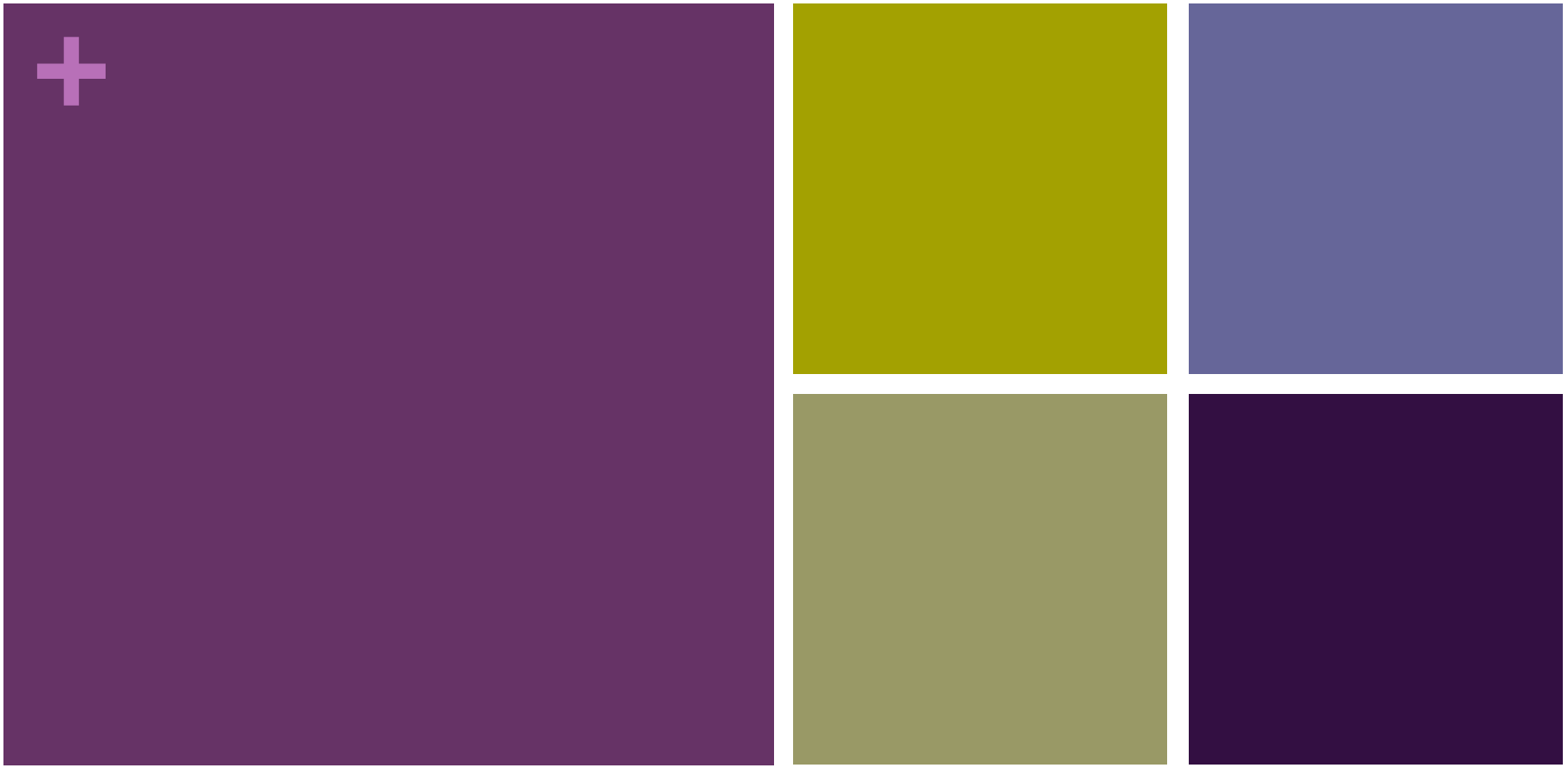
Warm Up

www.pollev.com/python002



Count Controlled Loops

Introduction to Computer Programming - Python

+

While Loop Review

When to use a While loop

- We can use while loops when we need to repeat a task multiple times in order to solve a particular problem. For example:
  - Print the phrase "Hello, World" 100 times
  - Ask the user to enter 10 price values and add them to an accumulator variable
- A while loop works well for tasks that require an unknown number of iterations. For example:
  - Ask the user to enter in a positive number. If the user enters a negative number, re-prompt them until they supply a positive number.

Do we need a While loop to solve this problem?

- Write a program that asks the user for two numbers
- If the first number is greater than the second number, add the numbers and display the total
- If the second number is greater than the first number, multiply the numbers and display the product

```
Enter 1st number: 5

Enter 2nd number: 10

Product:  50
```

Do we need a While loop to solve this problem?

- Write a program that asks the user for two numbers. Only accept positive values
- If the first number is greater than the second number, add the numbers and display the total
- If the second number is greater than the first number, multiply the numbers and display the product

```
Enter 1st number: 5

Enter 2nd number: -10
Sorry, try again
Enter 2nd number: 10

Product:  50
```

Do we need a While loop to solve this problem?

- Ask the user to enter in 5 price values
- Add these values to a total variable
- Print out the total at the end of the program plus 7% sales tax

```
Enter price: 1.00
Enter price: 2.00
Enter price: 3.00
Enter price: 3.00
Enter price: 1.00

Your total:  10.00
Tax:  0.70
Grand total:  10.07
```

Do we need a While loop to solve this problem?

- Ask the user to enter in a potentially ***unlimited*** number of price values
- Add these values to a total variable
- Print out the total at the end of the program plus 7% sales tax

```
Enter price, 0 to end: 1.00
Enter price, 0 to end: 2.00
Enter price, 0 to end: 3.00
Enter price, 0 to end: 3.00
Enter price, 0 to end: 1.00
Enter price, 0 to end: 0

Your total:  10.00
Tax:  0.70
Grand total:  10.07
```

+

Count Controlled Loops

Count Controlled Loops

- A count controlled loop is a repetition structure that iterates a specific number of times
- In contrast, a condition controlled loop iterates a variable number of times – we control the # of iterations through our Boolean condition

Count Controlled Loops

- You can write a count controlled loop using a while() loop. For example:

```
counter = 0
while counter < 5:
    print (“This will print 5 times”)
    counter += 1
```

Count Controlled loops

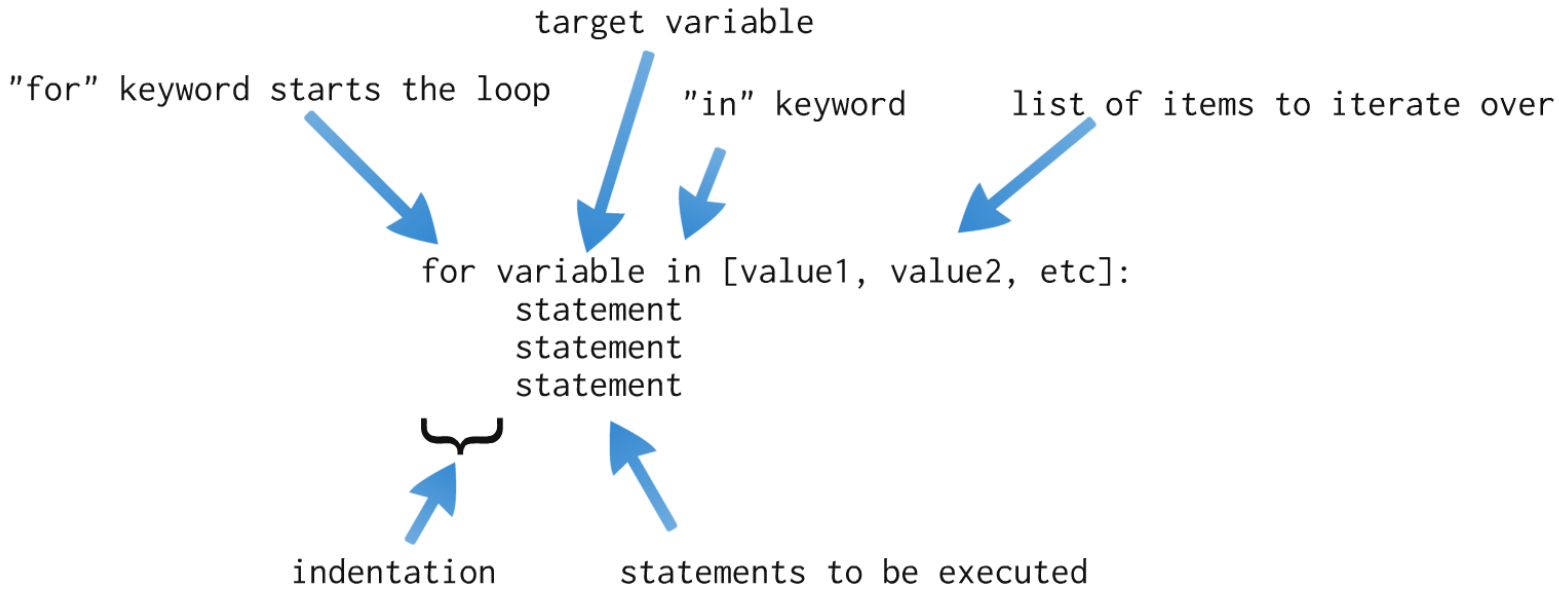
- Python (and all other programming languages) have special structures which can be used to implement a count controlled loop without needing to use a condition controlled loop (though you could always use a condition controlled loop if you wanted to)

The “for” loop

- The “for” loop is Python’s native count controlled loop.
- Example:

```
for num in [1,2,3,4,5]:
    print (“This will print 5 times”)
```

The “for” loop


target variable
"for" keyword starts the loop
"in" keyword
list of items to iterate over
for variable in [value1, value2, etc]:
statement
statement
statement
indentation
statements to be executed

The "for" loop

- The "for" loop will iterate once for each item defined in the list passed to it when the loop begins
- Lists in Python are defined by the square bracket characters "[" and "]". Items in a list are separated by a comma.
- The first time a "for" loop iterates the target variable will assume the value of the first item in the list
- The second time a "for" loop iterates the target variable will assume the value of the second item in the list
- This continues until you reach the end of the list

The “for” loop

```
for c in [1,2,3,4]:
  print (c)
```

```
1

2

3

4
```

The “for” loop

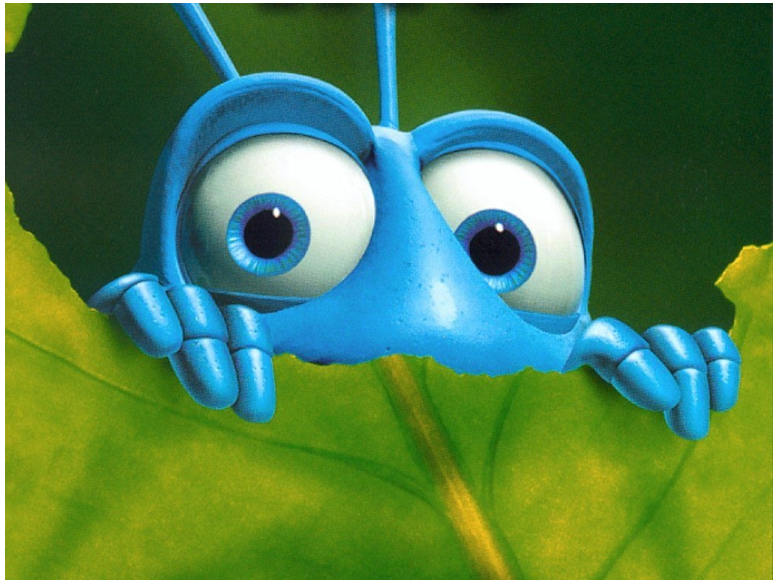
- We will talk more about lists near the end of the semester. With that said, lists can contain collections of different kinds of data. For example:

```
for name in ['Craig', 'John', 'Chris']:

   print ("The current user is:", name)
```

Programming Challenge: The Bug Collector

- A bug collector collects bugs every day for seven days.
- Write a program that keeps a running total of the number of bugs collected during the seven days. The loop you write should ask for the number of bugs collected for each day, and when it finishes it should display the total number of bugs collected.



Programming Challenge: Student Test Scores

- Write a program that iterates over the following student names:
  - John
  - Mary
  - Michael
  - Sophie
- Ask the user to input a test score for each student
- Calculate the average test score for the class

Programming Mechanics Challenge

- Rewrite the following loop as a "for" loop:

```
x = 0

while x < 5:
      print ("hi")
      x += 1
```

Programming Mechanics Challenge

- Rewrite the following loop as a "while" loop:

```
for x in [10,20,30,40]:
     print ("hi")
```

The range() function

The range() function

- So far we have been iterating over lists using pre-defined values in our for() loops
- Example:

```
for x in [1,2,3,4,5]:
    print ('hi')
```

- The range() function lets you dynamically generate lists based on criteria that you define

The range() function

```
for i in range(5):
   print ('iteration #', i)
```

```
iteration # 0

iteration # 1

iteration # 2

iteration # 3

iteration # 4
```

The range() function

- The range() function takes at least one argument. In its simplest form it takes a single integer.
- The range() function returns an "iterable", which is a Python data type similar to a list.
- When passed a single integer the range function will generate an iterable that will cause a for() loop from 0 to the number specified minus one.

The range() function

range() function call	iterable
`range(5)`	`[0,1,2,3,4]`
`range(10)`	`[0,1,2,3,4,5,6,7,8,9]`

The range() function

- You can pass additional parameters to the range() function to cause it to behave differently
- Examples:

```
range(1,5)      # set a start and end value for the range
                # [1,2,3,4]

range(5,10)     # [5,6,7,8,9]

range(0,10,2)   # set a start, end and step (or increment) value
                # [0,2,4,6,8]

range(1,10,2)   # [1,3,5,7,9]
```

Programming Challenge

- Write the range() function that will generate the following sequences:

1. [1,2,3,4,5,6,7,8,9,10]
2. [2,4,6,8,10]
3. [1,3,7,9,11]
4. [-3, -2, -1, 0, 1, 2, 3]
5. [5, 4, 3, 2, 1]

Flow of Execution

For loops

Flow of Execution

Code

```
print ("Welcome!")

for x in range(3):
      print (x)

print ("Goodbye!")
```

Output

Flow of Execution

Code

```
print ("Welcome!")

for x in range(3):
      print (x)

print ("Goodbye!")
```

Output

```
Welcome!
```

Flow of Execution

Code

```
print ("Welcome!")

for x in range(3):
      print (x)

print ("Goodbye!")
```

Output

```
Welcome!
```

Flow of Execution

Code

```
print ("Welcome!")

for x in range(3):
      print (x)

print ("Goodbye!")
```

Output

```
Welcome!
0
```

Flow of Execution

Code

```
print ("Welcome!")

for x in range(3):
      print (x)

print ("Goodbye!")
```

Output

```
Welcome!
0
```

Flow of Execution

Code

```
print ("Welcome!")

for x in range(3):
      print (x)

print ("Goodbye!")
```

Output

```
Welcome!
0
1
```

Flow of Execution

Code

```
print ("Welcome!")

for x in range(3):
      print (x)

print ("Goodbye!")
```

Output

```
Welcome!
0
1
```

Flow of Execution

Code

```
print ("Welcome!")

for x in range(3):
    print (x)

print ("Goodbye!")
```

Output

```
Welcome!
0
1
2
```

Flow of Execution

Code

```
print ("Welcome!")

for x in range(3):
      print (x)

print ("Goodbye!")
```

Output

```
Welcome!
0
1
2
Goodbye!
```

Reverse ranges

Reverse ranges

- The step value passed to the range() function does not necessarily have to be positive.
- If you pass a negative step value to the range() function it will count backwards for you.

Programming Challenge: Blast Off

- Write a countdown program that prompts the user for a max value (i.e. 30)
- Print out a countdown from that number down to zero, then print "blast off!"

Using loop targets

Loop targets

- In a for loop we generally use the target variable as a reference value for some kind of calculation
- Remember that the value of the target variable changes with each iteration of the loop

Programming Challenge: Squares

- Write a program that calculates the square of the numbers between 1 and 10
- Print out the number and its square as your loop iterates

Number	Square
1	1
2	4
3	9
4	16
…	…
10	100

Programming Challenge: Stair Steps

- Write a program that prints out the following pattern of characters:

```
**
****
******
********
**********
************
```

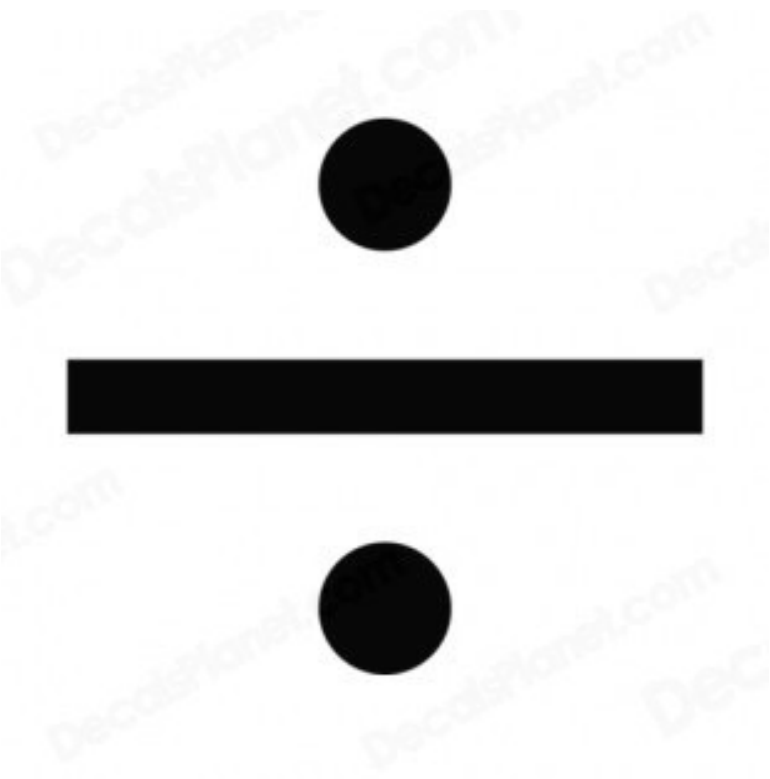
Programming Challenge: Stair Steps

- Write a program that prints out the following pattern of characters:

```
      **
    ****
  ******
********
**********
************
```

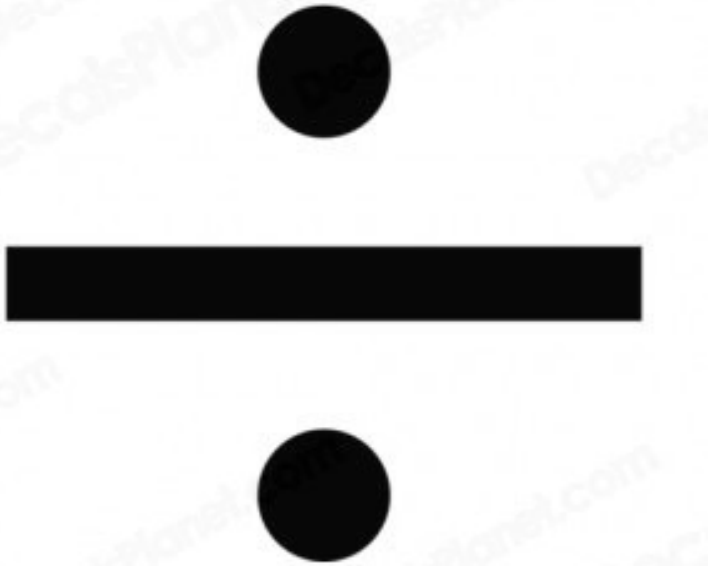
Programming Challenge: Divisibility

- Write a program that asks the user to enter in an integer
- Then find all numbers between 1 and 10,000 that are evenly divisible by that number



Programming Challenge: Divisibility

- Extensions:
  - Extend your program to collect two integers when it starts up
  - Find all numbers in the specified range that are divisible both of the two supplied numbers
  - Extension: print your results such that you print 10 #'s per line



User controlled ranges

User controlled ranges

- In many cases a programmer knows how many iterations he or she needs in order to accomplish a desired task
- However, sometimes we need to ask the user to control the # of iterations within a loop.
- You can easily do this by substituting a variable within the range() function to control the start, end and step values of the iterable that will be generated

Programming Challenge: (imperfect) Lottery # Picker

- Write a program that generates random lottery numbers for the user
- Ask the user for the number of digits they need as well as the high and low value of each digit (i.e. 6 digit number with digits ranging from 1 to 60)
- Generate the desired lottery number

+

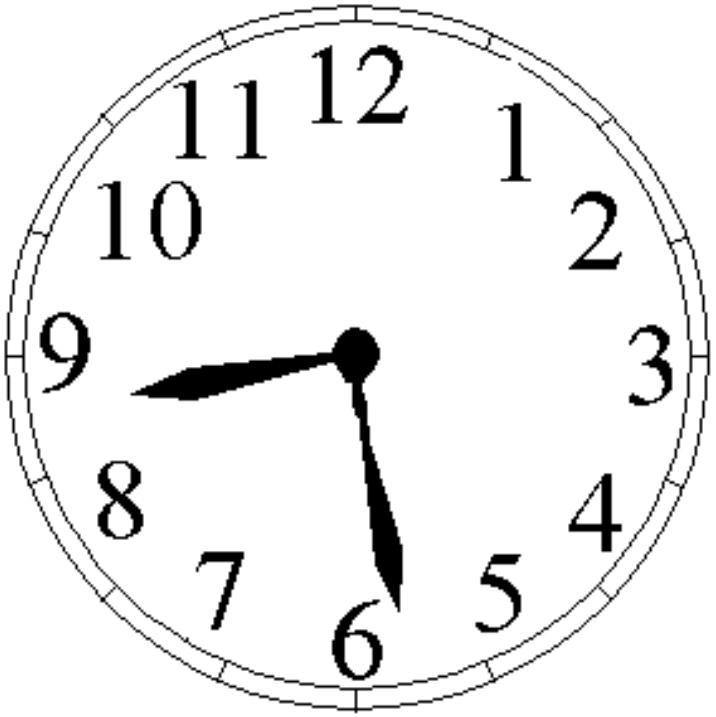
Nested Loops

Nested Loops

- A nested loop can be described as a "loop inside of a loop"
- It's the same idea as nested selection statements ("if" statements inside other "if" statements)

Programming Challenge: Clock Simulator

- Write a program that prints out every possible time value for a single day.
- Print out the hours, minutes and seconds starting at midnight and continue on to 11:59.59 PM.
- Output each value as follows:
  - 0:0:0
  - 0:0:1
  - 0:0:2
  - …
  - 23:59:59



Addition Tables

- Write a program that prints out an Addition table for the number 5. For example:

```
5 plus 1 is 6
5 plus 2 is 7
5 plus 3 is 8
…
5 plus 10 is 15
```

Addition Tables

- Write a program that prints out an Addition table for the numbers 5 and 6. For example:

```
5 plus 1 is 6
5 plus 2 is 7
5 plus 3 is 8
…
5 plus 10 is 15


6 plus 1 is 7
6 plus 2 is 8
6 plus 3 is 9
…
6 plus 10 is 16
```

Addition Tables

- Write a program that prints out the Addition tables for the numbers 1 through 10
- Extend your program to allow the user to type in a range of Addition tables they want printed. Example:

```
Addition Table Generator 2000!
Enter the first number in your range:  1
Enter the last number for your table: 20
```

Some notes on Nested Loops

- Some notes on nested loops:
  - The innermost loop will iterate through all its iterations for every single iteration of an outer loop
  - Inner loops complete their iterations faster than outer loops
  - To get the total number of iterations of a nested loop, multiply the number of iterations of all the loops

Programming Challenge

- Reproduce the pattern to the right using nested loops

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 6
```

Programming Challenge: Graphical Pattern

- Write a program that prints the pattern to the left using nested loops

Nested Loops in the Wild: Decorative Arts

- How many repeating and nested patterns can you isolate in the following Ancient Greek designs? (Source)