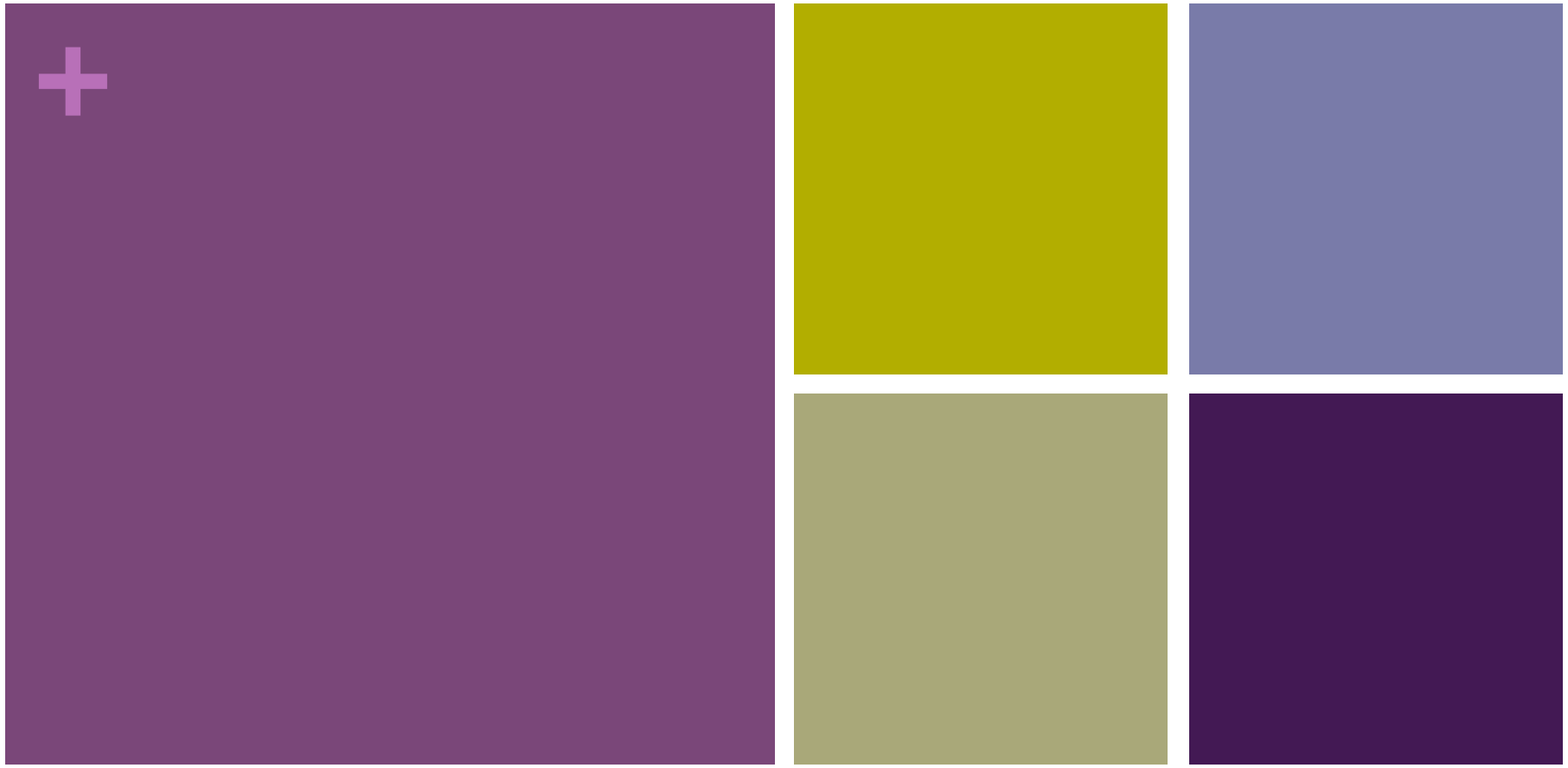
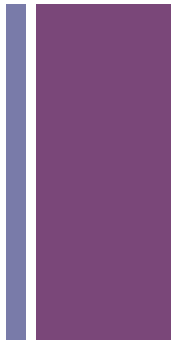


Lecture 06: The List Data Structure

Variables vs. Lists



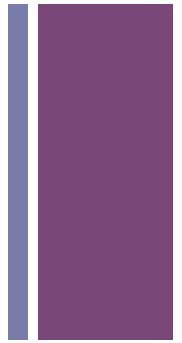
So far we have been working with variables, which can be thought of as "buckets" that hold a particular piece of data

Variables can only hold one piece of data at a time.
Example

```
x   =  5
y   =  5.0
z   =  'hello'
q   =  True
```

However, there are times when we need to keep track of multiple pieces of data at the same time, and a single variable is limited to holding just one thing at a time

Lists



- Lists are considered a "sequence" object. Sequence objects have the ability to hold multiple pieces of data at the same time.

- We can use a single sequence variable to hold any number of values.

- In most programming languages we call these "arrays." In Python we call these "lists."

Lists vs. Variables

List	Variable

Lists in Python

- You can create a list in Python by using bracket notation.

  Example:

```
my_list = [1, 2, 3]
```

- The above code will create a new list in Python that holds three integers – 1, 2 and 3 – in that order.

- Think of a list as a "book" that holds a series of sheets of paper (variables)

Lists in Python

Lists can contain any data type that we have covered so far.

Example:

```
my_list = ['Craig', 'John', 'Chris']
```

Lists can also mix data types. Example:

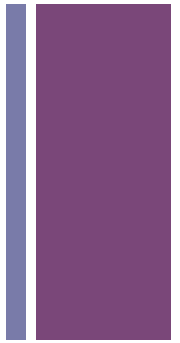
```
my_list = ['Craig', 5.0, True, 67]
```

You can print the value of a list using the print() function.

Example:

```
print (my_list)
```

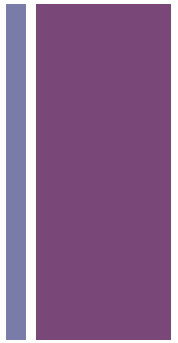
List Repetition



You can use the repetition operation (“*”) to ask Python to repeat a list, much like how you would repeat a string.
Example:

```
my_list = [1, 2, 3] * 3
print (my_list)

>> [1, 2, 3, 1, 2, 3, 1, 2, 3]
```

List Concatenation



You can use the concatenation operation (“+”) to ask Python to combine lists, much like how you would combine strings. Example:

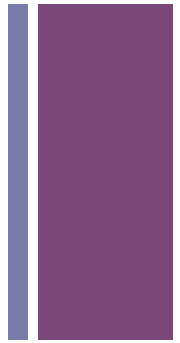
```
my_list = [1, 2, 3] + [99, 100, 101]
print (my_list)

[1, 2, 3, 99, 100, 101]
```

Indexing List Elements

- In a book you can reference a page by its page number
- In a list you can reference an element by its index number
- Indexes start at the number zero.
- Example:

```
my_list = ['Craig', 'John', 'Chris']
print (my_list)
print (my_list[1])
```

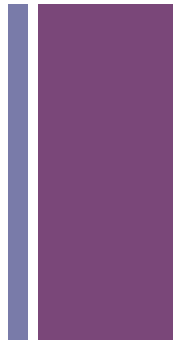
Invalid indexes



You will raise an exception if you attempt to access an element outside the range of a list. For example:

```
my_list = ['Craig', 'John', 'Chris’]

print (my_list[4]) # Index doesn’t

exist!
```

Changing the value of an item in a list



Lists are "mutable," which means that they can be changed once they have been created (unlike strings)

Example:

```
>>>my_list = [1, 2, 3]
>>>print (my_list)
[1,2,3]
>>>my_list[0] = 99
>>>print (my_list)
[99,2,3]
```

List Mechanics

List variables are considered "references"

This means that they "reference" or "point" to a specific region of your computer's memory. This behavior can cause some interesting side effects. For example, the following two list variables refer to the same list in memory.

```
mylist1 = [1,2,3]
mylist2 = mylist1

print (mylist1)
print (mylist2)

>> [1,2,3]
>> [1,2,3]
```

Creating Lists

You can create an empty list with no elements using the following syntax:

```
mylist = []
```

Sometimes you want to create a list that contains a certain number of "pre-set" elements. For example, to create a list with 10 elements that are all set to zero you could do the following:

```
mylist = [0] * 10
```

Iterating over a list

Using a "for" loop to iterate through a List

You can also use a for loop to iterate through a list. When you do this the target variable of your loop assumes each value of each element of the list in order. Example:

```
my_list = [1,2,3]

for number in my_list:
  print (number)
   >>  1
   >>  2
   >>  3
```

Programming Exercise: Count the A's

Given the following list:

```
grades =
[90,100,70,45,76,84,93,21,36,99,100]
```

Write a program that counts the # of A's (scores between 90 and 100)

Extension: Count the # of B's, C's, D's and F's

Exercise

Write a Python program to input the number of month, and print out the English name of the month. If the number is invalid, then print “sorry invalid”

Example

```
 Please input the number of the
 month: 1
January
      :13
‘sorry, wrong input’
```

Exercise 2

Write a Python program to input the number of month June, and print out the English name of the day of the week. Given that June 1 is Thursday. If the number of the day is invalid, then print “sorry invalid”

Please input the day of June: 5
June 5 is Monday.

Please input the day of June: 29
June 29 is Thursday.

Please input the day of June: 35
Invalid input.