15-112
Lecture 2

Loops

Instructor: Pat Virtue

Tuesday Logistics

As you walk in

Quiz will start at the beginning of lecture

- Have pencil/pen ready
- Don’t use your own scratch paper
  - We have some if you need it
- Silence phones

Quiz

Before we start

- Don’t open until we start
- Make sure your name and Andrew ID are on the front
- Read instruction page
- No questions (unless clarification on English)

Additional info

- 25 min

Announcements

Quiz

Grades

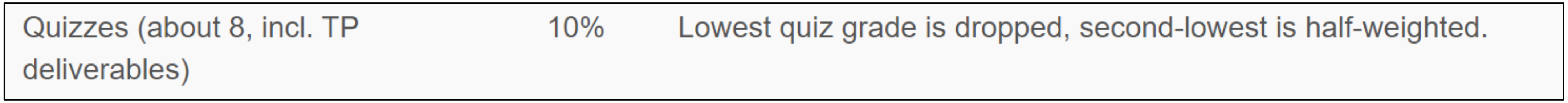
- Likely ready Wednesday
  - Superhero TAs!
- Very small impact on final grade

Fix-its!

- More information coming on Piazza

W

From Syllabus



Quizzes (about 8, incl. TP deliverables)	10%	Lowest quiz grade is dropped, second-lowest is half-weighted.

Announcements

Weekly Rhythm Assignments/Quizzes

- Today, HW2 released ← W
- Thu, Pre-reading 3 released ← F
- Sat, 8 pm: HW 2
- Mon, 8 pm: Pre-reading 3
- Next Tue, in-lec: Quiz 2

Thursday Logistics

Announcements

Quiz

- Review quiz results in Gradescope!
- ✓ Watch solution session recording if you missed the live zoom session
- Regrade requests
  - See Piazza for details
- Fix-its!
  - See Piazza for details

Canvas

- Work in progress: we’re getting scripts setup to sync Canvas Grades

Announcements

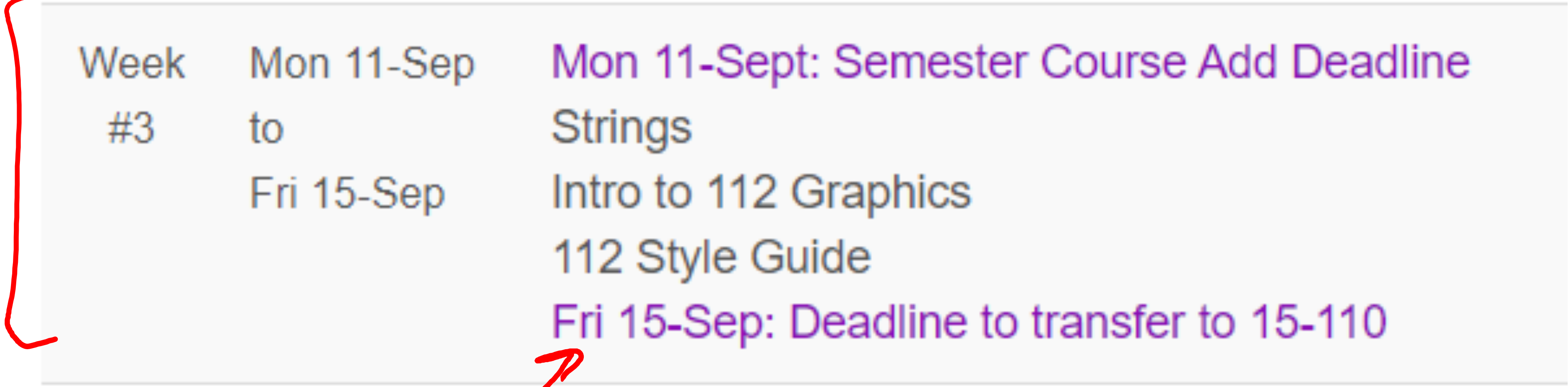
Weekly Rhythm Assignments/Quizzes

- Today, Pre-reading 3 released soon
- Fri: Fix-its due
- Sat, 8 pm: HW 2
- Mon, 8 pm: Pre-reading 3
- Next Tue, in-lec: Quiz 2

Announcements

Registration deadlines next week

From Schedule on course website



Week #3	Mon 11-Sep to Fri 15-Sep	Mon 11-Sept: Semester Course Add Deadline Strings Intro to 112 Graphics 112 Style Guide Fri 15-Sep: Deadline to transfer to 15-110

Loops

Poll 1

What does this code print?

```
for yGrid in range(-2, 2):
    ~~pixel = '+'~~
  #print('+', end=" ")
```

print(yGrid) -2 -1 0 1

A)
```
+ +
```

[✓ B)]
```
+ + + +
```

C)
```
+ + + + +
```

D)
```
+
+
```

E)
```
+
+
+
+
```

F)
```
+
+
+
+
+
```

G) I have no idea

Poll 2

What does this code print?

```
def printPlot(xMin, xMax, yMin, yMax):
    for yGrid in range(yMin, yMax+1):
        for xGrid in range(xMin, xMax+1):
            pixel = '+'
            print(pixel, end=" ")
        print()

printPlot(-3, 3, -2, 2)
```

#

range(-3, 4) 6

range(-2,3)

5

A)

```
+ + + + +
+ + + + +
+ + + + +
+ + + + +
+ + + + +
+ + + + +
+ + + + +
```

7

7

[✓ B)]

5

```
+ + + + + + +
+ + + + + + +
+ + + + + + +
+ + + + + + +
+ + + + + + +
```

C)

```
+ + + +
+ + + +
+ + + +
+ + + +
+ + + +
+ + + +
```

4

6

D)

4

```
+ + + + + +
+ + + + + +
+ + + + + +
+ + + + + +
```

E) I have no idea

Poll 3 (unused)

What does this code print?

```
def printPlot(xMin, xMax, yMin, yMax):
    for yGrid in range(yMin, yMax+1):
        for xGrid in range(xMin, xMax+1):
            if xGrid == 0 and yGrid == 0:
                pixel = '+'
            elif xGrid == 0:
                pixel = '|'
            elif yGrid == 0:
                pixel = '-'
            else:
                pixel = '.'
            print(pixel, end=" ")
        print()
printPlot(-3, 3, -2, 2)
```

A)

```
. . | . .
. . | . .
. . | . .
- - + - -
. . | . .
. . | . .
. . | . .
```

B)

```
. . . | . . .
. . . | . . .
- - - + - - -
. . . | . . .
. . . | . . .
```

C)

```
. . - . .
. . - . .
. . - . .
| | + | |
. . - . .
. . - . .
. . - . .
```

D)

```
. . . - . . .
. . . - . . .
| | | + | | |
. . . - . . .
. . . - . . .
```

E) I have no idea

Poll 3 (unused)

What does this code print?

```
def printPlot(xMin, xMax, yMin, yMax):
    for yGrid in range(yMin, yMax+1):
        for xGrid in range(xMin, xMax+1):
            if xGrid == 0 and yGrid == 0:
                pixel = '+'
            elif xGrid == 0:
                pixel = '|'
            elif yGrid == 0:
                pixel = '-'
            else:
                pixel = '.'
            print(pixel, end=" ")
        print()
printPlot(-3, 3, -2, 2)
```

A)

```
. . | . .
. . | . .
. . | . .
- - + - -
. . | . .
. . | . .
. . | . .
```

B)

```
. . . | . . .
. . . | . . .
- - - + - - -
. . . | . . .
. . . | . . .
```

C)

```
. . - . .
. . - . .
. . - . .
| | + | |
. . - . .
. . - . .
. . - . .
```

D)

```
. . . - . . .
. . . - . . .
| | | + | | |
. . . - . . .
. . . - . . .
```

E) I have no idea

Poll 4

Which code is better

[✓ A)]

```
def sumFromMToN(m, n):
    total = 0
    for x in range(m, n+1):
        total += x
    return total
```

, 1

B)

```
def sumFromMToN(m, n):
    total = 0
    x = m
    while x <= n:
        total += x
        x += 1
    return total
```

For Loops vs While Loops

Often, we can write our code using either

How do we choose

- ✓ For loops are often easier to reason about, especially if were looping over a known sequence
- ✓ While loops work well when we don’t know how many loops we need to do
- Easier to make mistakes with while loops
  - “Help! I run my code, but it doesn’t do anything!!”
  - Infinite loop!!

    Tip: Use ctrl-C to interrupt program execution in the console

    Tip: Include some print statements to see the loop in action

While Loops

Pick a number between 0 and 1000 (Unknown number of loops)

```
guessStr = input("Enter new guess: ")
guess = int(guessStr)
numAttempts = 1

while guess != secret:
    if guess > secret:
        print("--- Too high!")
    else:
        print("--- Too low!")
    guessStr = input(("Enter new guess: ")
    guess = int(guessStr)
    numAttempts += 1
print(f"You got it in {numAttemps}! The secret number was {secret}!")
```

Poll 5 (unused)

How many factors does the numer 16 have?

A. 2

B. 3

C. 4

[✓ D. 5]

E. 6

F. 7

G. 8

H. 16

1 2 4 8 16

Poll 6 (unused)

What is the n-th prime number when n=3?

isPrime(2) ?

0 A. 2

1 B. 3

C. 4

2 D. 5 guess

x E. 6

3 [✓ F.] 7

G. 8

H. 9

I. 10

J. 11

numFound = ~~0~~
~~1~~
~~2~~
3

(so, in 112, we're looking for n+1 things)

Pattern: Find the n-th thing

Find the n-th dino

Pattern: Find the n-th thing

Need

- A way to get to the next guess
- A way to check it: isThing(guess)

Sketch

```
Loop from guess to guess until you've found n (well actually n+1) things
    if isThing(guess):
        numFound += 1
```

Pattern: Find the n-th thing

Find the n-th prime

- NEED: isPrime(number)

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

Design: isPrime(n)

Use paper (or equivalent) to design your solutions!

Design: isPrime(n)

Then you can compare your code your paper examples

```
def isPrime(n):
    if n < 2:
        return False
    for factor in range(2, n):
        if n % factor == 0:
            return False
    return True
```

Pattern: Find the n-th thing

Find the n-th prime

- Assume we have isPrime(number)

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

Pattern: Find the n-th thing

Find the n-th prime

- Assume we have isPrime(number)

```python
def nthPrime(n):
    numFound = 0
    guess = 0  # First guess - 1
    while numFound <= n: # Note: Does one more loop when numFound == n !!
        guess += 1 # Next guess
        if isPrime(guess):
            numFound += 1
    return guess
```

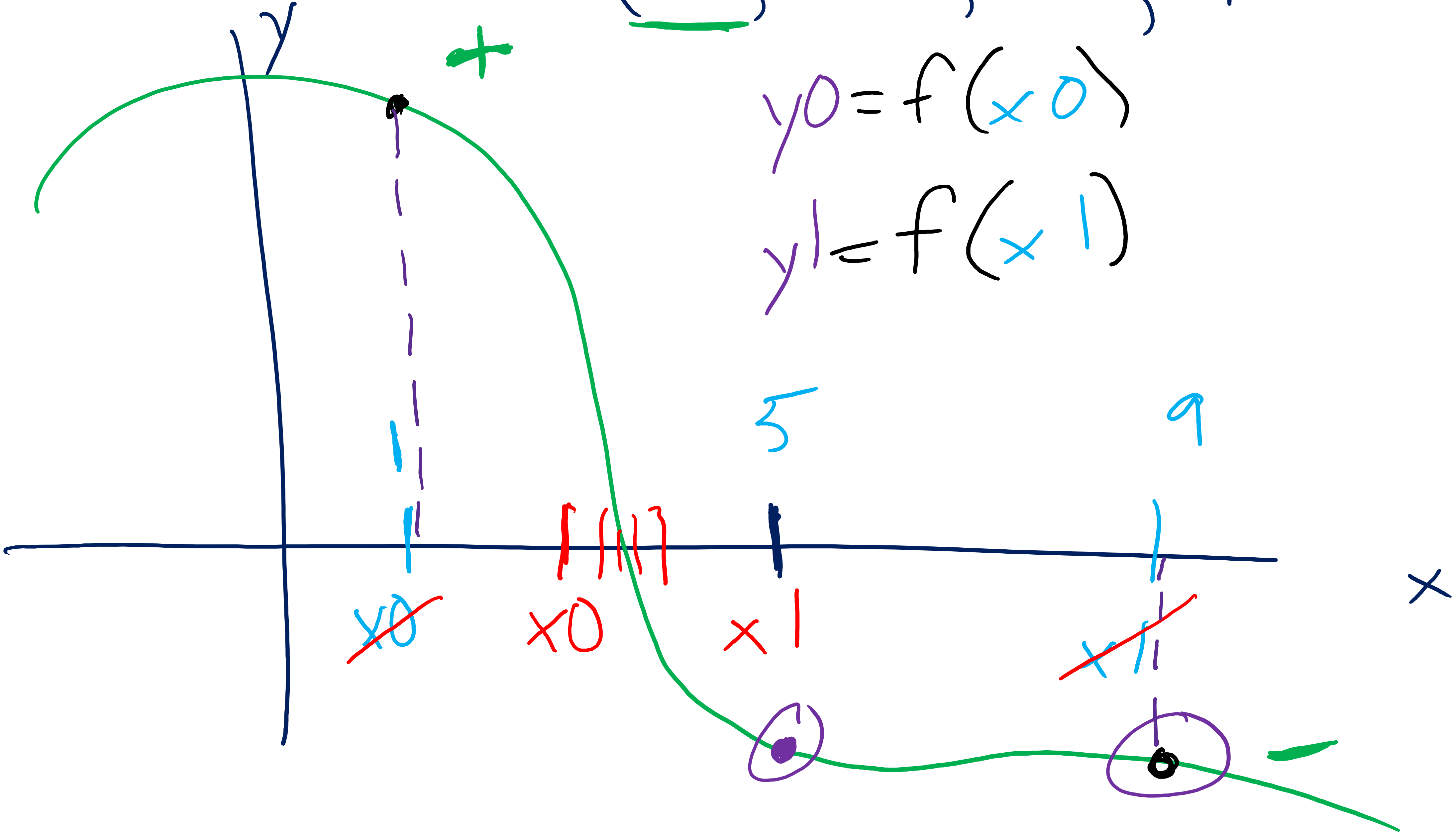
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

Bisection

findZeroWithBisection (f, x0, x1, eps)

y0=f(x0)

y1=f(x1)



Loops: Break and Continue

Poll 7

Which of these prints more lines?

A)

6 7 8 9

```
x = 0
while True:
    x += 1
    if x % 10 == 0:
        break
    print(x)
print('Done')
```

[✓ B)]

6 7 8 9 10 11

```
x = 0
while True:
    x += 1
    if x % 10 == 0:
        continue
    print(x)
print('Done')
```

C) Same

D) I have no idea

Previous Poll 2

What does this code print?

```
def printPlot(xMin, xMax, yMin, yMax):
    for yGrid in range(yMin, yMax+1):
        for xGrid in range(xMin, xMax+1):
            pixel = '+'
            print(pixel, end=" ")
        print()

printPlot(-3, 3, -2, 2)
```

#

range(-3, 4) 6

range(-2, 3)

A) 5 7

```
+ + + + +
+ + + + +
+ + + + +
+ + + + +
+ + + + +
+ + + + +
+ + + + +
```

[✓ B)] 7 5

```
+ + + + + + +
+ + + + + + +
+ + + + + + +
+ + + + + + +
+ + + + + + +
```

C) 4

```
+ + + +
+ + + +
+ + + +
+ + + +
+ + + +
+ + + +
```

D) 6 4

```
+ + + + + +
+ + + + + +
+ + + + + +
+ + + + + +
```

E) I have no idea

Previous Poll 3

What does this code print?

```
def printPlot(xMin, xMax, yMin, yMax):
    for yGrid in range(yMin, yMax+1):
        for xGrid in range(xMin, xMax+1):
            if xGrid == 0 and yGrid == 0:
                pixel = '+'
            elif xGrid == 0:
                pixel = '|'
            elif yGrid == 0:
                pixel = '-'
            else:
                pixel = '.'
            print(pixel, end=" ")
        print()
printPlot(-3, 3, -2, 2)
```

A)

```
. . | . .
. . | . .
. . | . .
- - + - -
. . | . .
. . | . .
. . | . .
```

B)

```
. . . | . . .
. . . | . . .
- - - + - - -
. . . | . . .
. . . | . . .
```

C)

```
. . - . .
. . - . .
. . - . .
| | + | |
. . - . .
. . - . .
. . - . .
```

D)

```
. . . - . . .
. . . - . . .
| | | + | | |
. . . - . . .
. . . - . . .
```

E) I have no idea

Previous Poll 3

What does this code print?

```python
def printPlot(xMin, xMax, yMin, yMax):
    for yGrid in range(yMin, yMax+1):
        for xGrid in range(xMin, xMax+1):
            if xGrid == 0 and yGrid == 0:
                pixel = '+'
            elif xGrid == 0:
                pixel = '|'
            elif yGrid == 0:
                pixel = '-'
            else:
                pixel = '.'
            print(pixel, end=" ")
        print()
printPlot(-3, 3, -2, 2)
```

A)

```
. . | . .
. . | . .
. . | . .
- - + - -
. . | . .
. . | . .
. . | . .
```

B)

```
. . . | . . .
. . . | . . .
- - - + - - -
. . . | . . .
. . . | . . .
```

C)

```
. . - . .
. . - . .
. . - . .
| | + | |
. . - . .
. . - . .
. . - . .
```

D)

```
. . . - . . .
. . . - . . .
| | | + | | |
. . . - . . .
. . . - . . .
```

E) I have no idea

Poll 8

```
def printPlot(xMin, xMax, yMin, yMax):
    for yGrid in range(yMin, yMax+1):
        if yGrid == 0:
            break
        for xGrid in range(xMin, xMax+1):
            if xGrid == 0 and yGrid == 0:
                pixel = '+'
            elif xGrid == 0:
                pixel = '|'
            elif yGrid == 0:
                pixel = '-'
            else:
                pixel = '.'
            print(pixel, end=" ")
        print()
```

-2 -1 0

printRow(xMin xMax, y)

Original for `printPlot(-3, 3, -2, 2)`

```
. . . | . . .
. . . | . . .
- - - + - - -
. . . | . . .
. . . | . . .
```

y=-2

y=-1

The **added code** will result in…?

Select ALL that apply

[✓ A) Fewer rows]

B) Fewer columns

C) A blank row in the center

D) A blank column in the center

E) None of the above

Poll 8

```
def printPlot(xMin, xMax, yMin, yMax):
    for yGrid in range(yMin, yMax+1):
        if yGrid == 0:
            break
        printRow(xMin, xMax, yGrid)
```

-2 -1 0

Original for `printPlot(-3, 3, -2, 2)`

```
.  .  .  |  .  .  .
.  .  .  |  .  .  .
-  -  -  +  -  -  -
.  .  .  |  .  .  .
.  .  .  |  .  .  .
```

y=-2

y=-1

The **added code** will result in...?

Select ALL that apply

[✓ A) Fewer rows]

B) Fewer columns

C) A blank row in the center

D) A blank column in the center

E) None of the above

Poll 9

```python
def printPlot(xMin, xMax, yMin, yMax):
    for yGrid in range(yMin, yMax+1):
        if yGrid == 0:
            continue
        for xGrid in range(xMin, xMax+1):
            if xGrid == 0 and yGrid == 0:
                pixel = '+'
            elif xGrid == 0:
                pixel = '|'
            elif yGrid == 0:
                pixel = '-'
            else:
                pixel = '.'
            print(pixel, end=" ")
        print()
```

-2 -1

printRow(xMin xMax, y)

Original for `printPlot(-3, 3, -2, 2)`

```
.  .  .  |  .  .  .
.  .  .  |  .  .  .
-  -  -  +  -  -  -
.  .  .  |  .  .  .
.  .  .  |  .  .  .
```

The **added code** will result in…?

Select ALL that apply

[✓ A) Fewer rows]

B) Fewer columns

~~C) A blank row in the center~~

D) A blank column in the center

E) None of the above

Poll 9

```
def printPlot(xMin, xMax, yMin, yMax):
    for yGrid in range(yMin, yMax+1):
        if yGrid == 0:
            continue
        printRow(xMin, xMax, yGrid)
```

-2 -1 0 1 2

x

Original for `printPlot(-3, 3, -2, 2)`

```
. . . | . . .
. . . | . . .
- - - + - - -
. . . | . . .
. . . | . . .
```

-2

-1

x

1

2

The **added code** will result in…?

Select ALL that apply

[✓ A) Fewer rows]

B) Fewer columns

~~C) A blank row in the center~~

D) A blank column in the center

E) None of the above

Break and Continue in Nested Loops

Break and continue will only affect their immediate surrounding loop

```
for tensDigit in range(1,6):
    for onesDigit in range(1, 6):
        value = 10*tensDigit + onesDigit
        print(value, end=' ‘)
    print()
```

```
11 12 13 14 15
21 22 23 24 25
31 32 33 34 35
41 42 43 44 45
51 52 53 54 55
```

```
for tensDigit in range(1,6):
    for onesDigit in range(1, 6):
        if onesDigit == 3:
            break
        value = 10*tensDigit + onesDigit
        print(value, end=' ‘)
    print()
```

```
11 12
21 22
31 32
41 42
51 52
```

Break and Continue in Nested Loops

Break and continue will only affect their immediate surrounding loop

```
for tensDigit in range(1,6):
    for onesDigit in range(1, 6):
        value = 10*tensDigit + onesDigit
        print(value, end=' ‘)
    print()
```

```
11 12 13 14 15
21 22 23 24 25
31 32 33 34 35
41 42 43 44 45
51 52 53 54 55
```

```
for tensDigit in range(1,6):
    for onesDigit in range(1, 6):
        if onesDigit == 3:
            continue
        value = 10*tensDigit + onesDigit
        print(value, end=' ‘)
    print()
```

```
11 12 14 15
21 22 24 25
31 32 34 35
41 42 44 45
51 52 54 55
```

Break and Continue in Nested Loops

```
for tensDigit in range(1,6):
    for onesDigit in range(1, 6):
        value = 10*tensDigit + onesDigit
        print(value, end=' ')
    print()
```

```
11 12 13 14 15
21 22 23 24 25
31 32 33 34 35
41 42 43 44 45
51 52 53 54 55
```

```
for tensDigit in range(1,6):
    for onesDigit in range(1, 6):
        if onesDigit == 3:
            break
        value = 10*tensDigit + onesDigit
        print(value, end=' ')
    print()
```

```
11 12
21 22
31 32
41 42
51 52
```

```
for tensDigit in range(1,6):
    for onesDigit in range(1, 6):
        if onesDigit == 3:
            continue
        value = 10*tensDigit + onesDigit
        print(value, end=' ')
    print()
```

```
11 12 14 15
21 22 24 25
31 32 34 35
41 42 44 45
51 52 54 55
```

Break in Nested Loops

```python
def printPlot(xMin, xMax, yMin, yMax):
    for yGrid in range(yMin, yMax+1):
        for xGrid in range(xMin, xMax+1):
            if yGrid == 0:
                break
            if xGrid == 0 and yGrid == 0:
                pixel = '+'
            elif xGrid == 0:
                pixel = '|'
            elif yGrid == 0:
                pixel = '-'
            else:
                pixel = '.'
            print(pixel, end=" ")
        print()
```

Original for `printPlot(-3, 3, -2, 2)`

```
. . . | . . .
. . . | . . .
- - - + - - -
. . . | . . .
. . . | . . .
```

The **added code** will result in…?

Select ALL that apply

A) Fewer rows

B) Fewer columns

C) A blank row in the center

D) A blank column in the center

E) None of the above

Break in Nested Loops

```
def printPlot(xMin, xMax, yMin, yMax):
    for yGrid in range(yMin, yMax+1):
        for xGrid in range(xMin, xMax+1):
            if yGrid == 0:
                break
            if xGrid == 0 and yGrid == 0:
                pixel = '+'
            elif xGrid == 0:
                pixel = '|'
            elif yGrid == 0:
                pixel = '-'
            else:
                pixel = '.'
            print(pixel, end=" ")
        print()
```

printPixel(xGrid, yGrid)

Original for printPlot(-3, 3, -2, 2)

```
. . . | . . .
. . . | . . .
- - - + - - -
. . . | . . .
. . . | . . .
```

The **added code** will result in…?

Select ALL that apply

A) Fewer rows

B) Fewer columns

C) A blank row in the center

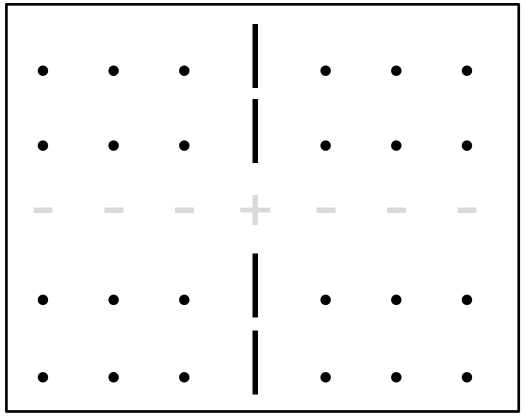
D) A blank column in the center

E) None of the above

Break in Nested Loops

```python
def printPlot(xMin, xMax, yMin, yMax):
    for yGrid in range(yMin, yMax+1):
        for xGrid in range(xMin, xMax+1):
            if yGrid == 0:
                break
            printPixel(xGrid, yGrid)
        print()
```

Original for `printPlot(-3, 3, -2, 2)`



```
.  .  .  |  .  .  .      yGrid: -2
.  .  .  |  .  .  .      yGrid: -1
-  -  -  +  -  -  -      yGrid:  0
.  .  .  |  .  .  .      yGrid:  1
.  .  .  |  .  .  .      yGrid:  2
```

The **added code** will result in…?

Select ALL that apply

A) Fewer rows

B) Fewer columns

[✓ C)] A blank row in the center

D) A blank column in the center

E) None of the above

Design: Patterns and Top-Down Design

Pattern: Find the n-th thing

Find the n-th prime

- More than one way to write it

```
def nthPrime(n):
  numFound = 0
  guess = 0  # First guess - 1
  while numFound <= n:
    guess += 1  # Next guess
    if isPrime(guess):
      numFound += 1
  return guess
```

```
def nthPrime(n):
  numFound = 0
  guess = 1  # First guess
  while True:
    if isPrime(guess):
      numFound += 1
    if numFound == n+1:
      return guess

    guess += 1  # Next guess
```

Poll 10 (unused)

Which version is better?

A)

```
def nthPrime(n):
  numFound = 0
  guess = 0  # First guess - 1
  while numFound <= n:
    guess += 1  # Next guess
    if isPrime(guess):
      numFound += 1
  return guess
```

B)

```
def nthPrime(n):
  numFound = 0
  guess = 1  # First guess
  while True:
    if isPrime(guess):
      numFound += 1
    if numFound == n+1:
      return guess

    guess += 1  # Next guess
```

Top-down Design

Start coding with a birds-eye-view of the task

As you code, assume you have completed versions of lower level tasks

Example: Find nthDooDad(n):

```
def nthDooDad(n):
  numFound = 0
  guess = 1  # First guess
  while True:
    if ???
```

Top-down Design

Start coding with a birds-eye-view of the task

As you code, assume you have completed versions of lower level tasks

Example: Find nthDooDad(n):

```
def nthDooDad(n):
  numFound = 0
  guess = 1  # First guess
  while True:
    if isDooDad(guess)
      numFound += 1
    if numFound == n+1:
      return guess

    guess += 1  # Next guess
```

```
def isDooDad(g):
    return isDoo(g) and isDad(g)

def isDoo(g):
    pass

def ifDad(g)
    pass
```

n-th Pattern

Need

- A way to get to the next guess
- A way to check it: isThing(guess)

Sketch

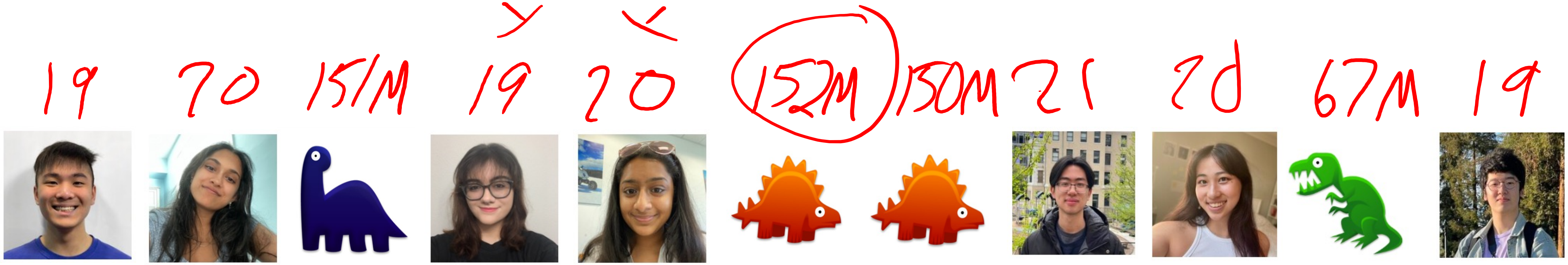
```
Loop from guess to guess until you've found n (well actually n+1) things
        if isThing(guess):
                numFound += 1
```

Best Pattern

Find the “best” thing in some collection

What “best” means depends on the application.

Example: Find the oldest TA



Best Pattern

Need

- Ability to loop through all items
- Ability to compare value of items

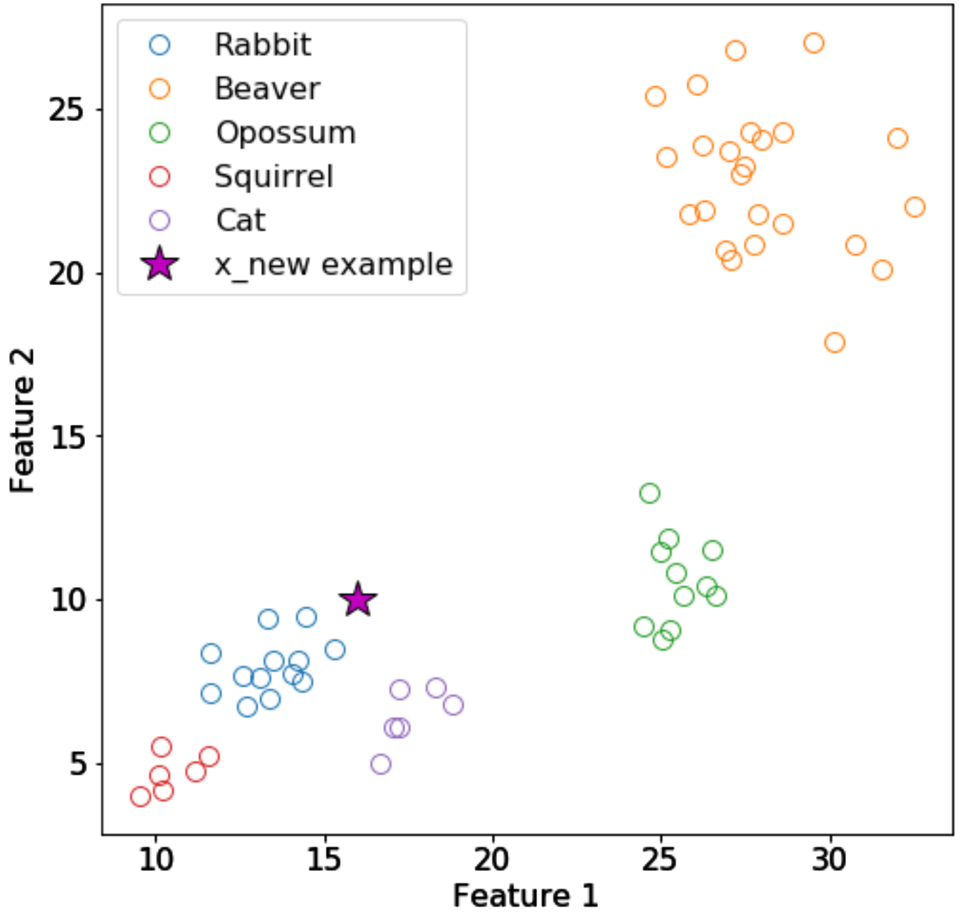
Sketch

```
Initialize bestValue (often some extreme or impossible value, like None)
Loop through all items:
        if item value is “better” than bestValue
                Update bestValue to value of current item
                Note: (Sometimes you also need to keep track of the item itself
                in cases where the item and the value of the item differ)
```

Top-down Design

Example: Best Pattern + Top-down Design

nearestNeighbor(newPoint, trainingPoints):

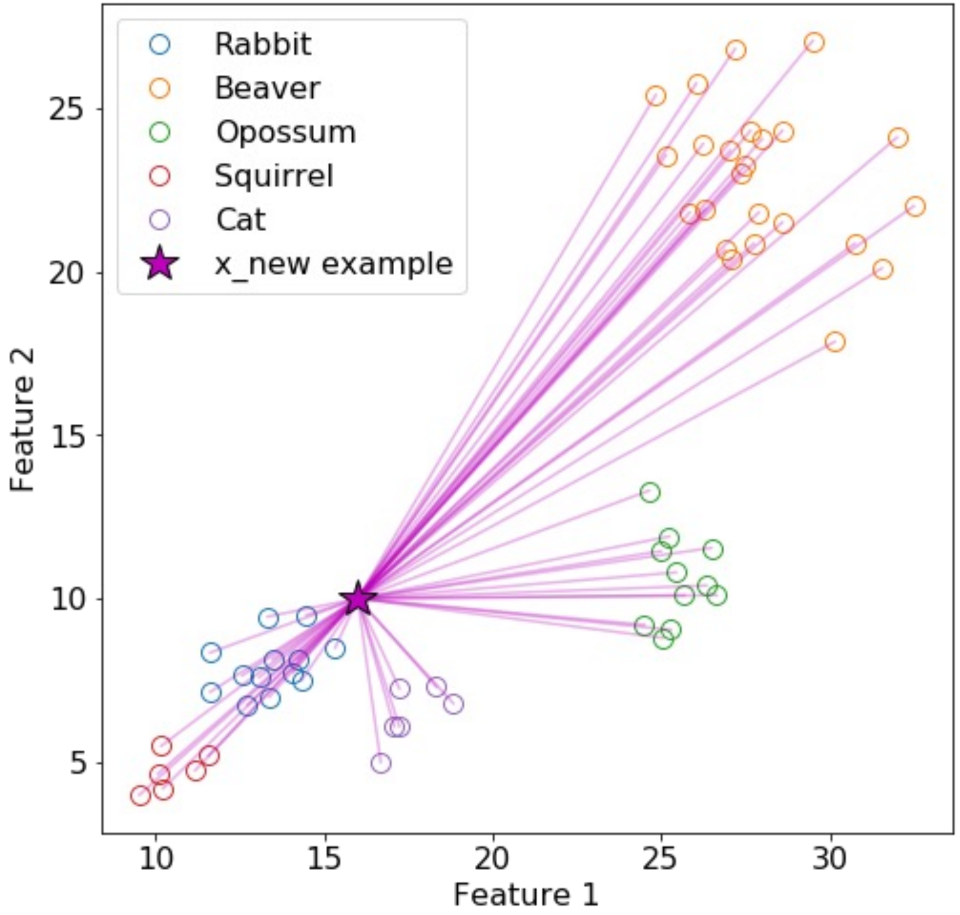


Top-down Design

Example: Best Pattern + Top-down Design

nearestNeighbor(newPoint, trainingPoints):

```
def getNumPoints(points):
    pass

def getPointFromPoints(points, i):
    pass

def distance(point1, point2):
    pass
```



```
def nearestNeighbor(newPoint, trainingPoints):
    bestDistance = math.inf
    bestPoint = None

    numPoints = getNumPoints(trainingPoints)
    for i in range(numPoints):
        trainingPoint = getPoint(trainingPoints, i)
        dist = distance(newPoint, trainingPoint)

        if dist <= bestDistance:
            bestDistance = dist
            bestPoint = trainingPoint

    return bestPoint
```

Style

Poll 11 (unused)

Which code is better?

A)

```
def distance(x1, y1, x2, y2):
    return ((x1-x2)**2 + (y1-y2)**2)**0.5
```

B)

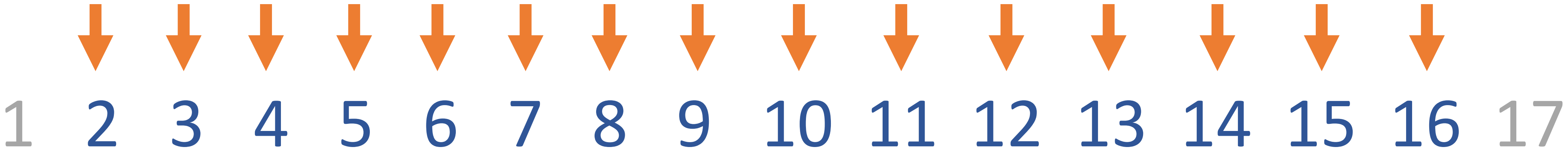
```
def distance(x1, y1, x2, y2):
    xDiff = x1 - x2
    yDiff = y1 - y2

    xDiffSquared = xDiff**2
    yDiffSquared = yDiff**2

    sumOfSquares = xDiffSquared + yDiffSquared

    result = sumOfSquares**0.5
    return result
```

Algorithm Design: Faster isPrime

Algorithm Design: isPrime(n)

This version is actually *really* slow

```python
def isPrime(n):
    if n < 2:
        return False

    for factor in range(2, n):
        if n % factor == 0:
            return False

    return True

isPrime(17)
```



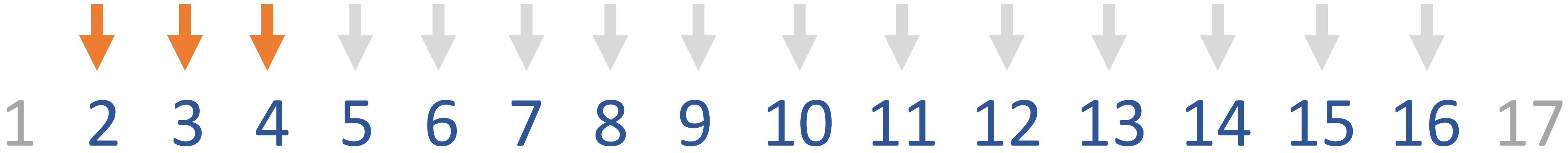
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

Algorithm Design: isPrime(n)

We can do better

```python
def isPrime(n):
    if n < 2:
        return False

    for factor in range(2, n):
        if n % factor == 0:
            return False

    return True

isPrime(17)
```

```python
def fasterIsPrime(n):
    if n < 2:
        return False

    for factor in range(2, int(n**0.5)+1):
        if n % factor == 0:
            return False

    return True

fasterIsPrime(17)
```



1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

Algorithm Design: isPrime(n)

Timing with the time library

```python
import time

startTime = time.time()
result = isPrime(7368791)
endTime = time.time()

elapsedTime = endTime - startTime
print(f'isPrime: {elapsedTime} sec')
```

```
isPrime: 1.1695044040679932 sec
```

```python
import time

startTime = time.time()
result = fasterIsPrime(7368791)
endTime = time.time()

elapsedTime = endTime - startTime
print(f’fasterIsPrime: {elapsedTime} sec')
```

```
fasterIsPrime: 0.0010092258453369141 sec
```

Over 1000x faster for `isPrime(7368791)`