

|

## Activity 1.1

---

|

## Python Challenge

---

|

|---| ---|

Name:

Course and Section:

Date Submitted:

Instructor:

Date Performed:

Date Submitted

---

**Instructions:** Answer the questions or complete the tasks outlined below, use the specific method described if applicable. Insert a code cell after each text cell.

**Objectives:**

- Test understanding of Python basics
  - Review Python basic to intermediate commands
- 

1. What is 3 to the power of 5?

```
In [17]: print(3**5)
```

243

2. Create a variable, `s`, containing the string "This course is amazing!". Using the variable, split the string into a list.

```
In [18]: s = "Imagine dragons!"  
s.split()
```

```
Out[18]: ['Imagine', 'dragons!']
```

3. Given the variables `height` and `mountain`, use `.format()` to print the following string:  
The height of Mt. Everest is 8848 meters.

```
In [19]: print("The heigh of {} is {} meters".format("Mt. Everest", 8848))
```

The heigh of Mt. Everest is 8848 meters

4. Given the following nested list, use indexing to grab the word "this".

```
lst = ['a','b',[4,10,11],['c',[1,66,['this']],2,111], 'e',7]
```

```
In [20]: lst = ['a','b',[4,10,11],['c',[1,66,['this']],2,111], 'e',7]  
print(lst[3][1][2])
```

['this']

5. Given the following nested dictionary, grab the word "that". This exercise is a little more difficult.

```
d = {'k1':['val1','val2','val3',{'we':['need','to','go',{'deeper':[1,2,3,'that']}]}]}
```

```
In [21]: d = {'k1':['val1','val2','val3',{'we':['need','to','go',{'deeper':  
[1,2,3,'that']}]}]}  
nested_list_item = d['k1'][3]['we'][3]['deeper'][3]  
print(nested_list_item)
```

that

6. Create a function, `GetDomain()`, that grabs the email website domain from a string in the form: `user@domain.com`.

So for example, passing `"user@domain.com"` would return: `domain.com`

```
In [22]: def GetDomain(gmail):
          at_find = gmail.find('@')
          if at_find == -1:
              return "No gmail found"

          head_start = gmail.rfind(' ', 0, at_find)
          head = gmail[head_start + 1 : at_find]
          tail_end = gmail.find(' ', at_find)

          if tail_end == -1:
              tail = gmail[at_find + 1:]
          else:
              tail = gmail[at_find + 1 : tail_end]

          return head + '@' + tail

          GetDomain("LAKWDJLAKWJDLAKWJDLAKWJ DALWKDJ ALWK DJALWKJ DLAKWDNAW DNLAKW
          DJLAWKJ DLAWK DJLKJA ImagineDragons@Gmail.com akljdalwkjdlkawjdlkawj")
```

```
Out[22]: 'ImagineDragons@Gmail.com'
```

7. Create a basic function, `findInternet()`, that returns `True` if the word 'Internet' is contained in the input string. Don't worry about edge cases like punctuation being attached to the word, but account for capitalization. (Hint: Please see <https://docs.python.org/2/reference/expressions.html#in>)

```
In [23]: import re
          def findInternet(string):
              find = "Internet"
              if re.search(find, string, re.IGNORECASE):
                  print("Found Internet!")
              else:
                  print("Wasn't found :<")

          findInternet("OPAJNDLKAWNDLKAWN LKnknlkNDLKNASLWDKNAW LKnd alkdn
          KNLAKWNDL KAWinterNet")
```

```
Found Internet!
```

8. Create a function, `countIoT()`, that counts the number of times the word "IoT" occurs in a string. Ignore edge cases but take into account capitalization.

```
In [24]: import re

def countIOT(text):
    matches = re.findall(r"IoT", text, re.IGNORECASE)
    return len(matches)

text = "OPAJNDLKAWNDLKAWN LKknklkNDLKNASLWDKNiot IOT iot iotOIT
ITOITOITOTITOITOITOITOIT O OIT OITITIO IOTAW LKnd alkdn KNLAKWNDL
KAWinterNet"

count = countIOT(text)
print(count)
```

6

9. Use lambda expressions and the filter() function to filter out words from a list that do not start with the letter 'd'. For example:

```
seq = ['data','salt','dairy','cat','dog']
```

should be filtered down to:

```
['data','dairy','dog']
```

```
In [25]: seq = ['data','salt','dairy','cat','dog']
```

10. Use lambda expressions and the map() function to convert a list of words to upper case. For example:

```
seq = ['data','salt','dairy','cat','dog']
```

should become:

```
['DATA','SALT','DAIRY','CAT','DOG']
```

11. Imagine a smart thermostat that is connected to the door, so that it can detect, in addition to the temperature, when people enter or leave the house.

Write a function that, if the temperature is lower than 20 degrees Celsius, and there are people in the house (encoded as a boolean value to be passed as a parameter to the function), turns on the heating by returning the string "Heating on". When the temperature reaches 23 degrees or there are no people in the house, it returns the string "Heating off". When none of these conditions are met, the function returns "Do nothing".

12. The function `zip(list1, list2)` returns a list of tuples, where the *i*-th tuple contains the *i*-th element from each of the argument lists. Use the `zip` function to create the following list of tuples:

```
zipped = [('Parking', -1), ('Shops', 0), ('Food Court', 1), ('Offices', 2)]
```

```
In [26]: # Code cell 13
         floor_types = ['Parking', 'Shops', 'Food Court', 'Offices']
         floor_numbers = range(-1, 3)
         #zipped = list(...)
         print(zipped)
```

```
-----
NameError                                Traceback (most
recent call last)
Cell In[26], line 5
      3 floor_numbers = range(-1, 3)
      4 #zipped = list(...)
----> 5 print(zipped)

NameError: name 'zipped' is not defined
```

13. Use the `zip` function and `dict()` to create a dictionary, `elevator_dict`, where the keys are the floor types and the values are the corresponding floor number so that:

```
elevator_dict[-1] = 'Parking'
```

```
In [ ]: # Code cell 14
        floor_types = ['Parking', 'Shops', 'Food Court', 'Offices']
        floors_numbers = range(-1,3)
        #elevator_dict = dict(...)
        print(elevator_dict)
```

```
In [ ]: # Code cell 15
        # Verify elevator_dict[-1]
        elevator_dict[-1]
```

14. Create an Elevator class. The constructor accepts the list of strings `floor_types` and the list of integers `floor_numbers`. The class implements the methods `'ask_which_floor'` and `'go_to_floor'`. The output of this methods should look as follows:

```
\
floor_types = ['Parking', 'Shops', 'Food Court', 'Offices']
floors_numbers = range(-1,4)
el = Elevator(floor_numbers, floor_types)
el.go_to_floor(1)
Going to Food Court floor!
el.go_to_floor(-2)
There is floor number -2 in this building.
el.ask_which_floor('Offices')
The floor Offices is the number: 2
el.ask_which_floor('Swimming Pool')
There is no Swimming Pool floor in this building.
\
```

**Good Job!**