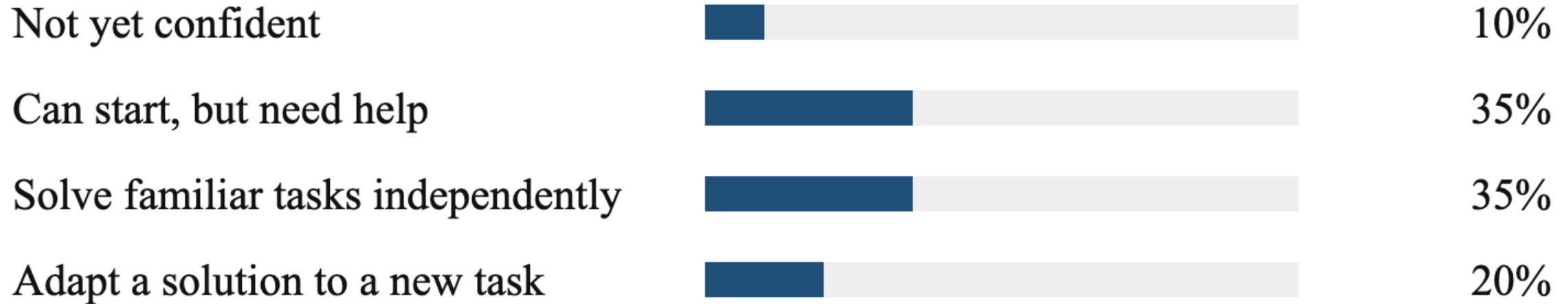


L-3

Python Collections and Loops

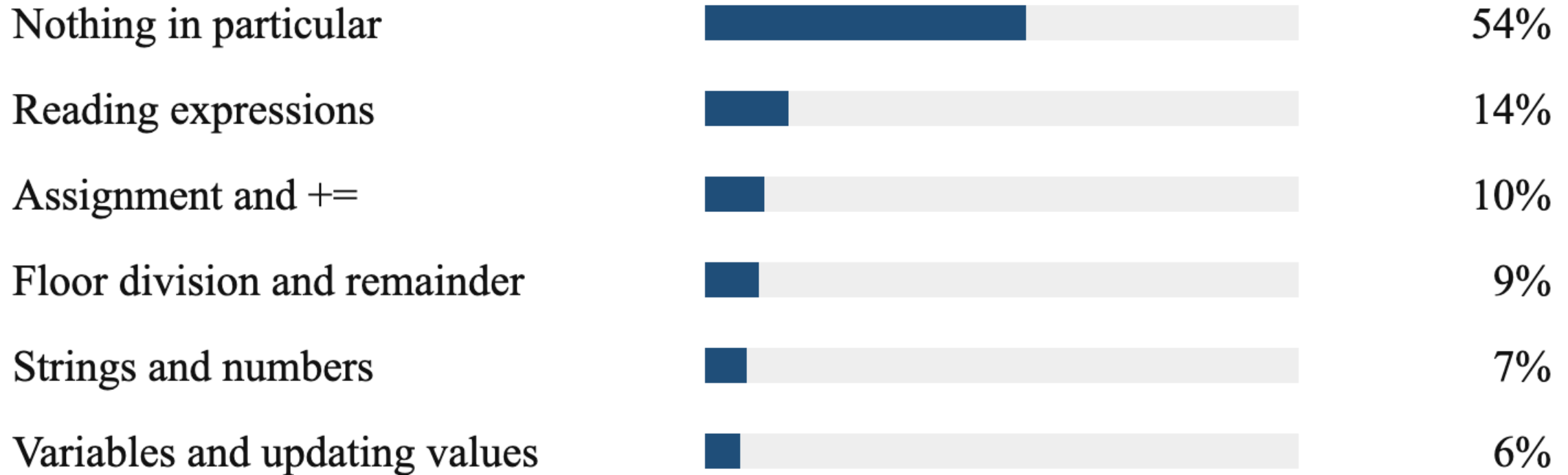
ENGG1330
Computer Programming I

L-2: How Confident Do You Feel?



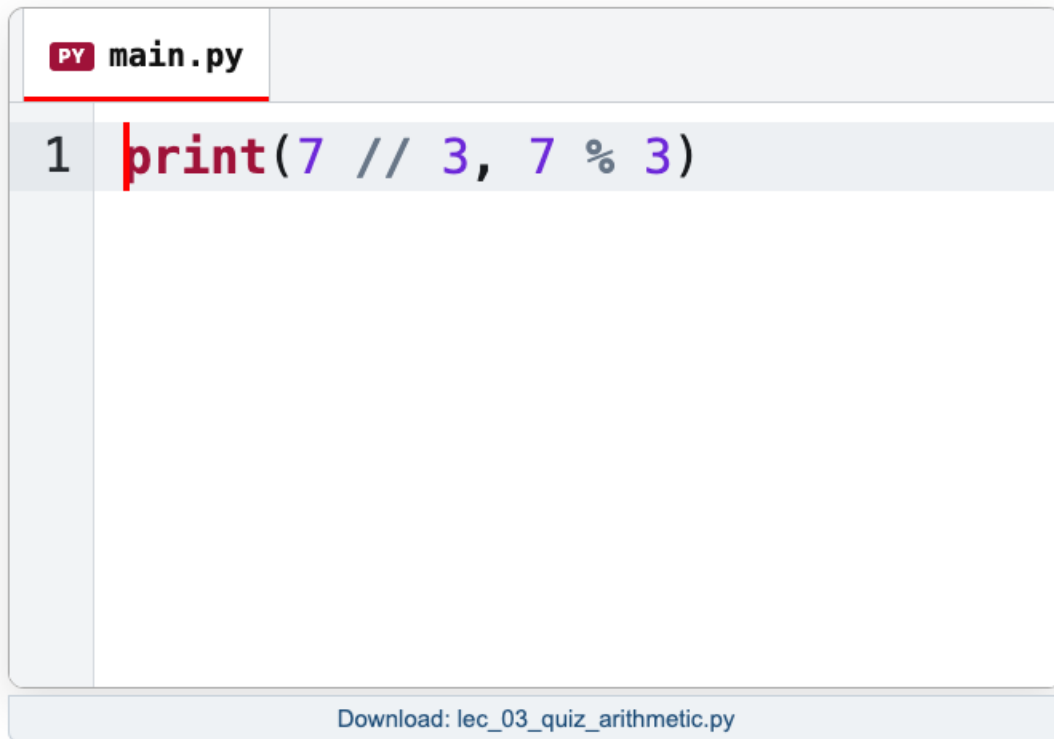
55% feel ready to work independently
45% still need support

L-2: What Felt Most Difficult?



L-2: What Did the Quiz Reveal?

L-2: What Did the Quiz Reveal?

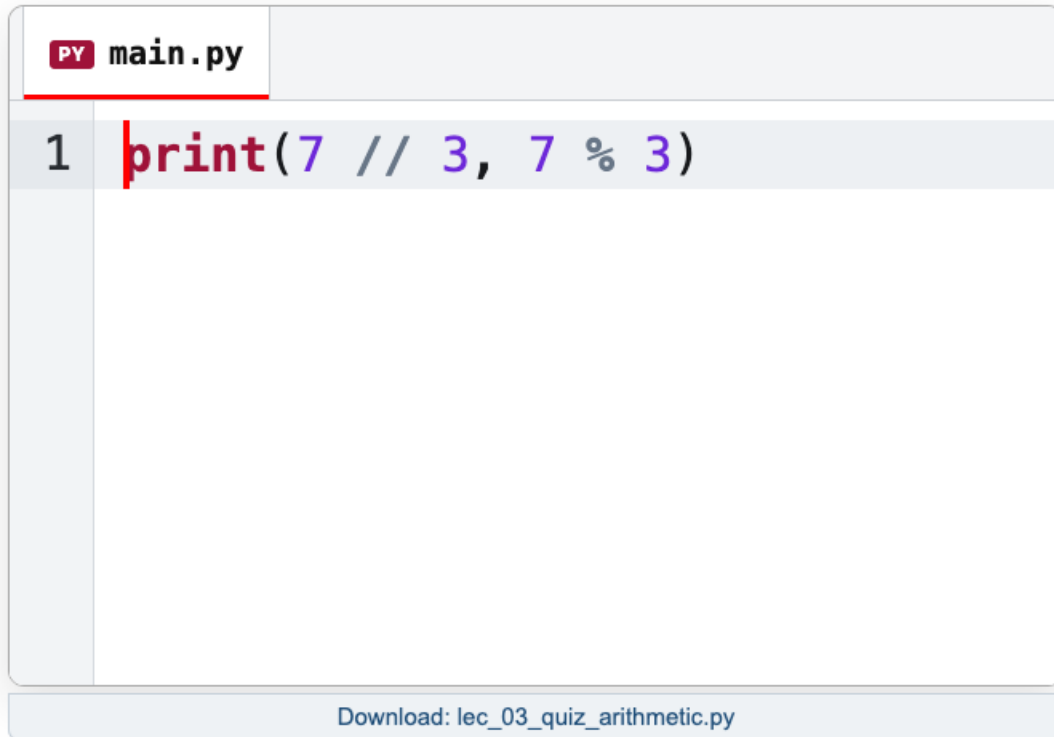


```
PY main.py
1 print(7 // 3, 7 % 3)
```

Download: [lec_03_quiz_arithmetic.py](#)

L-2: What Did the Quiz Reveal?

97% correct



```
PY main.py
1 print(7 // 3, 7 % 3)
```

Download: lec_03_quiz_arithmetic.py

L-2: What Did the Quiz Reveal?

97% correct

```
PY main.py
1 print(7 // 3, 7 % 3)
```

Download: lec_03_quiz_arithmetic.py

```
PY main.py
1 x = 5
2 y = x
3 x += 2
4 print(y)
```

Download: lec_03_quiz_assignment.py

L-2: What Did the Quiz Reveal?

97% correct

```
PY main.py
1 print(7 // 3, 7 % 3)
```

Download: lec_03_quiz_arithmetic.py

70% correct

```
PY main.py
1 x = 5
2 y = x
3 x += 2
4 print(y)
```

Download: lec_03_quiz_assignment.py

Participation

Participation

Lecture L-1

71.2%

Participation

Lecture L-1	71.2%
<hr/>	
Lecture L-2	92.1%
<hr/>	

Participation

Lecture L-1	71.2%
<hr/>	
Lecture L-2	92.1%
<hr/>	
Tutorial T-1	96.2%

Tuples

ENGG1330
Computer Programming I

A Tuple Is an Ordered Sequence

```
PY main.py
1 fruits = ('apple', 'orange', 'pear', 'banana')
2 print(fruits)
3 print(fruits[0])
```

[Download: lec_03_example_03.py](#)

A Tuple Is an Ordered Sequence

fruits

0	1	2	3
apple	orange	pear	banana

```
PY main.py
1 fruits = ('apple', 'orange', 'pear', 'banana')
2 print(fruits)
3 print(fruits[0])
```

Download: [lec_03_example_03.py](#)

A Tuple Is an Ordered Sequence

fruits[0]

0	1	2	3
apple	orange	pear	banana

```
PY main.py
1 fruits = ('apple', 'orange', 'pear', 'banana')
2 print(fruits)
3 print(fruits[0])
```

Download: [lec_03_example_03.py](#)

Indexing Works Like a String

```
PY main.py
1 | fruits = ('apple', 'orange', 'pear', 'banana')
2 | s = 'apple'
3 | print(fruits[0])
4 | print(s[0])
```

Download: lec_03_example_04.py

One Tuple Can Mix Types

```
PY main.py
1 t = (3, 3.14159, 'Bruce Lee', True)
2 print(type(t))
3 print(type(t[0]))
4 print(type(t[1]))
```

Download: [lec_03_example_06.py](#)

A Tuple Can Contain Tuples

```
PY main.py
1 t = ((1, 2), (2, 3), (3, 3))
2 print(t)
3 print(len(t))
4 print(t[0])
5 print(len(t[0]))
```

Download: lec_03_example_07.py

Unpacking Assigns Several Names

```
PY main.py
1 t = 'Dirk Schnieders', 29, 'A-'
2 name, age, grade = t
3 print('Name:', name)
4 print('Age:', age)
5 print('Grade:', grade)
```

Download: [lec_03_example_08.py](#)

Unpacking Assigns Several Names

Unpack t into three names

name	age	grade
Dirk Schnieders	29	A-

PY main.py

```
1 t = 'Dirk Schnieders', 29, 'A-'
2 name, age, grade = t
3 print('Name:', name)
4 print('Age:', age)
5 print('Grade:', grade)
```

Unpacking Assigns Several Names

name		
name	age	grade
Dirk Schnieders	29	A-

PY main.py

```
1 t = 'Dirk Schnieders', 29, 'A-'
2 name, age, grade = t
3 print('Name:', name)
4 print('Age:', age)
5 print('Grade:', grade)
```

Unpacking Assigns Several Names

name	age	grade
Dirk Schnieders	29	A-

PY main.py

```
1 t = 'Dirk Schnieders', 29, 'A-'
2 name, age, grade = t
3 print('Name:', name)
4 print('Age:', age)
5 print('Grade:', grade)
```

Unpacking Assigns Several Names

grade

name	age	grade
Dirk Schnieders	29	A-

PY main.py

```
1 t = 'Dirk Schnieders', 29, 'A-'
2 name, age, grade = t
3 print('Name:', name)
4 print('Age:', age)
5 print('Grade:', grade)
```

Joining Tuples with +

```
PY main.py
1 t1 = (1, 2, 3, 4, 5, 6)
2 t2 = (7, 8, 9)
3 t = t1 + t2
4 print(t)
```

Download: [lec_03_example_09.py](#)

A Slice Stops Before the Stop Index

0	1	2	3	4	5
kiwi	strawberry	apple	orange	pear	banana

PY main.py

```
1 fruits = ('kiwi', 'strawberry', 'apple', 'orange', 'pear', 'banana')
2 print(fruits[1:3])
3 print(fruits[:3])
4 print(fruits[2:])
5
```

A Slice Stops Before the Stop Index

`fruits[1:3]`

0	1	2	3	4	5
kiwi	strawberry	apple	orange	pear	banana

PY main.py

```
1 fruits = ('kiwi', 'strawberry', 'apple', 'orange', 'pear', 'banana')
2 print(fruits[1:3])
3 print(fruits[:3])
4 print(fruits[2:])
5
```

A Slice Stops Before the Stop Index

`fruits[:3]`

0	1	2	3	4	5
kiwi	strawberry	apple	orange	pear	banana

PY main.py

```
1 fruits = ('kiwi', 'strawberry', 'apple', 'orange', 'pear', 'banana')
2 print(fruits[1:3])
3 print(fruits[:3])
4 print(fruits[2:])
5
```

A Slice Stops Before the Stop Index

`fruits[2:]`

0	1	2	3	4	5
kiwi	strawberry	apple	orange	pear	banana

PY main.py

```
1 fruits = ('kiwi', 'strawberry', 'apple', 'orange', 'pear', 'banana')
2 print(fruits[1:3])
3 print(fruits[:3])
4 print(fruits[2:])
5
```

Negative Indices Count from the End

-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
0	1	2	3	4	5	6	7	8	9

PY main.py

```
1 t = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
2 print(t[-5:])
3 print(t[-3:])
4 print(t[:-3])
```

Download: [lec_03_example_11.py](#)

Negative Indices Count from the End

`t[-5:]`

-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
0	1	2	3	4	5	6	7	8	9

PY main.py

```
1 t = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
2 print(t[-5:])
3 print(t[-3:])
4 print(t[:-3])
```

Download: [lec_03_example_11.py](#)

Negative Indices Count from the End

`t[-3:]`

-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
0	1	2	3	4	5	6	7	8	9

PY main.py

```
1 t = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
2 print(t[-5:])
3 print(t[-3:])
4 print(t[:-3])
```

Download: [lec_03_example_11.py](#)

Negative Indices Count from the End

`t[:-3]`

-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
0	1	2	3	4	5	6	7	8	9

PY main.py

```
1 t = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
2 print(t[-5:])
3 print(t[-3:])
4 print(t[:-3])
```

Download: [lec_03_example_11.py](#)

A Slice Can Specify a Step

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

PY main.py

```
1 t = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14)
2 print(t[0:10])
3 print(t[0:10:2])
4 print(t[0::2])
5 print(t[::-2])
6
```

A Slice Can Specify a Step

`t[0:10]`

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

PY main.py

```
1 t = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14)
2 print(t[0:10])
3 print(t[0:10:2])
4 print(t[0::2])
5 print(t[:,2])
6
```

A Slice Can Specify a Step

`t[0:10:2]`

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

PY main.py

```
1 t = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14)
2 print(t[0:10])
3 print(t[0:10:2])
4 print(t[0::2])
5 print(t[::2])
6
```

A Slice Can Specify a Step

`t[0::2]`

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

PY main.py

```
1 t = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14)
2 print(t[0:10])
3 print(t[0:10:2])
4 print(t[0::2])
5 print(t[:,2])
6
```

A Slice Can Specify a Step

`t[::2]`

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

PY main.py

```
1 t = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14)
2 print(t[0:10])
3 print(t[0:10:2])
4 print(t[0::2])
5 print(t[::2])
6
```

Lists

ENGG1330
Computer Programming I

Lists Use Square Brackets

PY main.py

```
1 fruits = ['apple', 'orange', 'pear', 'banana']  
2 print(fruits[-2])  
3 print(fruits[-1])  
4 print(fruits[-3])  
5 print(len(fruits))
```

Lists Use Square Brackets

`fruits[-2]`

-4	-3	-2	-1
apple	orange	pear	banana

PY main.py

```
1 fruits = ['apple', 'orange', 'pear', 'banana']
2 print(fruits[-2])
3 print(fruits[-1])
4 print(fruits[-3])
5 print(len(fruits))
```

Lists Use Square Brackets

`fruits[-1]`

-4	-3	-2	-1
apple	orange	pear	banana

PY main.py

```
1 fruits = ['apple', 'orange', 'pear', 'banana']
2 print(fruits[-2])
3 print(fruits[-1])
4 print(fruits[-3])
5 print(len(fruits))
```

Lists Use Square Brackets

`fruits[-3]`

-4	-3	-2	-1
apple	orange	pear	banana

PY main.py

```
1 fruits = ['apple', 'orange', 'pear', 'banana']
2 print(fruits[-2])
3 print(fruits[-1])
4 print(fruits[-3])
5 print(len(fruits))
```

Lists Use Square Brackets

`len(fruits) = 4`

-4	-3	-2	-1
apple	orange	pear	banana

PY main.py

```
1 fruits = ['apple', 'orange', 'pear', 'banana']
2 print(fruits[-2])
3 print(fruits[-1])
4 print(fruits[-3])
5 print(len(fruits))
```

Tuple Elements Cannot Be Replaced

PY main.py

```
1 fruits = ('apple', 'orange', 'pear', 'banana')
2 fruits[-1] = 'pineapple'
3 print(fruits)
```

[Download: lec_03_example_15.py](#)

List Elements Can Be Replaced

PY main.py

```
1 fruits = ['apple', 'orange', 'pear', 'banana']  
2 fruits[-1] = 'pineapple'  
3 print(fruits)
```

[Download: lec_03_example_16.py](#)

Two Names Can Refer to the Same List

```
main.py
1 a = ['apple', 'pear']
2 b = a
3 b[0] = 'kiwi'
4 print('a:', a, 'b:', b)
5 b = ['banana']
6 print('a:', a, 'b:', b)
```

Download: [lec_03_shared_list.py](#)

Two Names Can Refer to the Same List

`b = a`: both names refer to the same list



PY main.py

```
1 a = ['apple', 'pear']
2 b = a
3 b[0] = 'kiwi'
4 print('a:', a, 'b:', b)
5 b = ['banana']
6 print('a:', a, 'b:', b)
```

Two Names Can Refer to the Same List

`b[0] = 'kiwi': the shared list changes`



PY main.py

```
1 a = ['apple', 'pear']
2 b = a
3 b[0] = 'kiwi'
4 print('a:', a, 'b:', b)
5 b = ['banana']
6 print('a:', a, 'b:', b)
```

Two Names Can Refer to the Same List

`b = ['banana']`: only `b` refers to the new list



PY main.py

```
1 a = ['apple', 'pear']
2 b = a
3 b[0] = 'kiwi'
4 print('a:', a, 'b:', b)
5 b = ['banana']
6 print('a:', a, 'b:', b)
```

Choosing a Tuple or a List

Tuple	List (Default)
A fixed group of related values	A general collection of items
(name, age, grade)	['apple', 'orange']
Prevents accidental element replacement	Add, remove, or replace items

Choosing a Tuple or a List

Tuple	List (Default)
A fixed group of related values	A general collection of items
(name, age, grade)	['apple', 'orange']
Prevents accidental element replacement	Add, remove, or replace items
Tuples usually use less memory and can be dictionary keys when all their elements are hashable	

Choosing a Tuple or a List

Tuple	List (Default)
A fixed group of related values	A general collection of items
(name, age, grade)	['apple', 'orange']
Prevents accidental element replacement	Add, remove, or replace items
Tuples usually use less memory and can be dictionary keys when all their elements are hashable	
A variable can refer to a different tuple	

Adding an Item to a List

```
PY main.py
1 fruits = []
2 fruits.append('apple')
3 fruits.append('orange')
4 fruits.insert(1, 'pineapple')
5 print(fruits)
```

Download: lec_03_example_18.py

Adding an Item to a List

`append('apple')`

0

apple

PY main.py

```
1 fruits = []  
2 fruits.append('apple')  
3 fruits.append('orange')  
4 fruits.insert(1, 'pineapple')  
5 print(fruits)
```

Download: [lec_03_example_18.py](#)

Adding an Item to a List

`append('orange')`

0	1
apple	orange

PY main.py

```
1 fruits = []
2 fruits.append('apple')
3 fruits.append('orange')
4 fruits.insert(1, 'pineapple')
5 print(fruits)
```

Download: [lec_03_example_18.py](#)

Adding an Item to a List

```
insert(1, 'pineapple')
```

0	1	2
apple	pineapple	orange

PY main.py

```
1 fruits = []
2 fruits.append('apple')
3 fruits.append('orange')
4 fruits.insert(1, 'pineapple')
5 print(fruits)
```

Download: lec_03_example_18.py

Deleting an Item from a List

```
PY main.py
1 fruits = ['apple', 'orange', 'pear', 'banana']
2 fruits.insert(2, 'pineapple')
3 print(fruits)
4 del fruits[2]
5 print(fruits)
```

Download: [lec_03_example_19.py](#)

Deleting an Item from a List

`insert(2, 'pineapple')`

0	1	2	3	4
apple	orange	pineapple	pear	banana

PY main.py

```
1 fruits = ['apple', 'orange', 'pear', 'banana']
2 fruits.insert(2, 'pineapple')
3 print(fruits)
4 del fruits[2]
5 print(fruits)
```

Download: [lec_03_example_19.py](#)

Deleting an Item from a List

`del fruits[2]: later items shift left`

0	1	2	3
apple	orange	pear	banana

PY main.py

```
1 fruits = ['apple', 'orange', 'pear', 'banana']
2 fruits.insert(2, 'pineapple')
3 print(fruits)
4 del fruits[2]
5 print(fruits)
```

[Download: lec_03_example_19.py](#)

Finding Help for a Method

PY main.py

```
1 fruits = ['apple', 'orange', 'pear']  
2 help(fruits.pop)
```

Python Documentation

docs.python.org/3

Dictionaries

ENGG1330
Computer Programming I

A Dictionary Maps Keys to Values

```
PY main.py
1 fruit_by_code = {10: 'apple', 20: 'orange',
2                  30: 'pear', 40: 'banana'}
3 print(fruit_by_code[20])
4 print(fruit_by_code[10])
```

Download: [lec_03_example_22.py](#)

A Dictionary Maps Keys to Values

fruit_by_code[20]

10	20	30	40
apple	orange	pear	banana

```
PY main.py
1 fruit_by_code = {10: 'apple', 20: 'orange',
2                  30: 'pear', 40: 'banana'}
3 print(fruit_by_code[20])
4 print(fruit_by_code[10])
```

Download: [lec_03_example_22.py](#)

A Dictionary Maps Keys to Values

fruit_by_code[10]

10	20	30	40
apple	orange	pear	banana

```
PY main.py
1 fruit_by_code = {10: 'apple', 20: 'orange',
2                  30: 'pear', 40: 'banana'}
3 print(fruit_by_code[20])
4 print(fruit_by_code[10])
```

Download: [lec_03_example_22.py](#)

Keys Can Be Strings or Numbers

```
PY main.py
1 fruits = {'favourite': 'apple',
2           10: 'orange', 20: 'pear', 30: 'banana'}
3 print(fruits['favourite'])
4 print(fruits[10])
```

Download: [lec_03_example_23.py](#)

Keys Can Be Strings or Numbers

`fruits['favourite']`

'favourite'	10	20	30
apple	orange	pear	banana

PY main.py

```
1 fruits = {'favourite': 'apple',  
2         10: 'orange', 20: 'pear', 30: 'banana'}  
3 print(fruits['favourite'])  
4 print(fruits[10])
```

[Download: lec_03_example_23.py](#)

Keys Can Be Strings or Numbers

fruits[10]

'favourite'	10	20	30
apple	orange	pear	banana

PY main.py

```
1 fruits = {'favourite': 'apple',  
2         | 10: 'orange', 20: 'pear', 30: 'banana'}  
3 print(fruits['favourite'])  
4 print(fruits[10])
```

[Download: lec_03_example_23.py](#)

Looking Up a Mood

```
PY main.py
1 mood = {'sunny': ':-) ',
2         'cloudy': ':-| ',
3         'rainy': ':-( ' }
4 print(mood['sunny'])
5 print(mood['rainy'])
```

[Download: lec_03_example_24.py](#)

Looking Up a Mood

mood['sunny']

'sunny'	'cloudy'	'rainy'
:)	:	:(

PY main.py

```
1 mood = {'sunny': ':)',  
2         'cloudy': ':|',  
3         'rainy': ':( '  
4 print(mood['sunny'])  
5 print(mood['rainy'])
```

[Download: lec_03_example_24.py](#)

Looking Up a Mood

mood['rainy']

'sunny'	'cloudy'	'rainy'
:~)	:~	:~(

PY main.py

```
1 mood = {'sunny': ':~)',  
2         'cloudy': ':~|',  
3         'rainy': ':~('}  
4 print(mood['sunny'])  
5 print(mood['rainy'])
```

Download: lec_03_example_24.py

Updating a Dictionary Value

```
PY main.py
1 stock = {'apple': 5, 'bun': 3}
2 print(stock['apple'])
3 stock['apple'] -= 2
4 print(stock['apple'])
```

Download: [lec_03_example_25.py](#)

Updating a Dictionary Value

```
stock['apple'] = 5
```

apple	bun
5	3

PY main.py

```
1 stock = {'apple': 5, 'bun': 3}
2 print(stock['apple'])
3 stock['apple'] -= 2
4 print(stock['apple'])
```

[Download: lec_03_example_25.py](#)

Updating a Dictionary Value

```
stock['apple'] -= 2
```

apple	bun
3	3

PY main.py

```
1 stock = {'apple': 5, 'bun': 3}
2 print(stock['apple'])
3 stock['apple'] -= 2
4 print(stock['apple'])
```

[Download: lec_03_example_25.py](#)

Sets

ENGG1330
Computer Programming I

A Set Stores Distinct Elements

```
PY main.py
1 shapes = {'square', 'triangle', 'circle'}
2 print('polygon' in shapes)
3 print('polygon' not in shapes)
```

[Download: lec_03_example_27.py](#)

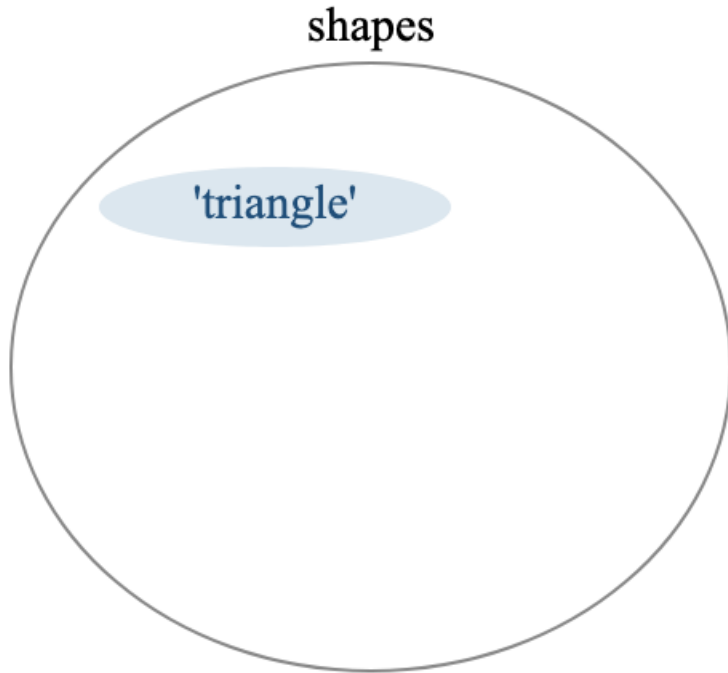
Duplicates Disappear in a Set

PY main.py

```
1 shapes = set() # Create an empty set
2 shapes.add('triangle')
3 shapes.add('circle')
4 shapes.add('triangle')
5 print(shapes)
6 shapes.add('polygon')
7 print(shapes)
```

Duplicates Disappear in a Set

`add('triangle')`

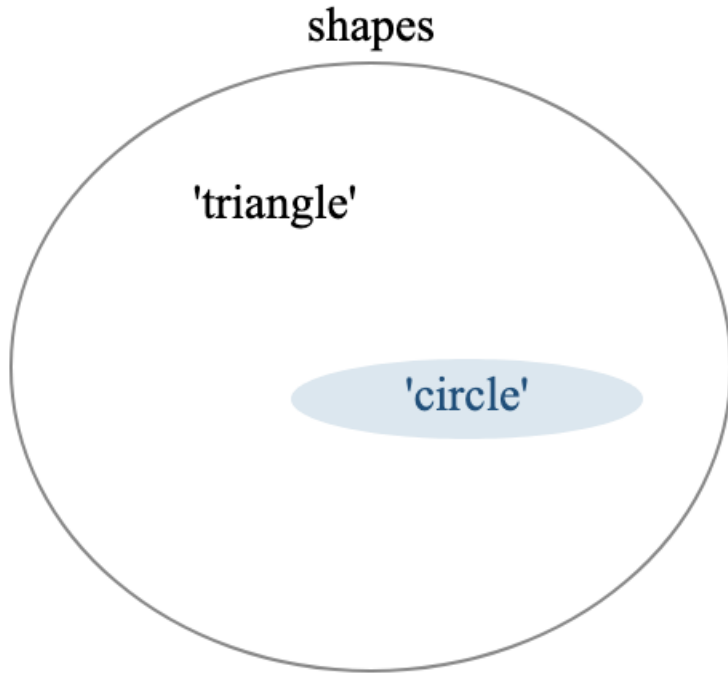


PY main.py

```
1 shapes = set() # Create an empty set
2 shapes.add('triangle')
3 shapes.add('circle')
4 shapes.add('triangle')
5 print(shapes)
6 shapes.add('polygon')
7 print(shapes)
```

Duplicates Disappear in a Set

`add('circle')`

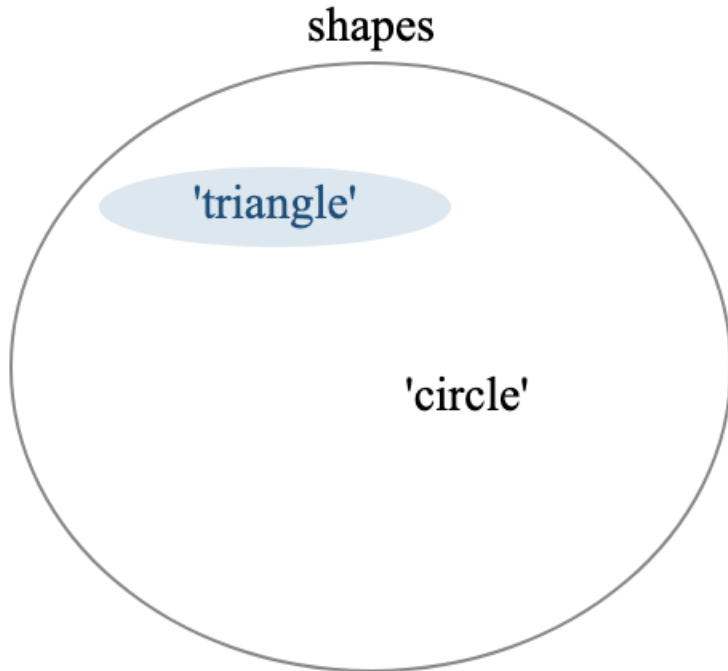


PY main.py

```
1 shapes = set() # Create an empty set
2 shapes.add('triangle')
3 shapes.add('circle')
4 shapes.add('triangle')
5 print(shapes)
6 shapes.add('polygon')
7 print(shapes)
```

Duplicates Disappear in a Set

`add('triangle')` again: no duplicate

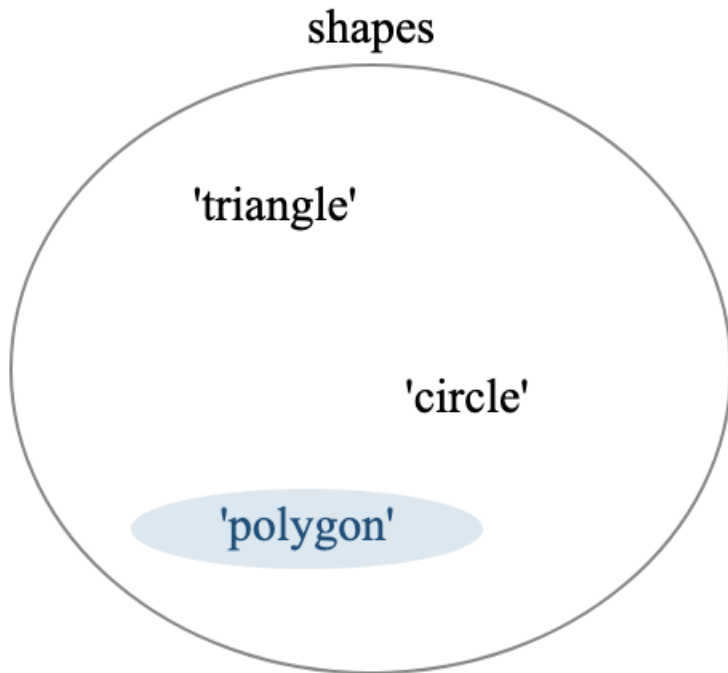


PY main.py

```
1 shapes = set() # Create an empty set
2 shapes.add('triangle')
3 shapes.add('circle')
4 shapes.add('triangle')
5 print(shapes)
6 shapes.add('polygon')
7 print(shapes)
```

Duplicates Disappear in a Set

`add('polygon')`



PY main.py

```
1 shapes = set() # Create an empty set
2 shapes.add('triangle')
3 shapes.add('circle')
4 shapes.add('triangle')
5 print(shapes)
6 shapes.add('polygon')
7 print(shapes)
```

Removing an Element from a Set

PY main.py

```
1 shapes = {'square', 'triangle', 'circle', 'polygon'}  
2 shapes.remove('square')  
3 print(shapes)
```

[Download: lec_03_example_29.py](#)

Useful Set Operations

Operation	Python	Meaning
Subset	<code>s <= t</code>	Every element of s is in t
Superset	<code>s >= t</code>	Every element of t is in s
Union	<code>s t</code>	Elements from either set
Intersection	<code>s & t</code>	Elements common to both
Difference	<code>s - t</code>	Elements in s, but not t
Symmetric difference	<code>s ^ t</code>	Elements in exactly one set

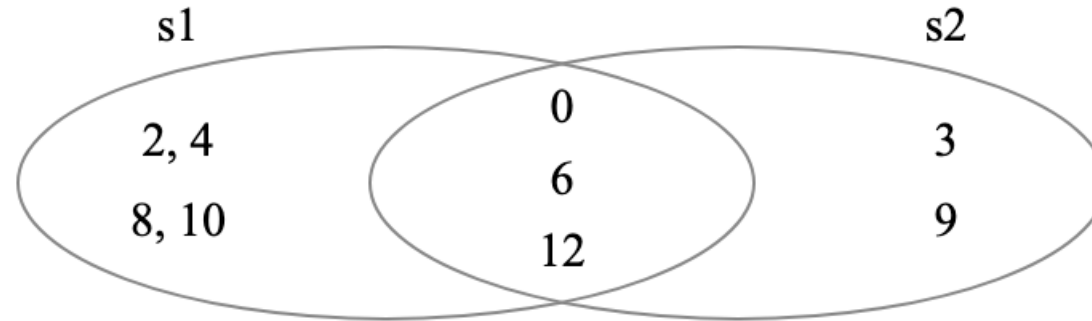
Intersection Finds Shared Elements

```
PY main.py
1 s1 = {0, 2, 4, 6, 8, 10, 12}
2 s2 = {0, 3, 6, 9, 12}
3 print(s1 & s2)
```

Download: lec_03_example_31.py

Intersection Finds Shared Elements

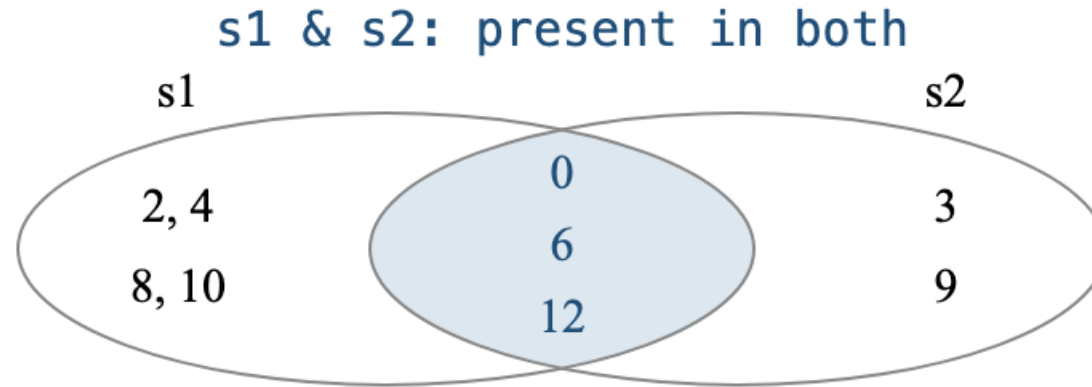
Compare membership in the two sets



PY main.py

```
1 s1 = {0, 2, 4, 6, 8, 10, 12}
2 s2 = {0, 3, 6, 9, 12}
3 print(s1 & s2)
```

Intersection Finds Shared Elements



PY main.py

```
1 s1 = {0, 2, 4, 6, 8, 10, 12}
2 s2 = {0, 3, 6, 9, 12}
3 print(s1 & s2)
```

While Loops

ENGG1330
Computer Programming I

A Loop Repeats Its Body

Body

The statements that repeat

A Loop Repeats Its Body

Body

The statements that repeat

Iteration

One repetition of the body

while Checks Before Each Iteration

```
PY main.py
1 count = 0
2 while count < 3:
3     print(count)
4     count += 1
5 print('Done')
```

Download: lec_03_example_35.py

while Checks Before Each Iteration

Condition

PY main.py

```
1 count = 0
2 while count < 3:
3     print(count)
4     count += 1
5 print('Done')
```

Download: [lec_03_example_35.py](#)

while Checks Before Each Iteration

Condition

Loop body

PY main.py

```
1 count = 0
2 while count < 3:
3     print(count)
4     count += 1
5 print('Done')
```

Download: [lec_03_example_35.py](#)

while Checks Before Each Iteration

Condition

Loop body

After the loop

PY main.py

```
1 count = 0
2 while count < 3:
3     print(count)
4     count += 1
5 print('Done')
```

Download: [lec_03_example_35.py](#)

Updating the Counter Ends This Loop

```
PY main.py
1 count = 0
2 while count < 3:
3     print('8')
4     count += 1
```

Download: [lec_03_example_36.py](#)

Updating the Counter Ends This Loop

Update

```
PY main.py
1 count = 0
2 while count < 3:
3     print('8')
4     count += 1
```

Download: [lec_03_example_36.py](#)

Repeat Until the Answer Is Correct

```
PY main.py
1 answer = 0
2
3 while answer != 4:
4     answer = int(input('2 * 2 = '))
5
6 print('Well done. Here is a cookie!')
```

Download: [lec_03_example_37.py](#)

Counting the Attempts

PY main.py

```
1 answer = 0
2 count = 0
3 while answer != 4:
4     answer = int(input('2 * 2 = '))
5     count += 1
6 print('Well done. Here is a cookie!')
7 print('It took you', count, 'attempts.')
```

A Loop Can Contain Another Loop

```
PY main.py
1 i = 0
2 while i < 2:
3     j = 0
4     while j < 2:
5         print(i, j)
6         j += 1
7     i += 1
```

Download: [lec_03_example_39.py](#)

A Loop Can Contain Another Loop

The inner loop runs for each i

i	j	Printed pair
0	0	0 0

PY main.py

```
1 i = 0
2 while i < 2:
3     j = 0
4     while j < 2:
5         print(i, j)
6         j += 1
7     i += 1
```

A Loop Can Contain Another Loop

Advance j

i	j	Printed pair
0	1	0 1

PY main.py

```
1 i = 0
2 while i < 2:
3     j = 0
4     while j < 2:
5         print(i, j)
6         j += 1
7     i += 1
```

A Loop Can Contain Another Loop

Advance i , then reset j to 0

i	j	Printed pair
1	0	1 0

```
PY main.py
1 i = 0
2 while i < 2:
3     j = 0
4     while j < 2:
5         print(i, j)
6         j += 1
7     i += 1
```

Download: [lec_03_example_39.py](#)

A Loop Can Contain Another Loop

Advance j again

i	j	Printed pair
1	1	1 1

PY main.py

```
1 i = 0
2 while i < 2:
3     j = 0
4     while j < 2:
5         print(i, j)
6         j += 1
7     i += 1
```

For Loops

ENGG1330
Computer Programming I

for Visits Each Element

```
PY main.py
1 primes = [2, 3, 5, 7]
2 for prime in primes:
3     print(prime)
4 print('Done')
```

Download: [lec_03_example_41.py](#)

for Visits Each Element

Next element

PY main.py

```
1 primes = [2, 3, 5, 7]
2 for prime in primes:
3     print(prime)
4 print('Done')
```

Download: [lec_03_example_41.py](#)

for Visits Each Element

Next element

Loop body

PY main.py

```
1 primes = [2, 3, 5, 7]
2 for prime in primes:
3     print(prime)
4 print('Done')
```

Download: [lec_03_example_41.py](#)

for Visits Each Element

Next element

Loop body

After the loop

PY main.py

```
1 primes = [2, 3, 5, 7]
2 for prime in primes:
3     print(prime)
4 print('Done')
```

Download: [lec_03_example_41.py](#)

A Loop Can Build a Running Total

```
PY main.py
1 | scores = (4, 7, 3)
2 | total = 0
3 | for score in scores:
4 |     total += score
5 |     print(total)
6 | print('Final:', total)
```

Download: [lec_03_example_42.py](#)

A Loop Can Build a Running Total

Start total at 0

score	total
Not read yet	0

PY main.py

```
1 scores = (4, 7, 3)
2 total = 0
3 for score in scores:
4     total += score
5     print(total)
6 print('Final:', total)
```

A Loop Can Build a Running Total

$$0 + 4 = 4$$

score	total
4	4

PY main.py

```
1 scores = (4, 7, 3)
2 total = 0
3 for score in scores:
4     total += score
5     print(total)
6 print('Final:', total)
```

A Loop Can Build a Running Total

$$4 + 7 = 11$$

score	total
7	11

PY main.py

```
1 | scores = (4, 7, 3)
2 | total = 0
3 | for score in scores:
4 |     total += score
5 |     print(total)
6 | print('Final:', total)
```

A Loop Can Build a Running Total

$$11 + 3 = 14$$

score	total
3	14

PY main.py

```
1 scores = (4, 7, 3)
2 total = 0
3 for score in scores:
4     total += score
5     print(total)
6 print('Final:', total)
```

A Loop Can Build a Running Total

The loop has finished

Final total

14

PY main.py

```
1 scores = (4, 7, 3)
2 total = 0
3 for score in scores:
4     total += score
5     print(total)
6 print('Final:', total)
```

Keeping the Largest Value Seen

PY main.py

```
1 scores = (4, 7, 3, 9, 6)
2 best = scores[0]
3 for score in scores:
4     if score > best:
5         best = score
6     print(score, best)
```

Keeping the Largest Value Seen

Start with the first score

score	best
Not read yet	4

PY main.py

```
1 scores = (4, 7, 3, 9, 6)
2 best = scores[0]
3 for score in scores:
4     if score > best:
5         best = score
6     print(score, best)
```

Keeping the Largest Value Seen

4 > 4 is false: keep best

score	best
4	4

PY main.py

```
1 scores = (4, 7, 3, 9, 6)
2 best = scores[0]
3 for score in scores:
4     if score > best:
5         best = score
6     print(score, best)
```

Keeping the Largest Value Seen

7 > 4 is true: update best

score	best
7	7

PY main.py

```
1 scores = (4, 7, 3, 9, 6)
2 best = scores[0]
3 for score in scores:
4     if score > best:
5         best = score
6     print(score, best)
```

Keeping the Largest Value Seen

3 > 7 is false: keep best

score	best
3	7

PY main.py

```
1 scores = (4, 7, 3, 9, 6)
2 best = scores[0]
3 for score in scores:
4     if score > best:
5         best = score
6     print(score, best)
```

Keeping the Largest Value Seen

9 > 7 is true: update best

score	best
9	9

PY main.py

```
1 | scores = (4, 7, 3, 9, 6)
2 | best = scores[0]
3 | for score in scores:
4 |     if score > best:
5 |         best = score
6 |     print(score, best)
```

Keeping the Largest Value Seen

6 > 9 is false: keep best

score	best
6	9

PY main.py

```
1 scores = (4, 7, 3, 9, 6)
2 best = scores[0]
3 for score in scores:
4     if score > best:
5         best = score
6     print(score, best)
```

Looping Through a String

```
PY main.py
1 for c in 'Dirk':
2     print(c)
```

Download: [lec_03_example_43.py](#)

Looping Through Dictionary Keys

PY main.py

```
1 mood = {'sunny': ':-) ', 'cloudy': ':-| ', 'rainy': ':-( '}  
2 for key in mood:  
3     print('Key:', key, 'Value:', mood[key])
```

[Download: lec_03_example_44.py](#)

range Stops Before Its Stop Value

```
main.py
1 for x in range(3, 8):
2     print('->', x)
```

Download: lec_03_example_45.py

range Stops Before Its Stop Value

`range(3, 8):` start included, stop excluded

3	4	5	6	7	8
3	4	5	6	7	Excluded

```
main.py
1 for x in range(3, 8):
2     print('->', x)
```

Download: lec_03_example_45.py

range Stops Before Its Stop Value

The first iteration

3	4	5	6	7	8
3	4	5	6	7	Excluded

PY main.py

```
1 for x in range(3, 8):  
2     print('->', x)
```

range Stops Before Its Stop Value

The last iteration

3	4	5	6	7	8
3	4	5	6	7	Excluded

PY main.py

```
1 for x in range(3, 8):  
2     print('->', x)
```

Nested for Loops

```
PY main.py
1 for i in range(2):
2     for j in range(2):
3         print(i, j)
```

Download: [lec_03_example_46.py](#)

Nested for Loops

First outer value, first inner value

i	j	Printed pair
0	0	0 0

PY main.py

```
1 for i in range(2):  
2     for j in range(2):  
3         print(i, j)
```

[Download: lec_03_example_46.py](#)

Nested for Loops

Advance the inner loop

i	j	Printed pair
0	1	0 1

PY main.py

```
1 for i in range(2):  
2     for j in range(2):  
3         print(i, j)
```

[Download: lec_03_example_46.py](#)

Nested for Loops

The next outer value starts a new inner loop

i	j	Printed pair
1	0	1 0

PY main.py

```
1 for i in range(2):  
2     for j in range(2):  
3         print(i, j)
```

[Download: lec_03_example_46.py](#)

Nested for Loops

Advance the inner loop again

i	j	Printed pair
1	1	1 1

PY main.py

```
1 for i in range(2):  
2     for j in range(2):  
3         print(i, j)
```

[Download: lec_03_example_46.py](#)

A Decision Inside a Loop

```
PY main.py
1 for i in range(1, 5):
2     if i % 2 == 0:
3         print(i)
4     else:
5         print('odd')
```

Download: lec_03_example_47.py

A Negative Step Counts Down

```
PY main.py
1 num = int(input('Number: '))
2 for i in range(num, 0, -1):
3     print(i)
4 print('Blast off!')
```

Download: [lec_03_example_48.py](#)

Using an Index in a Loop

```
PY main.py
1 fruits = ('apple', 'orange', 'pear', 'banana')
2 for i in range(len(fruits)):
3     print(i, '=', fruits[i])
```

Download: [lec_03_example_49.py](#)

Using an Index in a Loop

```
i = 0: fruits[i] = 'apple'
```

0	1	2	3
apple	orange	pear	banana

PY main.py

```
1 fruits = ('apple', 'orange', 'pear', 'banana')
2 for i in range(len(fruits)):
3     print(i, '=', fruits[i])
```

[Download: lec_03_example_49.py](#)

Using an Index in a Loop

```
i = 1: fruits[i] = 'orange'
```

0	1	2	3
apple	orange	pear	banana

PY main.py

```
1 fruits = ('apple', 'orange', 'pear', 'banana')
2 for i in range(len(fruits)):
3     print(i, '=', fruits[i])
```

[Download: lec_03_example_49.py](#)

Using an Index in a Loop

```
i = 2: fruits[i] = 'pear'
```

0	1	2	3
apple	orange	pear	banana

PY main.py

```
1 fruits = ('apple', 'orange', 'pear', 'banana')
2 for i in range(len(fruits)):
3     print(i, '=', fruits[i])
```

[Download: lec_03_example_49.py](#)

Using an Index in a Loop

```
i = 3: fruits[i] = 'banana'
```

0	1	2	3
apple	orange	pear	banana

PY main.py

```
1 fruits = ('apple', 'orange', 'pear', 'banana')
2 for i in range(len(fruits)):
3     print(i, '=', fruits[i])
```

[Download: lec_03_example_49.py](#)

enumerate Gives the Index and Value

```
main.py
1 fruits = ('apple', 'orange', 'pear', 'banana')
2 for index, value in enumerate(fruits):
3     print(index, '=', value)
```

[Download: lec_03_example_50.py](#)

enumerate Gives the Index and Value

index = 0, value = 'apple'

0	1	2	3
apple	orange	pear	banana

PY main.py

```
1 fruits = ('apple', 'orange', 'pear', 'banana')
2 for index, value in enumerate(fruits):
3     print(index, '=', value)
```

[Download: lec_03_example_50.py](#)

enumerate Gives the Index and Value

index = 1, value = 'orange'

0	1	2	3
apple	orange	pear	banana

PY main.py

```
1 fruits = ('apple', 'orange', 'pear', 'banana')
2 for index, value in enumerate(fruits):
3     print(index, '=', value)
```

Download: lec_03_example_50.py

enumerate Gives the Index and Value

index = 2, value = 'pear'

0	1	2	3
apple	orange	pear	banana

PY main.py

```
1 fruits = ('apple', 'orange', 'pear', 'banana')
2 for index, value in enumerate(fruits):
3     print(index, '=', value)
```

Download: lec_03_example_50.py

enumerate Gives the Index and Value

index = 3, value = 'banana'

0	1	2	3
apple	orange	pear	banana

PY main.py

```
1 fruits = ('apple', 'orange', 'pear', 'banana')
2 for index, value in enumerate(fruits):
3     print(index, '=', value)
```

Download: [lec_03_example_50.py](#)

break Leaves the Loop

PY main.py

```
1- for val in 'string':  
2-     if val == 'i':  
3-         break  
4-     print(val)  
5- print('Done')
```

break Leaves the Loop

Leave the loop

PY main.py

```
1- for val in 'string':  
2-     if val == 'i':  
3-         break  
4-     print(val)  
5- print('Done')
```

continue Skips to the Next Iteration

```
PY main.py
1- for val in 'string':
2-     if val == 'i':
3-         continue
4-     print(val)
5- print('Done')
```

Download: [lec_03_example_52.py](#)

continue Skips to the Next Iteration

Next iteration

PY main.py

```
1- for val in 'string':  
2-     if val == 'i':  
3-         continue  
4-     print(val)  
5- print('Done')
```

L-3 Survey

Complete the full survey during the lecture

Coding Challenge

To Earn Your Participation Mark

- Complete the full survey during the lecture
- Pass all tests in **at least 3 of the 5 challenges** by **23:59 HKT today**

Both required · 1 mark · No partial credit