Python 101

Lists

Announcements

quiz: `quiz04` due Tues 09/24

lab: `lab02` on Fri 09/27

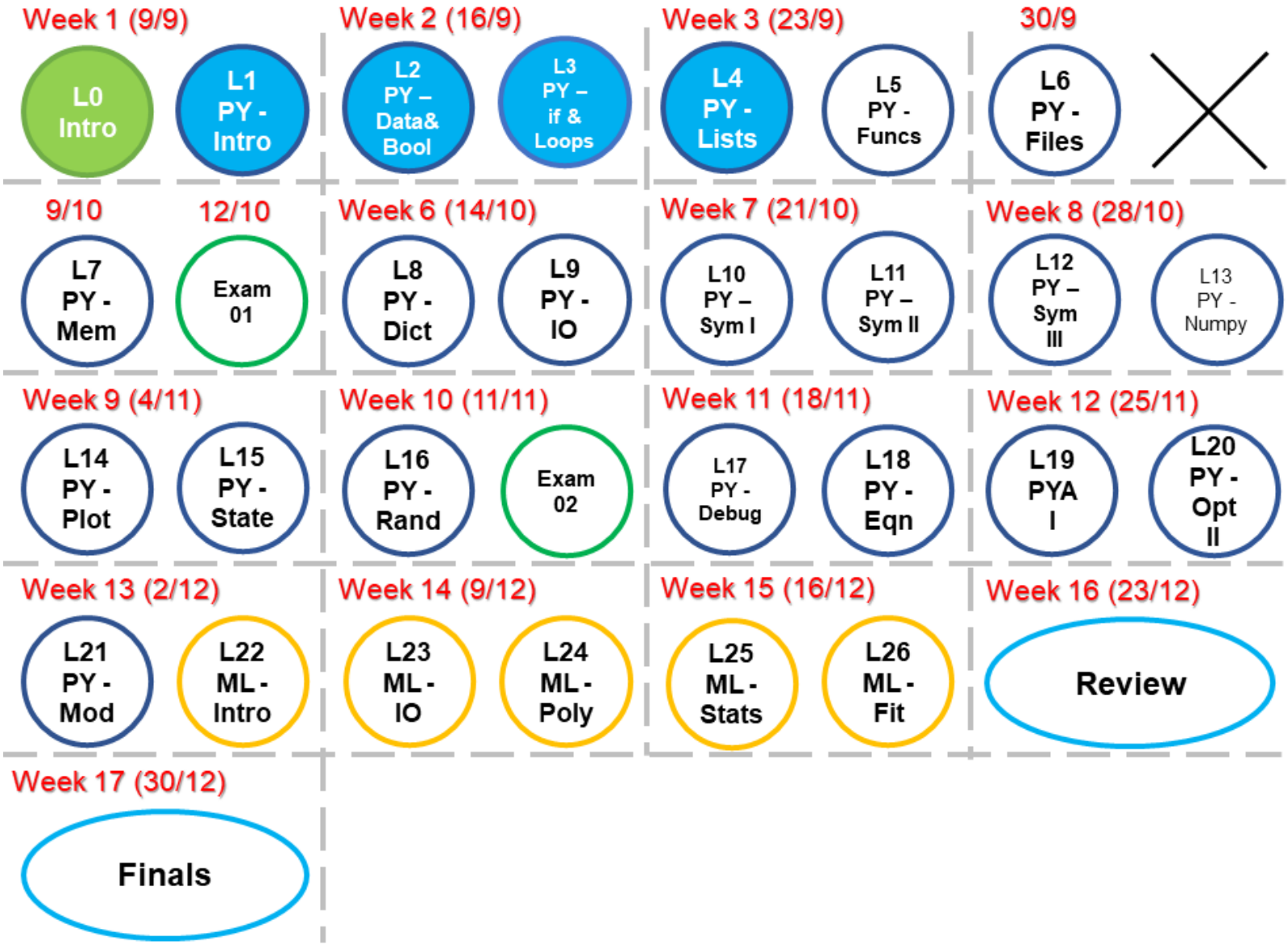
lab: `lab03` on Sun 09/29

hw: `hw02` due today 09/23

I have office hours today! I can save your homework 2!

I see a lot of 0 in your quizzes?!!!@@@

Roadmap


Week 1 (9/9)
L0 Intro
L1 PY - Intro
Week 2 (16/9)
L2 PY – Data& Bool
L3 PY – if & Loops
Week 3 (23/9)
L4 PY - Lists
L5 PY - Funcs
30/9
L6 PY - Files
9/10
12/10
Week 6 (14/10)
Week 7 (21/10)
Week 8 (28/10)
L7 PY - Mem
Exam 01
L8 PY - Dict
L9 PY - IO
L10 PY – Sym I
L11 PY – Sym II
L12 PY – Sym III
L13 PY - Numpy
Week 9 (4/11)
Week 10 (11/11)
Week 11 (18/11)
Week 12 (25/11)
L14 PY - Plot
L15 PY - State
L16 PY - Rand
Exam 02
L17 PY - Debug
L18 PY - Eqn
L19 PYA I
L20 PY - Opt II
Week 13 (2/12)
Week 14 (9/12)
Week 15 (16/12)
Week 16 (23/12)
L21 PY - Mod
L22 ML - Intro
L23 ML - IO
L24 ML - Poly
L25 ML - Stats
L26 ML - Fit
Review
Week 17 (30/12)
Finals

Objectives

A. Apply the `list` data type as a container, including indexing.
B. Employ `for` loops using `list`s as iterators.
C. Use *method*s built in data types to manipulate data.

Recap on if and Loops

Question 1 if statements

day = 3, What will be the output?

```
if day => 1:
    print(”Monday, really? When was Sunday!”)
elif day => 2:
    print(”Tuesday....”)
elif day => 3:
    print(”Wednesday, Hump day!”)
else:
    print(”Boring...”)
```

Question 1 if statements

day = 3, What will be the output?

```
if day => 1:
    print(”Monday, really? When was Sunday!”)
elif day => 2:
    print(”Tuesday....”)
elif day => 3:
    print(”Wednesday, Hump day!”)
else:
    print(”Boring...”)
```

Ans: Syntax Error! Why????

Question 2 if statements

day = 3, What will be the output?

```
if day >= 1:
    print(”Monday, So happy to attend CS101!”)
elif day >= 2:
    print(”Tuesday, Exciting stuff coming”)
elif day >= 3:
    print(”Wednesday, CS101!”)
else:
    print(”Boring...”)
```

Question 2 if statements

day = 3, What will be the output?

```
if day >= 1:
    print(”Monday, So happy to attend CS101!”)
elif day >= 2:
    print(”Tuesday, Exciting stuff coming”)
elif day >= 3:
    print(”Wednesday, CS101!”)
else:
    print(”Boring...”)
```

Ans:
’Monday, So happy to attend CS101!’

Question 4 while loop

```
i = 0
sum = 0
while i < 5:
    if (i % 2) == 1 or (i % 2) == 0:
        sum += i
        i += 1
```

What is the value of `sum`?

A `15`

B `0`

C `10`

D `1`

E None of the above.

Question 4 while loop

```
i = 0
sum = 0
while i < 5:
    if (i % 2) == 1 or (i % 2) == 0:
        sum += i
        i += 1
```

What is the value of `sum`?

A `15`

B `0`

C `10` ⋆

D `1`

E None of the above.

Question 5 while loop

```
i = 0
sum = 0
while i < 5:
    if (i % 2) == 1 or (i % 2) == 0:
        sum += i
        i += 1
    print(i)
print(sum)
```

How many times will `i` be printed?

Question 5 while loop

```
i = 0
sum = 0
while i < 5:
    if (i % 2) == 1 or (i % 2) == 0:
        sum += i
        i += 1
    print(i)
print(sum)
```

How many times will `i` be printed?

ans: 5

How many times will `sum` be printed?

Question 5 while loop

```
i = 0
sum = 0
while i < 5:
    if (i % 2) == 1 or (i % 2) == 0:
        sum += i
        i += 1
    print(i)
print(sum)
```

How many times will `i` be printed?
ans: 5
How many times will `sum` be printed?
ans: 1

Question 6 range

```
range(1,10) = [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Question 6 range

```
range(1,10) = [1, 2, 3, 4, 5, 6, 7, 8, 9]
range(5) = ?
```

Question 6 range

`range(1,10) = [1, 2, 3, 4, 5, 6, 7, 8, 9]`

`range(5) = ?`

ans: `[0, 1, 2, 3, 4]` as `range(5) = range(,5,)`

Question 7 for

```
for i in range(1,10):
    print(i)
```

Question 7 for

```
for i in range(1,10):
    print(i)
```

1
2
...
9

Question 8 for

```
x = 5
for i in x:
    print(i)
```

Question 8 for

```
x = 5
for i in x:
    print(i)
```

Error

```
x = 5
for i in range(x):
    print(i)
```

Question 8 for

```
x = 5
for i in x:
    print(i)
```

Error

```
x = 5
for i in range(x):
    print(i)
```

0
1
...
4

Containers: lists

`list` is a data type

The `list` type represents an ordered collection of items.

Containers hold values of any type (doesn't have to be the same).

list statement

We create a `list` as follows:

- opening bracket `[`
- one or more comma-separated data values
- closing bracket `]`

list statement

lists work a bit like strings:

```
x = [ 10, 3.14, "Ride" ]

print( x[1] )
print( x[1:3] )
print( x[1:10] )
print( x[0::2] )
print( len(x) )
```

list statement

`list`s work a bit like strings:

```
x = [ 10, 3.14, "Ride" ]

print( x[1] )
print( x[1:3] )
print( x[1:10] )
print( x[0::2] )
print( len(x) )

3.14
```

list statement

lists work a bit like strings:

```
x = [ 10, 3.14, "Ride" ]

print( x[1] )
print( x[1:3] )
print( x[1:10] )
print( x[0::2] )
print( len(x) )
```

```
3.14
[3.14, "Ride"]
```

list statement

lists work a bit like strings:

```
x = [ 10, 3.14, "Ride" ]

print( x[1] )
print( x[1:3] )
print( x[1:10] )
print( x[0::2] )
print( len(x) )
```

```
3.14
[3.14, "Ride"]
[3.14, "Ride"]
```

list statement

lists work a bit like strings:

```
x = [ 10, 3.14, "Ride" ]

print( x[1] )
print( x[1:3] )
print( x[1:10] )
print( x[0::2] )
print( len(x) )
```

```
3.14
[3.14, "Ride"]
```

`[3.14, "Ride"]` => python treat the slice as boundary not exact index number

```
[10, "Ride"]
```

list statement

lists work a bit like strings:

```
x = [ 10, 3.14, "Ride" ]

print( x[1] )
print( x[1:3] )
print( x[1:10] )
print( x[0::2] )
print( len(x) )
```

```
3.14
[3.14, "Ride"]
[3.14, "Ride"]
```

=> python treat the slice as boundary not exact index number

```
[10, "Ride"]
3
```

for and list

```
things = [ '1 thing', '2 things', 'nothing',
                                  'everything' ]
for thing in things:
    print( 'I have %s.' % thing )
```

Ans:

for and list

```
things = [ '1 thing', '2 things', 'nothing',
                                  'everything' ]
for thing in things:
    print( 'I have %s.' % thing )
```

Ans:

I have 1 thing.

I have 2 things.

I have nothing.

I have everything.

Methods

Like attributes in variables, functions can be built inside a data type as well.

Use attribute operator `.` to access these built-in functions.

Methods

Like attributes in variables, functions can be built inside a data type as well.

Use attribute operator `.` to access these built-in functions.

```
"REALLY A NICE DINNER LAST NIGHT!".lower()
```

Methods

Like attributes in variables, functions can be built inside a data type as well.

Use attribute operator `.` to access these built-in functions.

```
"REALLY A NICE DINNER LAST NIGHT!".lower()
"especially the spicy orange juice?!?".upper()
```

Methods

Like attributes in variables, functions can be built inside a data type as well.

Use attribute operator `.` to access these built-in functions.

```
”REALLY A NICE DINNER LAST NIGHT!”.lower()
”especially the spicy orange juice?!?”.upper()
(1 + 1j).conjugate()
```

Methods

Like attributes in variables, functions can be built inside a data type as well.

Use attribute operator `.` to access these built-in functions.

```
”REALLY A NICE DINNER LAST NIGHT!”.lower()
”especially the spicy orange juice?!?”.upper()
(1 + 1j).conjugate()
```

”Value” infront of the ”`.`” operator is treated like an argument.

Most (not all) RETURN their value.

list methods

We *can* change `list` content—they are *mutable*.

```
x = [ 4,1,2,3 ]
x[3] = -2          # sets an element
x.append(5)        # adds an element
del x[1]           # removes an element
x.sort()           # sorts in place
```

Question 1

Given,

```
y = [10]
z = [1.0, 22, 'so pretty and handsome', 10+9.7j]
```

What will the following commands output?

A. `y + z`

Question 1

Given,

```
y = [10]
z = [1.0, 22, 'so pretty and handsome', 10+9.7j]
```

What will the following commands output?

A. `y + z`

= [10, 1.0, 22, 'so pretty and handsome', 10+9.7j]

B. `y * 4`

Question 1

Given,

```
y = [10]
z = [1.0, 22, 'so pretty and handsome', 10+9.7j]
```

What will the following commands output?

A. `y + z`

= [10, 1.0, 22, 'so pretty and handsome', 10+9.7j]

B. `y * 4`

= [10, 10, 10, 10]

Remember `list` is a container

C. `z[ 2 ][ 0:3 ]`

Question 1

Given,

```
y = [10]
z = [1.0, 22, 'so pretty and handsome', 10+9.7j]
```

What will the following commands output?

A. `y + z`

= [10, 1.0, 22, 'so pretty and handsome', 10+9.7j]

B. `y * 4`

= [10, 10, 10, 10]

Remember `list` is a container

C. `z[ 2 ][ 0:3 ]`

= 'so '

Question 2

Given,

```
x = ['hello', 2, 'everyone']
```

What will the following commands output?

1. `x + 5`

Question 2

Given,

```
x = ['hello', 2, 'everyone']
```

What will the following commands output?

1. `x + 5`

   error

2. `x + [5]`

Question 2

Given,

```
x = ['hello', 2, 'everyone']
```

What will the following commands output?

1. `x + 5`

   error

2. `x + [5]`

   = ['hello', 2, 'everyone', 5]

   `what is x now?`

Question 2

Given,

```
x = ['hello', 2, 'everyone']
```

What will the following commands output?

1. `x + 5`

   error

2. `x + [5]`

   = ['hello', 2, 'everyone', 5]

   `what is x now?`

   = ['hello', 2, 'everyone'] `=> remains the same!`

3. `x.append(5)`

Question 2

Given,

```
x = ['hello', 2, 'everyone']
```

What will the following commands output?

1. `x + 5`

   error

2. `x + [5]`

   = ['hello', 2, 'everyone', 5]

   `what is x now?`

   = ['hello', 2, 'everyone'] `=> remains the same!`

3. `x.append(5)`

   = ['hello', 2, 'everyone', 5]

   `what is x now?`

Question 2

Given,

```
x = ['hello', 2, 'everyone']
```

What will the following commands output?

1. `x + 5`

   error

2. `x + [5]`

   = ['hello', 2, 'everyone', 5]

   `what is x now?`

   = ['hello', 2, 'everyone'] `=> remains the same!`

3. `x.append(5)`

   = ['hello', 2, 'everyone', 5]

   `what is x now?`

   = ['hello', 2, 'everyone', 5] ! `=> Changed!`

Question 3

Given,

```
x = ['I', 'love', 'to', 'study']
y = ['not', 'holiday']
```

What will the following commands output?

CASE 1

```
x + y
x
```

Question 3

Given,

```
x = ['I', 'love', 'to', 'study']
y = ['not', 'holiday']
```

What will the following commands output?

CASE 1

```
x + y
x
```

> ['I', 'love', 'to', 'study', 'not', 'holiday']
> x = ['I', 'love', 'to', 'study']

Question 3

Given,

```
x = ['I', 'love', 'to', 'study']
y = ['not', 'holiday']
```

What will the following commands output?

CASE 1

```
x + y
x
```

> ['I', 'love', 'to', 'study', 'not', 'holiday']
> x = ['I', 'love', 'to', 'study']

CASE 2

```
x.append( y )
x
```

Question 3

Given,

```
x = ['I', 'love', 'to', 'study']
y = ['not', 'holiday']
```

What will the following commands output?

CASE 1

```
x + y
x
```

> ['I', 'love', 'to', 'study', 'not', 'holiday']
> x = ['I', 'love', 'to', 'study']

CASE 2

```
x.append( y )
x
```

x = ['I', 'love', 'to', 'study', ['not', 'holiday']]

Casting range output as list

```
range( 0, 6, 2 )
list( range( 0, 6, 2 ) )
[ 0, 2, 4 ]
```

Fancy slicing for containers

```
a = list( range( 10 ) )
#   [ 0,1,2,3,4,5,6,7,8,9 ]

a[ :4 ]       # from beginning to index 4 (exc.)
a[ 6: ]       # from index 6 to end
a[ : ]        # copy a list
a[ 1:-1:2 ]   # from index 1 to -1 by twos
a[ 1::2 ]     # odd indices only
a[ ::2 ]      # even indices only
a[ ::-1 ]     # reverse a list (!)
```

Question

```
x = 0
for i in [ 1,4950,99,100 ][ 0:-1 ]:
    x = i
```

What is the final value which `x` assumes?

A `0`

B `99`

C `100`

D `4950`

Question

```
x = 0
for i in [ 1,4950,99,100 ][ 0:-1 ]:
    x = i
```

What is the final value which `x` assumes?

A `0`

B `99` ⋆

C `100`

D `4950`

String Methods

String methods

`upper()`	convert to upper-case
`lower()`	convert to lower-case
`count( str1 )`	count occurrences of `str1`
`replace( s1, s2 )`	replace `s1` by `s2`
`strip()`	remove whitespace at both ends

String comparison methods

These produce Boolean output.

`isdigit()`	Does a string contain only numbers?
`isalpha()`	Does a string contain only text?
`isalnum()`	Does a string contain text and number
`islower()`	Are all the letters in a string lower-case?
`isupper()`	Are all the letters in a string contain upper-case?

Example: String comparison

```
answer = input( 'How do you feel?' )
```

Example: String comparison

```
answer = input( 'How do you feel?' )
```

What is type(answer)?

Example: String comparison

```
answer = input( 'How do you feel?' )
```

What is type(answer)? String!

```
if not answer.isalpha():
    print( "Excellent! I don't understand you." )
else:
    print( "Ah, you feel %s." % answer )
```

Fun time

Write a code for testing a user's new password. These rules should be applied:

A. Minimum length of 8
B. Both upper and lower case letters
C. have both letters and digits

Hint: Can use `isdigit()`, `isalpha()`, `islower()`, `isupper()`

Solution

```
if len( try ) < 8:
    # must be 8 characters at a minimum
     print(”False”)
else:
     if try.isupper() or try.islower():
     # must have both upper- & lower-case letters
          print(”False”)
     else:
          if try.isalpha() or try.isdigit():
          # must have letters and numbers
               print(”False”)
          else:
               print(”True”)
```

However, this code will allow special characters like ” !@#$%&*(),...” to pass as well. See next lecture for a better answer.

Summary

Summary

1. `list`
2. `for ... in list:`
3. `list.method()` that affects the `list`
4. `string.method()`