students graduating with a tech degree (out of 24,000 graduates).*
  Versus Stanford where 90% take a computer science course & half major in computer science.
- And how does this help me?
  Redesigned CSci 127:
  - *Following Harvard: a rigorous & practical course to attract everyone.*

Frequently Asked Questions

From lecture slips & recitation sections.

- Why all the fuss about women in computer science? There's lots of women.
  Well, actually, there's not.
- Yes, but I'm a guy. How does this help me?
  And it's not just women. Across CUNY there were < 1000 *students graduating with a tech degree (out of 24,000 graduates).*
  Versus Stanford where 90% take a computer science course & half major in computer science.
- And how does this help me?
  Redesigned CSci 127:
  - *Following Harvard: a rigorous & practical course to attract everyone.*
  - *Automated grading to give immediate feedback.*

Frequently Asked Questions

From lecture slips & recitation sections.

- Why all the fuss about women in computer science? There's lots of women.
  Well, actually, there's not.
- Yes, but I'm a guy. How does this help me?
  And it's not just women. Across CUNY there were < 1000 *students graduating with a tech degree (out of 24,000 graduates).*
  Versus Stanford where 90% take a computer science course & half major in computer science.
- And how does this help me?
  Redesigned CSci 127:
  - *Following Harvard: a rigorous & practical course to attract everyone.*
  - *Automated grading to give immediate feedback.*
  - *Redirected resources to be student-focused (i.e. UTAs).*

Frequently Asked Questions

From lecture slips & recitation sections.

- Why all the fuss about women in computer science? There's lots of women.
  Well, actually, there's not.
- Yes, but I'm a guy. How does this help me?
  And it's not just women. Across CUNY there were < 1000 *students graduating with a tech degree (out of 24,000 graduates).*
  Versus Stanford where 90% take a computer science course & half major in computer science.
- And how does this help me?
  Redesigned CSci 127:
  - *Following Harvard: a rigorous & practical course to attract everyone.*
  - *Automated grading to give immediate feedback.*
  - *Redirected resources to be student-focused (i.e. UTAs).*
  - *Dedicated lab space & new computers.*

Frequently Asked Questions

From lecture slips & recitation sections.

- Why all the fuss about women in computer science? There's lots of women.
  Well, actually, there's not.
- Yes, but I'm a guy. How does this help me?
  And it's not just women. Across CUNY there were < 1000 *students graduating with a tech degree (out of 24,000 graduates).*
  Versus Stanford where 90% take a computer science course & half major in computer science.
- And how does this help me?
  Redesigned CSci 127:
  - *Following Harvard: a rigorous & practical course to attract everyone.*
  - *Automated grading to give immediate feedback.*
  - *Redirected resources to be student-focused (i.e. UTAs).*
  - *Dedicated lab space & new computers.*

  More students means more variety in upper division electives, more students with interests similar to yours, and more links to research and industry.

Frequently Asked Questions

From lecture slips & recitation sections.

- Why all the fuss about women in computer science? There's lots of women.
  Well, actually, there's not.
- Yes, but I'm a guy. How does this help me?
  And it's not just women. Across CUNY there were < 1000 *students graduating with a tech degree (out of 24,000 graduates).*
  Versus Stanford where 90% take a computer science course & half major in computer science.
- And how does this help me?
  Redesigned CSci 127:
  - *Following Harvard: a rigorous & practical course to attract everyone.*
  - *Automated grading to give immediate feedback.*
  - *Redirected resources to be student-focused (i.e. UTAs).*
  - *Dedicated lab space & new computers.*

  More students means more variety in upper division electives, more students with interests similar to yours, and more links to research and industry.
- Okay, but why the visitors?

Frequently Asked Questions

From lecture slips & recitation sections.

- Why all the fuss about women in computer science? There's lots of women.
  Well, actually, there's not.
- Yes, but I'm a guy. How does this help me?
  And it's not just women. Across CUNY there were < 1000 *students graduating with a tech degree (out of 24,000 graduates).*
  Versus Stanford where 90% take a computer science course & half major in computer science.
- And how does this help me?
  Redesigned CSci 127:
  - *Following Harvard: a rigorous & practical course to attract everyone.*
  - *Automated grading to give immediate feedback.*
  - *Redirected resources to be student-focused (i.e. UTAs).*
  - *Dedicated lab space & new computers.*

  More students means more variety in upper division electives, more students with interests similar to yours, and more links to research and industry.
- Okay, but why the visitors?
  Their support has made this possible.

Today's Topics

- Functions
- Prof. Xu
- Github
- Final Exam Overview

Functions

- Functions are a way to break code into pieces, that can be easily reused.

```
#Name:  your name here
#Date:  October 2017
#This program, uses functions,
#     says hello to the world!

def main():
     print("Hello, World!")

if __name__ == "__main__":
     main()
```

Functions

```
#Name:  your name here
#Date:  October 2017
#This program, uses functions,
#     says hello to the world!

def main():
     print("Hello, World!")

if __name__ == "__main__":
     main()
```

- Functions are a way to break code into pieces, that can be easily reused.
- Many languages require that all code must be organized with functions.

Functions

```
#Name:  your name here
#Date:  October 2017
#This program, uses functions,
#     says hello to the world!

def main():
     print("Hello, World!")

if __name__ == "__main__":
     main()
```

- Functions are a way to break code into pieces, that can be easily reused.
- Many languages require that all code must be organized with functions.
- The opening function is often called `main()`

Functions

```
#Name:  your name here
#Date:  October 2017
#This program, uses functions,
#     says hello to the world!

def main():
     print("Hello, World!")

if __name__ == "__main__":
     main()
```

- Functions are a way to break code into pieces, that can be easily reused.
- Many languages require that all code must be organized with functions.
- The opening function is often called `main()`
- You **call** or **invoke** a function by typing its name, followed by any input parameters, surrounded by parenthesis:

Functions

```
#Name:  your name here
#Date:  October 2017
#This program, uses functions,
#     says hello to the world!

def main():
     print("Hello, World!")

if __name__ == "__main__":
     main()
```

- Functions are a way to break code into pieces, that can be easily reused.
- Many languages require that all code must be organized with functions.
- The opening function is often called `main()`
- You **call** or **invoke** a function by typing its name, followed by any input parameters, surrounded by parenthesis:
  Example: `print("Hello", "World")`

Functions

```
#Name:  your name here
#Date:  October 2017
#This program, uses functions,
#     says hello to the world!

def main():
     print("Hello, World!")

if __name__ == "__main__":
     main()
```

- Functions are a way to break code into pieces, that can be easily reused.
- Many languages require that all code must be organized with functions.
- The opening function is often called `main()`
- You **call** or **invoke** a function by typing its name, followed by any input parameters, surrounded by parenthesis:
  Example: `print("Hello", "World")`
- Can write, or **define** your own functions,

Functions

```
#Name:  your name here
#Date:  October 2017
#This program, uses functions,
#     says hello to the world!

def main():
     print("Hello, World!")

if __name__ == "__main__":
     main()
```

- Functions are a way to break code into pieces, that can be easily reused.
- Many languages require that all code must be organized with functions.
- The opening function is often called `main()`
- You **call** or **invoke** a function by typing its name, followed by any input parameters, surrounded by parenthesis:
  Example: `print("Hello", "World")`
- Can write, or **define** your own functions, which are stored, until invoked or called.

In Pairs or Triples:

Predict what the code will do:

```
#Greet loop example

def greetLoop(person):
    print("Greetings")
    for i in range(5):
        print("Hello", person)

greetLoop("Thomas")
```

```
# From "Teaching with Python" by John Zelle

def happy():
    print("Happy Birthday to you!")

def sing(P):
    happy()
    happy()
    print("Happy Birthday dear " + P + "!")
    happy()


sing("Fred")
sing("Thomas")
sing("Hunter")
```

In Pairs or Triples:

Predict what the code will do:

```
#Greet loop example

def greetLoop(person):
    print("Greetings")
    for i in range(5):
        print("Hello", person)

greetLoop("Thomas")
```

```
# From "Teaching with Python" by John Zelle

def happy():
    print("Happy Birthday to you!")

def sing(P):
    happy()
    happy()
    print("Happy Birthday dear " + P + "!")
    happy()


sing("Fred")
sing("Thomas")
sing("Hunter")
```

```
def prob4():
    verse = "jam tomorrow and jam yesterday,"
    print("The rule is,")
    c = mystery(verse)
    w = enigma(verse,c)
    print(c,w)
def mystery(v):
    print(v)
    c = v.count("jam")
    return(c)
def enigma(v,c):
    print("but never", v[-1])
    for i in range(c):
        print("jam")
    return("day.")
prob4()
```

```
#Fall 2013 Final Exam, 5

def kuwae( inLst ):
    tot = 1
    for item in inLst:
        tot = tot * item
    return tot

def foo( inLst ):
    if ( inLst[-1] > inLst[0] ):
        return kuwae( inLst )
    else:
        return -1

foo( [2, 4, 6, 8] )

foo( [4002, 328, 457, 1] )
```

Python Tutor

```
#Greet loop example

def greetLoop(person):
    print("Greetings")
    for i in range(5):
        print("Hello", person)

greetLoop("Thomas")
```

```
# From "Teaching with Python" by John Zelle

def happy():
    print("Happy Birthday to you!")

def sing(P):
    happy()
    happy()
    print("Happy Birthday dear " + P + "!")
    happy()


sing("Fred")
sing("Thomas")
sing("Hunter")
```

(Demo with `pythonTutor`)

Input Parameters & Return Values

- Functions can have **input parameters**.

```
def totalWithTax(food,tip):
    total = 0
    tax = 0.0875
    total = food + food * tax
    total = total + tip
    return(total)

lunch = float(input('Enter lunch total: '))
lTip = float(input('Enter lunch tip:' ))
lTotal = totalWithTax(lunch, lTip)
print('Lunch total is', lTotal)

dinner= float(input('Enter dinner total: '))
dTip = float(input('Enter dinner tip:' ))
dTotal = totalWithTax(dinner, dTip)
print('Dinner total is', dTotal)
```

Input Parameters & Return Values

```
def totalWithTax(food,tip):
    total = 0
    tax = 0.0875
    total = food + food * tax
    total = total + tip
    return(total)

lunch = float(input('Enter lunch total: '))
lTip = float(input('Enter lunch tip:' ))
lTotal = totalWithTax(lunch, lTip)
print('Lunch total is', lTotal)

dinner= float(input('Enter dinner total: '))
dTip = float(input('Enter dinner tip:' ))
dTotal = totalWithTax(dinner, dTip)
print('Dinner total is', dTotal)
```

- Functions can have **input parameters**.
- Surrounded by parenthesis, both in the function definition, and in the function call (invocation).

Input Parameters & Return Values

```
def totalWithTax(food,tip):
    total = 0
    tax = 0.0875
    total = food + food * tax
    total = total + tip
    return(total)

lunch = float(input('Enter lunch total: '))
lTip = float(input('Enter lunch tip:' ))
lTotal = totalWithTax(lunch, lTip)
print('Lunch total is', lTotal)

dinner= float(input('Enter dinner total: '))
dTip = float(input('Enter dinner tip:' ))
dTotal = totalWithTax(dinner, dTip)
print('Dinner total is', dTotal)
```

- Functions can have **input parameters**.
- Surrounded by parenthesis, both in the function definition, and in the function call (invocation).
- The "placeholders" in the function definition: **formal parameters**.

Input Parameters & Return Values

```
def totalWithTax(food,tip):
    total = 0
    tax = 0.0875
    total = food + food * tax
    total = total + tip
    return(total)

lunch = float(input('Enter lunch total: '))
lTip = float(input('Enter lunch tip:' ))
lTotal = totalWithTax(lunch, lTip)
print('Lunch total is', lTotal)

dinner= float(input('Enter dinner total: '))
dTip = float(input('Enter dinner tip:' ))
dTotal = totalWithTax(dinner, dTip)
print('Dinner total is', dTotal)
```

- Functions can have **input parameters**.
- Surrounded by parenthesis, both in the function definition, and in the function call (invocation).
- The "placeholders" in the function definition: **formal parameters**.
- The ones in the function call: **actual parameters**

Input Parameters & Return Values

```
def totalWithTax(food,tip):
    total = 0
    tax = 0.0875
    total = food + food * tax
    total = total + tip
    return(total)

lunch = float(input('Enter lunch total: '))
lTip = float(input('Enter lunch tip:' ))
lTotal = totalWithTax(lunch, lTip)
print('Lunch total is', lTotal)

dinner= float(input('Enter dinner total: '))
dTip = float(input('Enter dinner tip:' ))
dTotal = totalWithTax(dinner, dTip)
print('Dinner total is', dTotal)
```

- Functions can have **input parameters**.
- Surrounded by parenthesis, both in the function definition, and in the function call (invocation).
- The "placeholders" in the function definition: **formal parameters**.
- The ones in the function call: **actual parameters**
- Functions can also **return values** to where it was called.

Input Parameters & Return Values

```
def totalWithTax(food,tip):
    total = 0
    tax = 0.0875
    total = food + food * tax
    total = total + tip
    return(total)

lunch = float(input('Enter lunch total: '))
lTip = float(input('Enter lunch tip:' ))
lTotal = totalWithTax(lunch, lTip)
print('Lunch total is', lTotal)

dinner= float(input('Enter dinner total: '))
dTip = float(input('Enter dinner tip:' ))
dTotal = totalWithTax(dinner, dTip)
print('Dinner total is', dTotal)
```

Formal Parameters

Actual Parameters

- Functions can have **input parameters**.
- Surrounded by parenthesis, both in the function definition, and in the function call (invocation).
- The "placeholders" in the function definition: **formal parameters**.
- The ones in the function call: **actual parameters**.
- Functions can also **return values** to where it was called.

Input Parameters

```
def totalWithTax(food,tip):
    total = 0
    tax = 0.0875
    total = food + food * tax
    total = total + tip
    return(total)

lunch = float(input('Enter lunch total: '))
lTip = float(input('Enter lunch tip:' ))
lTotal = totalWithTax(lunch, lTip)
print('Lunch total is', lTotal)

dinner= float(input('Enter dinner total: '))
dTip = float(input('Enter dinner tip:' ))
dTotal = totalWithTax(dinner, dTip)
print('Dinner total is', dTotal)
```

Formal Parameters

Actual Parameters

- When called, the actual parameter values are copied to the formal parameters.

Input Parameters

```
def totalWithTax(food,tip):
    total = 0
    tax = 0.0875
    total = food + food * tax
    total = total + tip
    return(total)

lunch = float(input('Enter lunch total: '))
lTip = float(input('Enter lunch tip:' ))
lTotal = totalWithTax(lunch, lTip)
print('Lunch total is', lTotal)

dinner= float(input('Enter dinner total: '))
dTip = float(input('Enter dinner tip:' ))
dTotal = totalWithTax(dinner, dTip)
print('Dinner total is', dTotal)
```

Formal Parameters

Actual Parameters

- When called, the actual parameter values are copied to the formal parameters.
- All the commands inside the function are performed on the copies.

Input Parameters

```
def totalWithTax(food,tip):
    total = 0
    tax = 0.0875
    total = food + food * tax
    total = total + tip
    return(total)

lunch = float(input('Enter lunch total: '))
lTip = float(input('Enter lunch tip:' ))
lTotal = totalWithTax(lunch, lTip)
print('Lunch total is', lTotal)

dinner= float(input('Enter dinner total: '))
dTip = float(input('Enter dinner tip:' ))
dTotal = totalWithTax(dinner, dTip)
print('Dinner total is', dTotal)
```

Formal Parameters

Actual Parameters

- When called, the actual parameter values are copied to the formal parameters.
- All the commands inside the function are performed on the copies.
- The actual parameters do not change.

Input Parameters

```
def totalWithTax(food,tip):
    total = 0
    tax = 0.0875
    total = food + food * tax
    total = total + tip
    return(total)

lunch = float(input('Enter lunch total: '))
lTip = float(input('Enter lunch tip:' ))
lTotal = totalWithTax(lunch, lTip)
print('Lunch total is', lTotal)

dinner= float(input('Enter dinner total: '))
dTip = float(input('Enter dinner tip:' ))
dTotal = totalWithTax(dinner, dTip)
print('Dinner total is', dTotal)
```

Formal Parameters

Actual Parameters

- When called, the actual parameter values are copied to the formal parameters.
- All the commands inside the function are performed on the copies.
- The actual parameters do not change.
- The copies are discarded when the function is done.

Input Parameters

```
def totalWithTax(food,tip):
    total = 0
    tax = 0.0875
    total = food + food * tax
    total = total + tip
    return(total)

lunch = float(input('Enter lunch total: '))
lTip = float(input('Enter lunch tip:' ))
lTotal = totalWithTax(lunch, lTip)
print('Lunch total is', lTotal)

dinner= float(input('Enter dinner total: '))
dTip = float(input('Enter dinner tip:' ))
dTotal = totalWithTax(dinner, dTip)
print('Dinner total is', dTotal)
```

Formal Parameters

Actual Parameters

- When called, the actual parameter values are copied to the formal parameters.
- All the commands inside the function are performed on the copies.
- The actual parameters do not change.
- The copies are discarded when the function is done.
- The time a variable exists is called its **scope**.

Python Tutor

```
def totalWithTax(food,tip):
    total = 0
    tax = 0.0875
    total = food + food * tax
    total = total + tip
    return(total)

lunch = float(input('Enter lunch total: '))
lTip = float(input('Enter lunch tip:' ))
lTotal = totalWithTax(lunch, lTip)
print('Lunch total is', lTotal)

dinner= float(input('Enter dinner total: '))
dTip = float(input('Enter dinner tip:' ))
dTotal = totalWithTax(dinner, dTip)
print('Dinner total is', dTotal)
```

(Demo with pythonTutor)

Input Parameters: What about Lists?

```
#Fall 2013 Final Exam, 5

def kuwae( inLst ):
    tot = 1
    for item in inLst:
        tot = tot * item
    return tot

def foo( inLst ):
    if ( inLst[-1] > inLst[0] ):
        return kuwae( inLst )
    else:
        return -1

foo( [2, 4, 6, 8] )

foo( [4002, 328, 457, 1] )
```

- When called, the actual parameter values are copied to the formal parameters.

Input Parameters: What about Lists?

```
#Fall 2013 Final Exam, 5

def kuwae( inLst ):
    tot = 1
    for item in inLst:
        tot = tot * item
    return tot

def foo( inLst ):
    if ( inLst[-1] > inLst[0] ):
        return kuwae( inLst )
    else:
        return -1

foo( [2, 4, 6, 8] )

foo( [4002, 328, 457, 1] )
```

- When called, the actual parameter values are copied to the formal parameters.
- What is copied with a list?

Input Parameters: What about Lists?

```
#Fall 2013 Final Exam, 5

def kuwae( inLst ):
    tot = 1
    for item in inLst:
        tot = tot * item
    return tot

def foo( inLst ):
    if ( inLst[-1] > inLst[0] ):
        return kuwae( inLst )
    else:
        return -1

foo( [2, 4, 6, 8] )

foo( [4002, 328, 457, 1] )
```

- When called, the actual parameter values are copied to the formal parameters.
- What is copied with a list?
- The address of the list, but not the individual elements.

Input Parameters: What about Lists?

```
#Fall 2013 Final Exam, 5

def kuwae( inLst ):
    tot = 1
    for item in inLst:
        tot = tot * item
    return tot

def foo( inLst ):
    if ( inLst[-1] > inLst[0] ):
        return kuwae( inLst )
    else:
        return -1

foo( [2, 4, 6, 8] )

foo( [4002, 328, 457, 1] )
```

- When called, the actual parameter values are copied to the formal parameters.
- What is copied with a list?
- The address of the list, but not the individual elements.
- The actual parameters do not change, but the inside elements might.

Input Parameters: What about Lists?

```
#Fall 2013 Final Exam, 5

def kuwae( inLst ):
    tot = 1
    for item in inLst:
        tot = tot * item
    return tot

def foo( inLst ):
    if ( inLst[-1] > inLst[0] ):
        return kuwae( inLst )
    else:
        return -1

foo( [2, 4, 6, 8] )

foo( [4002, 328, 457, 1] )
```

- When called, the actual parameter values are copied to the formal parameters.
- What is copied with a list?
- The address of the list, but not the individual elements.
- The actual parameters do not change, but the inside elements might.
- Easier to see with a demo.

Python Tutor

```
#Fall 2013 Final Exam, 5

def kuwae( inLst ):
    tot = 1
    for item in inLst:
        tot = tot * item
    return tot

def foo( inLst ):
    if ( inLst[-1] > inLst[0] ):
        return kuwae( inLst )
    else:
        return -1

foo( [2, 4, 6, 8] )

foo( [4002, 328, 457, 1] )
```

(Demo with pythonTutor)

In Pairs or Triples:

```
def bar(n):
    if n <= 8:
        return 1
    else:
        return 0

def foo(l):
    n = bar(l[-1])
    return l[n]
```

- What are the formal parameters for the functions?
- What is the output of:

```
r = foo([1,2,3,4])
print("Return:  ", r)
```

- What is the output of:

```
r = foo([1024,512,256,128])
print("Return:  ", r)
```

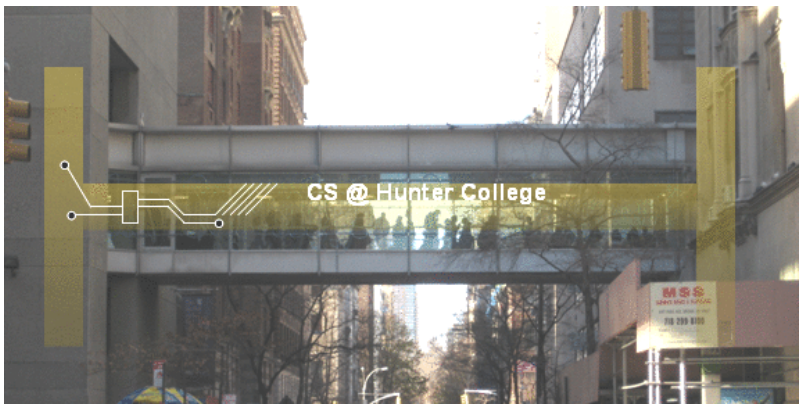
Python Tutor

```
def bar(n):
    if n <= 8:
        return 1
    else:
        return 0

def foo(l):
    n = bar(l[-1])
    return l[n]
```

(Demo with pythonTutor)

CS Survey Talk



Prof. Jia Xu
(machine translation)

Github

- Like Google docs for code...

Octocat

Github

Octocat

- Like Google docs for code...
- Used to share code, documents, etc.

Github

Octocat

- Like Google docs for code...
- Used to share code, documents, etc.
- More formally: `git` is a version control protocol for tracking changes and versions of documents.

Github

Octocat

- Like Google docs for code...
- Used to share code, documents, etc.
- More formally: `git` is a version control protocol for tracking changes and versions of documents.
- Github is a service that provides hosting for repositories ('repos') of code.

Github

Octocat

- Like Google docs for code...
- Used to share code, documents, etc.
- More formally: `git` is a version control protocol for tracking changes and versions of documents.
- Github is a service that provides hosting for repositories ('repos') of code.
- Also convenient place to host websites (i.e. `stjohn.github.io`).

Github

Octocat

- Like Google docs for code...
- Used to share code, documents, etc.
- More formally: `git` is a version control protocol for tracking changes and versions of documents.
- Github is a service that provides hosting for repositories ('repos') of code.
- Also convenient place to host websites (i.e. `stjohn.github.io`).
- In lab, we will set up github accounts and copy ('clone') documents from the class repo. (More in future courses.)

In Pairs or Triples:

```
def prob4(amy, beth):
    if amy > 4:
        print("Easy case")
        kate = -1
    else:
        print("Complex case")
        kate = helper(amy,beth)
    return(kate)
```

```
def helper(meg,jo):
    s = ""
    for j in range(meg):
        print(j, ": ", jo[j])
        if j % 2 == 0:
            s = s + jo[j]
            print("Building s:", s)
    return(s)
```

- What are the formal parameters for the functions?
- What is the output of:

```
r = prob4(4,"city")
print("Return:  ", r)
```

- What is the output of:

```
r = prob4(4,"university")
print("Return:  ", r)
```

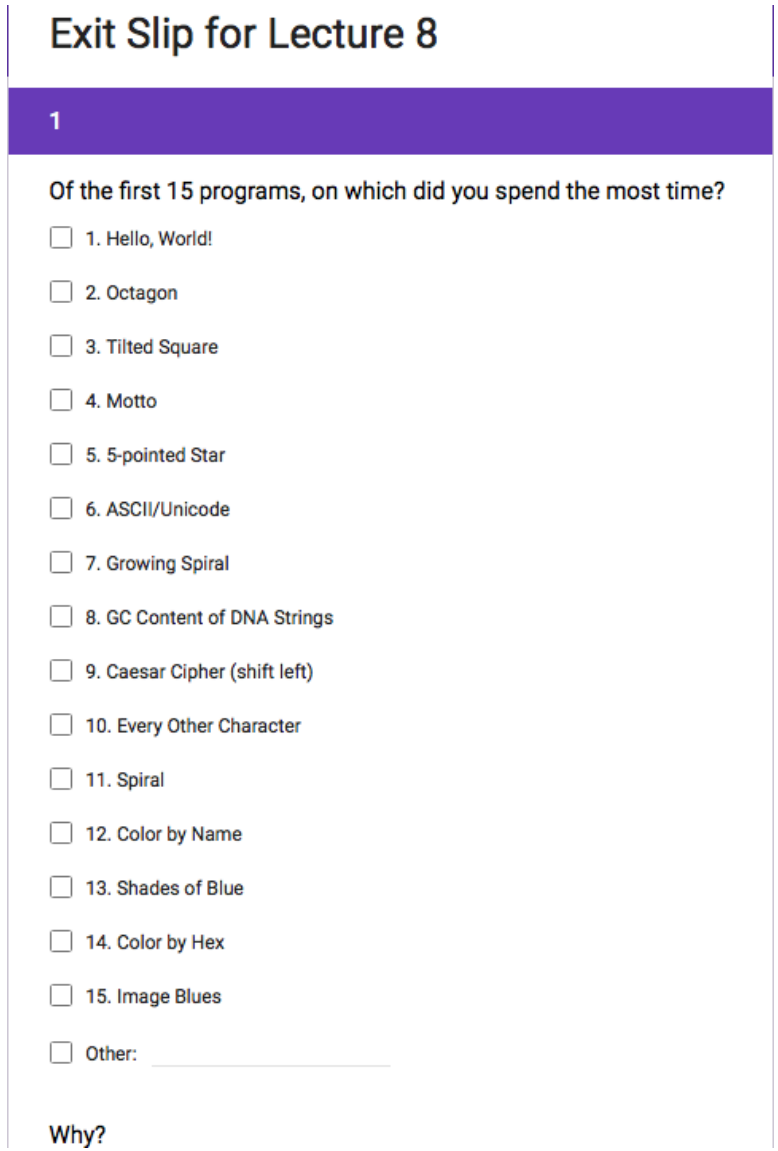
Python Tutor

```
def prob4(amy, beth):
     if amy > 4:
          print("Easy case")
          kate = -1
     else:
          print("Complex case")
          kate = helper(amy,beth)
     return(kate)
```

```
def helper(meg,jo):
     s = ""
     for j in range(meg):
          print(j, ": ", jo[j])
          if j % 2 == 0:
               s = s + jo[j]
               print("Building s:", s)
     return(s)
```

(Demo with pythonTutor)

Lecture Slips: `tinyurl.com/y84y38tq`



Exit Slip for Lecture 8

1

Of the first 15 programs, on which did you spend the most time?

- [ ] 1. Hello, World!
- [ ] 2. Octagon
- [ ] 3. Tilted Square
- [ ] 4. Motto
- [ ] 5. 5-pointed Star
- [ ] 6. ASCII/Unicode
- [ ] 7. Growing Spiral
- [ ] 8. GC Content of DNA Strings
- [ ] 9. Caesar Cipher (shift left)
- [ ] 10. Every Other Character
- [ ] 11. Spiral
- [ ] 12. Color by Name
- [ ] 13. Shades of Blue
- [ ] 14. Color by Hex
- [ ] 15. Image Blues
- [ ] Other: _____

Why?

Recap: Functions

- Functions are a way to break code into pieces, that can be easily reused.

```
#Name:  your name here
#Date:  October 2017
#This program, uses functions,
#     says hello to the world!

def main():
     print("Hello, World!")

if __name__ == "__main__":
     main()
```

Recap: Functions

```
#Name:  your name here
#Date:  October 2017
#This program, uses functions,
#     says hello to the world!

def main():
     print("Hello, World!")

if __name__ == "__main__":
     main()
```

- Functions are a way to break code into pieces, that can be easily reused.
- You **call** or **invoke** a function by typing its name, followed by any inputs, surrounded by parenthesis:

Recap: Functions

```
#Name:  your name here
#Date:  October 2017
#This program, uses functions,
#     says hello to the world!

def main():
     print("Hello, World!")

if __name__ == "__main__":
     main()
```

- Functions are a way to break code into pieces, that can be easily reused.
- You **call** or **invoke** a function by typing its name, followed by any inputs, surrounded by parenthesis: Example: `print("Hello", "World")`

Recap: Functions

```
#Name:  your name here
#Date:  October 2017
#This program, uses functions,
#     says hello to the world!

def main():
     print("Hello, World!")

if __name__ == "__main__":
     main()
```

- Functions are a way to break code into pieces, that can be easily reused.
- You **call** or **invoke** a function by typing its name, followed by any inputs, surrounded by parenthesis: Example: `print("Hello", "World")`
- Can write, or **define** your own functions,

Recap: Functions

```
#Name:  your name here
#Date:  October 2017
#This program, uses functions,
#     says hello to the world!

def main():
     print("Hello, World!")

if __name__ == "__main__":
     main()
```

- Functions are a way to break code into pieces, that can be easily reused.
- You **call** or **invoke** a function by typing its name, followed by any inputs, surrounded by parenthesis: Example: `print("Hello", "World")`
- Can write, or **define** your own functions, which are stored, until invoked or called.

Recap: Functions

```
#Name:  your name here
#Date:  October 2017
#This program, uses functions,
#     says hello to the world!

def main():
     print("Hello, World!")

if __name__ == "__main__":
     main()
```

- Functions are a way to break code into pieces, that can be easily reused.
- You **call** or **invoke** a function by typing its name, followed by any inputs, surrounded by parenthesis: Example: `print("Hello", "World")`
- Can write, or **define** your own functions, which are stored, until invoked or called.
- Functions can have **input parameters** that bring information into the function,

Recap: Functions

```
#Name:  your name here
#Date:  October 2017
#This program, uses functions,
#     says hello to the world!

def main():
     print("Hello, World!")

if __name__ == "__main__":
     main()
```

- Functions are a way to break code into pieces, that can be easily reused.
- You **call** or **invoke** a function by typing its name, followed by any inputs, surrounded by parenthesis: Example: `print("Hello", "World")`
- Can write, or **define** your own functions, which are stored, until invoked or called.
- Functions can have **input parameters** that bring information into the function,
- and **return values** that send information back.
- Both input parameters and return values are optional.