

Attendance Assignment

```
Microsoft Windows [Version 10.0.18363.1082]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\dillk>python
Python 3.8.6rc1 (tags/v3.8.6rc1:08bd63d, Sep  7 2020, 23:10:23) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> S='Santa Clara University'
>>> S[0:5]
'Santa'
>>> S[0:11]
'Santa Clara'
>>> S[-1]
'y'
>>> S[-2]
't'
>>> S[12:]
'University'
>>>
```

Exercise 1:

```
>>> a = 123 + 222
>>> print(a)
345
>>> b = 1.5 * 4
>>> print(b)
6.0
>>> c = 2**10
>>> print(c)
1024
>>> import math
>>> print(math.pi)
3.141592653589793

>>> print(math.sqrt(36))
6.0
>>> import random
>>> a = random.random()
>>> print('a=', a)
a= 0.09905317311279294
>>> b = random.choice([1,2,3,4])
>>> print('b=', b)
b= 4
```

(Image is cut because I made an error)

Exercise 2:

```
>>>
>>> S='Spam'
>>> len(S)
4

>>> S[0]
'S'
>>> S[1]
'p'
>>> S[-1]
'm'
>>> S[-2]
'a'
>>> S[len(S) - 1]
'm'
>>> S[1:3]
'pa'
>>> S = 'z' + S[1:]
>>> S
'zpam'
```

(Image is cut because I made an error)

Exercise 3:

```
>>> L = [123, 'spam', 1.23]
>>> len(L)
3
>>> L[0]
123
>>> L[: -1]
[123, 'spam']
>>> L + [456]
[123, 'spam', 1.23, 456]
>>> L*2
[123, 'spam', 1.23, 123, 'spam', 1.23]
>>> L
[123, 'spam', 1.23]
>>> M = ['bb', 'aa', 'cc']
>>> M.sort()
>>> M
['aa', 'bb', 'cc']
>>> M.reverse()
>>> M
['cc', 'bb', 'aa']
>>> M = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
>>> M[1]
[4, 5, 6]
>>> M[1][2]
6
>>> diag = [M[i][i] for i in [0, 1, 2]]
>>> diag
[1, 5, 9]
>>> doubles = [c * 2 for c in 'spam']
>>> doubles
['ss', 'pp', 'aa', 'mm']
>>> list(range(4))
[0, 1, 2, 3]
>>> list(range(-6, 7, 2))
[-6, -4, -2, 0, 2, 4, 6]
>>> [[x**2, x**3] for x in range(4)]
[[0, 0], [1, 1], [4, 8], [9, 27]]
>>> [[x, x/2, x*2] for x in range(-6, 7, 2) if x > 0]
[[2, 1.0, 4], [4, 2.0, 8], [6, 3.0, 12]]
```

Exercise 4:

```
>>> D = {'food': 'Spam', 'quantity': 4, 'color': 'pink'}
>>> D['food']
'Spam'
>>> D['quantity'] += 1
>>> D
{'food': 'Spam', 'quantity': 5, 'color': 'pink'}
>>> D = {}
>>> D['name'] = 'Bob'
>>> D['job'] = 'dev'
>>> D['age'] = 40
>>> D
{'name': 'Bob', 'job': 'dev', 'age': 40}
>>> print(D['name'])
Bob
```

```
>>> bob1 = dict(name='Bob', job = 'dev', age=40)
>>> bob1
{'name': 'Bob', 'job': 'dev', 'age': 40}
>>> bob2 = dict(zip(['name', 'job', 'age'], ['Bob', 'dev', 40]))
>>> bob2
{'name': 'Bob', 'job': 'dev', 'age': 40}
```

(Image is cut because I made an error)

Exercise 5:

```
>>> T = (1, 2, 3, 4)
>>> len(T)
4
>>> T + (5, 6)
(1, 2, 3, 4, 5, 6)
>>> T[0]
1
>>> T.index(4)
3
>>> T.count(4)
1
>>> T[0] = 2
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
>>> T = (2,) + T[1:]
>>> T
(2, 2, 3, 4)
>>> T = 'spam', 3.0, [11,22,33]
>>> T[1]
3.0
>>> T[2][1]
22
>>>
```

Exercise 6:

```
test_if.py x
Python > COEN 166 > test_if.py > x
1 x = 1
2 if x:
3     y = 2
4     if y:
5         print('block2')
6     print('block1')
7 print('block0')
8
9 choice = 'ham'
10 if choice == 'spam': # the equivalent if statement
11     print(1.25)
12 elif choice == 'ham':
13     print(1.99)
14 elif choice == 'eggs':
15     print(0.99)
16 elif choice == 'bacon':
17     print(1.10)
18 else:
19     print('Bad choice')
20

Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

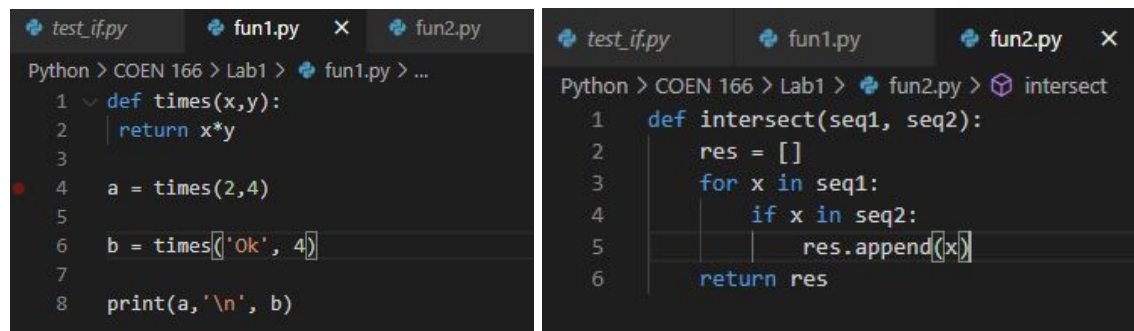
PS C:\Users\Dillon\Desktop\CodeStuff> py test_if.py
C:\Users\Dillon\AppData\Local\Programs\Python\Python38\python.exe: can't open file 'test_if.py': [Errno 2] No such file or directory
PS C:\Users\Dillon\Desktop\CodeStuff> cd Python
PS C:\Users\Dillon\Desktop\CodeStuff\Python> cd '.\COEN 166\'
PS C:\Users\Dillon\Desktop\CodeStuff\Python\COEN 166> py test_if.py
block2
block1
block0
1.99
PS C:\Users\Dillon\Desktop\CodeStuff\Python\COEN 166>
```

(done in visual studio code, which is why it looks a little different)

Exercise 7:

```
>>> x = 'spam'
>>> while x:
...     print(x, end='')
...     x = x[1:]
...
spampamamm>>>
>>> a = 0; b = 10
>>> while a < b:
...     print(a, end='')
...     a += 1
...
0123456789>>>
>>> x = 10
>>> while x:
...     x = x-1
...     if x%2 != 0: continue
...     print(x, end='')
...
86420>>>
>>> for x in ["spam", "eggs", "ham"]:
...     print(x, end='')
...
spameggsham>>>
>>> sum = 0
>>> for x in [1,2,3,4]:
...     sum = sum + x
...
>>> sum
10
>>> prod = 1
>>> for item in [1,2,3,4]: prod*=item
...
>>> prod
24
>>>
```

Exercise 8:



The image shows two side-by-side screenshots of a code editor. The left window, titled 'fun1.py', contains the following code:

```
1 def times(x,y):
2     return x*y
3
4 a = times(2,4)
5
6 b = times('Ok', 4)
7
8 print(a, '\n', b)
```

The right window, titled 'fun2.py', contains the following code:

```
1 def intersect(seq1, seq2):
2     res = []
3     for x in seq1:
4         if x in seq2:
5             res.append(x)
6     return res
```

```
>>> def times(x,y):
...     return x*y
...
>>> a = times(2,4)
>>>
>>> b = times('Ok', 4)
>>>
>>> b = times('Ok', 4)
>>>
>>> def times(x,y):
...     return x*y
...
>>> a = times(2,4)
>>> b = times('Ok', 4)
>>> print(a, '\n', b)
8
OkOkOkOk
>>> def intersect(seq1, seq2):
...     res = []
...     for x in seq1:
...         if x in seq2:
...             res.append(x)
...     return res
...
>>> s1 = "SPAM"
>>> s2 = "SCAM"
>>> result1 = intersect(s1, s2)
>>> print(result1)
['S', 'A', 'M']
>>> result2 = intersect([1,2,3], (1, 4))
>>> print(result2)
[1]
>>>
```

In fun1.py:

If the parameters are both integers, it simply returns the product.

If one is a number x and the other is a string, it returns a string where the string is printed out x times.

In fun2.py:

For every element in `seq1`, add that to the array `res` if that element is also in `seq2`.

Exercise 9:

```
Python > COEN166 > Lab1 > test1_module.py > ...
1 import module # Get module as a whole
2 result = module.adder(module.a, module.b) # Qualify to get names
3 print(result)
```

```
Python > COEN166 > Lab1 > test2_module.py > ...
1 c = 5
2 d = 10
3
4 from module import adder # Copy out an attribute
5 result = adder(c, d) # No need to qualify name
6 print(result)
7
8 from module import a, b, multiplier # Copy out multiple attributes
9 result = multiplier(a, b)
10 print(result)
```

```
Python > COEN166 > Lab1 > test3_module.py > ...
1 from module import * # Copy out all attributes
2 result1 = adder(a, b)
3 result2 = multiplier(a, b)
4 print(result1, '\n', result2)
```

```
PS C:\Users\Dillon\Desktop\CodeStuff\Python\COEN166\Lab1> py test1_module.py
30
PS C:\Users\Dillon\Desktop\CodeStuff\Python\COEN166\Lab1> py test2_module.py
15
200
PS C:\Users\Dillon\Desktop\CodeStuff\Python\COEN166\Lab1> py test3_module.py
30
200
PS C:\Users\Dillon\Desktop\CodeStuff\Python\COEN166\Lab1> 
```

Test1: It takes the variables (a and b) in module.py, uses those values as parameters in the function adder of module.py. Those values are added together and printed.

Test2: It initializes 2 variables. Then, it imports the adder function from module.py. It uses the function to add c and d, then prints the result. It also imports multiple attributes from module.py. The variables a and b are used as parameters in the multiplier function. The result is printed.

Test3: It imports all attributes of module.py. First it adds a and b together and stores that in the variable result1. Then, it multiplies a and b together and stores that in the variable result2. Then, it prints both out.

Exercise 10:

```
minmax.py X minmax2.py module.py test1_module.py test2_module.py
Python > COEN166 > Lab1 > minmax.py > minmax
1 def minmax(test,array):
2     res = array[0]
3     for arg in array[1:]:
4         if test(arg, res):
5             res = arg
6     return res
7 def lessthan(x,y): return x<y
8 def grtrthan(x,y): return x>y
9
10 print(minmax(lessthan, [4,2,1,5,6,3])) # self-test code
11 print(minmax(grtrthan, [4,2,1,5,6,3]))
```

```
minmax.py minmax2.py X module.py test1_module.py test2_module.py test3_module.py
Python > COEN166 > Lab1 > minmax2.py > ...
1 def minmax(test,array):
2     res = array[0]
3     for arg in array[1:]:
4         if test(arg, res):
5             res = arg
6     return res
7 def lessthan(x,y): return x<y
8 def grtrthan(x,y): return x>y
9
10 if __name__ == '__main__':
11     print(minmax(lessthan, [4,2,1,5,6,3])) # self-test code
12     print(minmax(grtrthan, [4,2,1,5,6,3]))
```

```
PS C:\Users\Dillon\Desktop\CodeStuff\Python\COEN166\Lab1> py minmax.py
1
6
PS C:\Users\Dillon\Desktop\CodeStuff\Python\COEN166\Lab1> 
```

The two print statements call the minmax function, which takes the lessthan function and an array as parameters. For every element in the array, if the element is less than res (initially set as the first array element), then set res as that element. After that, return res. Finally, the

self-test code will print that number out. The lowest number is 1. The second self test code works in the same way, except if the element is greater than res, res will have the value of the element. So, 6 will be printed out.

```
PS C:\Users\Dillon\Desktop\CodeStuff\Python\COEN166\Lab1> py minmax2.py
1
6
PS C:\Users\Dillon\Desktop\CodeStuff\Python\COEN166\Lab1> 
```

Minmax2.py works in the same way, except it checks to make sure if it is the top level programming file before actually calling the minmax function.

```
>>> import minmax
1
6
>>> import minmax2
>>> 
```

Calling these two files in python shell demonstrates a case where minmax2 will not print because it is not a top level programming file. Minmax will run because it doesn't care whether it is a top level programming file.

Exercise 11:

```
class1.py x class2.py minmax.py minmax2.py module.py test1_module.py test2_modul

Python > COEN166 > Lab1 > class1.py > ...
1 class FirstClass: # define a class object
2     def setdata(self,value1, value2): # Define class's methods
3         self.data1=value1 # self is the instance
4         self.data2=value2
5     def display(self):
6         print(self.data1, '\n', self.data2, '\n')
7
8 x=FirstClass() # make one instance
9 x.setdata("King Arthur",-5) # Call methods: self is x
10 x.display()
11 x.data1="QQ"
12 x.data2=-3
13 x.display()
14 x.anothername="spam"
15 x.display()
16 print(x.anothername)
```



```
class1.py class2.py X minmax.py minmax2.py module.py test1_module.py
Python > COEN166 > Lab1 > class2.py > ...
1 class FirstClass:
2     def setdata(self,value1, value2):
3         self.data1=value1
4         self.data2=value2
5     def display(self):
6         print(self.data1, '\n', self.data2, '\n')
7
8 class SecondClass(FirstClass): # inherits setdata and display
9     def adder(self,val1,val2): # create adder
10         a=val1+val2
11         print(a)
12
13
14 z=SecondClass() # make one instance
15 z.setdata(10,20)
16 z.display()
17 z.adder(-5,-10)
```

```
PS C:\Users\Dillon\Desktop\CodeStuff\Python\COEN166\Lab1> py class1.py
King Arthur
-5
QQ
-3
QQ
-3
spam
```

The code first creates a class that consists of two functions. Setdata takes two values and sets them as variables in the self object. Display just prints out data1 and data2.

First, x is created as an object of the class FirstClass. Then, we call set data to set x's variables as "King Arthur" and -5. Then, display() is called and this data is printed. Then, we directly access both variables of x and change them to "QQ" and -3 respectively. We print this data again. We then add a third variable to object x and call it anothername. When we call display, it only prints the original two variables we added earlier. So, we have to print x.anothername if we want to see it.

```
PS C:\Users\Dillon\Desktop\CodeStuff\Python\COEN166\Lab1> py class2.py
10
20
-15
PS C:\Users\Dillon\Desktop\CodeStuff\Python\COEN166\Lab1> 
```

We see the definition of the First Class class, then we see a new class that inherits the FirstClass. This means that all of the functions and variables defined in the class are also present in SecondClass.

First, we create z, an instance of SecondClass. We call setdata, which is a FirstClass function. However, this works because SecondClass inherited that function when it was defined. So, z's data1 and data2 variables are set to 10 and 20 respectively. Next, we see a display() function, which also works despite being a FirstClass function for the same reason. We see 10 and 20 pop up on the terminal. Finally, we see adder() called, which prints out the sum of val1 and val2. In this case, Val1 = -5 and Val2 = -10. We see -15 on the terminal.