

# Midterm 3 Study Guide

CSC 110

## Dictionary Loop Tables

### Loop table 1

Give the function below:

```
def keys_to_string(dictionary):  
    new_string = ""  
    for key in dictionary:  
        new_string += key  
    return new_string
```

Complete the loop table below with the corresponding values of `key`, and `new_string` for each loop interaction the following function call:

```
keys_to_string({"b": 0, "a": 1, "n": 1})
```

RESPONSE FOR LOOP 1:

key	.....new_string.....

## Loop table 2

Give the function below:

```
def values_total(dictionary):  
    total = 0  
    for value in dictionary.values():  
        total += value  
    return total
```

Complete the loop table below with the corresponding values of `value`, and `new_string` for each loop interaction the following function call:

```
values_total({"b": 0, "a": 1, "n": 1, "d": 3, "c": 3, "e": 2})
```

RESPONSE FOR LOOP 2:

value	total

### Loop table 3

Give the function below:

```
def difference_list(dictionary):  
    result = []  
    for key, value in dictionary.items():  
        result.append(key - value)  
    return result
```

Complete the loop table below with the corresponding values of **key**, **value**, and ‘**result**’ for each loop interaction the following function call:

```
difference_list({5: 0, 4: 1, 3: 1, 6: 4})
```

RESPONSE FOR LOOP 3:

key	value	.....result.....

## Loop table 4

Give the function below:

```
def make_set(dictionary):  
    result = set()  
    for key, value in dictionary.items():  
        result.add(key)  
        result.add(value)  
    return result
```

Complete the loop table below with the corresponding values of `key`, `value`, and `result` for each loop interaction the following function call:

```
make_set({5: 0, 4: 1, 3: 1, 6: 4})
```

RESPONSE FOR LOOP 4:

key	value	.....result.....

## Conditionals

### Evaluate the code

Evaluate the code below (when the code throws an error, the answer should be **ERROR**)

```
x = 4
if x < 0:
    x += 1
else:
    x *= 10

x # what's the value of x?
```

```
x = 10
if x > 10 and x < 0:
    x = 20
x # what's the value of x?
```

```
x = 10
if x < 10 or x > 0:
    x = 20
x # what's the value of x?
```

## For loops

### Evaluate the code

Evaluate the code below (when the code throws an error, the answer should be **ERROR**)

```
my_dict = {2: 0, 3: 1, 1: 0}
for key in my_dict:
    my_dict.pop(key)
my_dict
```

```
my_dict = {2: 0, 3: 1, 1: 0}
my_set = {2, 3}
for value in my_set:
    my_dict.pop(value)
my_dict
```

```
my_set = {2, 1, 4, 5, 7, 10, 23, 44}
for value in my_set:
    my_set.discard(value)
my_set
```

```
names = ["Ann", "Philipp", "Beatrice", "Paul"]
double_letter = []
for name in names:
    for i in range(len(name)-1):
```

```
    if name[i] == name[i+1]:
        double_letter.append(name)
double_letter
```

```
names = ["Ann", "Philipp", "Beatrice", "Paul"]
two_vowels = []
for name in names:
    for i in range(len(name)-1):
        if name[i] in "aeiou" and name[i+1] in "aeiou":
            two_vowels.append(name)
two_vowels
```

## Lists

### Evaluate the code

Evaluate the code below (when the code throws an error, the answer should be ERROR)

```
numbers = [1, 2]
numbers.append(5)
numbers.append(0)
numbers
```

```
numbers = [1, 2]
numbers.remove(1)
numbers
```

```
numbers = [1, 2]
numbers.pop(1)
numbers
```

```
numbers = [1, 2]
numbers[1]
numbers
```

```
numbers = [2, 3, 20, 1, 2, 40, 2]
numbers[7]
```

```
numbers = [2, 3, 20]
numbers[3] = 10
numbers
```

```
numbers = [2, 3, 20]
numbers.insert(0, 10)
numbers
```

## Dictionaries

### Evaluate the code

Evaluate the code below (when the code throws an error, the answer should be ERROR)

```
counts = {"a": 1, "b": 2}
counts[1] = "c"
counts
```

```
counts = {"a": 1, "b": 2}
counts["a"] = "c"
counts
```

```
counts = {"a": 1}
counts.append("b": 2)
counts
```

## Tuples

### Evaluate the code

Evaluate the code below (when the code throws an error, the answer should be **ERROR**)

```
names = ("Ann", "Philipp", "Beatrice", "Paul")
names[3]
```

```
names = ("Ann", "Philipp", "Beatrice", "Paul")
names[4]
```

```
names = ("Ann", "Philipp", "Beatrice", "Paul")
names[1]
```

## Files

### Evaluate the code

Evaluate the code below (when the code throws an error, the answer should be **ERROR**)

*names.txt*

```
Marcellin Burke
Albrecht Libby
Niki Zoey
Braxton Marvin
Marvin Brown
Ann Brown
```

```
f = open("names.txt", "r")
f.write("Ann\n")
f.close()
```

```
f = open("names.txt", "r")
names = []
for line in f:
    names.append(line)
f.close()
names
```

```
f = open("names.txt", "r")
names = []
for line in f:
    name = line.strip()
    names.append(name)
f.close()
names
```

```
f = open("names.txt", "r")
names = []
for line in f:
    line_names = line.strip().split(" ")
    for n in line_names:
        names.append(n)
f.close()
names
```

```
f = open("names.txt", "r")
names = []
for line in f:
    line_names = line.split(" ")
    for n in line_names:
        names.append(n)
f.close()
names
```

```
f = open("names.txt", "r")
names = {}
for line in f:
    line_names = line.strip().split(" ")
    for n in line_names:
        if n in names:
            names[n] += 1
        else:
            names[n] = 1
f.close()
names
```

## Writing code

### Strings

Write a python function named  `censor_vowels`  that takes a string as argument and returns a string that instead of vowels has `*`. (strings are not mutable)



Test cases:

```
assert censor_vowels("banana") == "b*n*n*"
assert censor_vowels("shirt") == "sh*rt"
```

## Lists

Write a python function named `replace_vowels` that takes a list of strings of length one as argument, and **mutates** the argument list by replacing the vowels with `*`.

Test cases:

```
original_list = ["b", "a", "n", "a", "n", "a"]
replace_vowels(original_list)
assert original_list == ["b", "*", "n", "*", "n", "*"]
assert replace_vowels(["s", "h", "i", "r", "t"]) == ["s", "h", "*", "r", "t"]
```

## KEY

## Dictionary Loop Tables

### Loop table 1

Give the function below:

```
def keys_to_string(dictionary):
    new_string = ""
    for key in dictionary:
        new_string += key
    return new_string
```

Complete the loop table below with the corresponding values of `key`, and `new_string` for each loop interaction the following function call:

```
keys_to_string({"b": 0, "a": 1, "n": 1})
```

RESPONSE FOR LOOP 1:

key	.....new_string.....
b	b
a	ba
n	ban

## Loop table 2

Give the function below:

```
def values_total(dictionary):  
    total = 0  
    for value in dictionary.values():  
        total += value  
    return total
```

Complete the loop table below with the corresponding values of `value`, and `new_string` for each loop interaction the following function call:

```
values_total({"b": 0, "a": 1, "n": 1, "d": 3, "c": 3, "e": 2})
```

RESPONSE FOR LOOP 2:

value	total
0	0
1	1
1	2
3	5
3	8
2	10

### Loop table 3

Give the function below:

```
def difference_list(dictionary):  
    result = []  
    for key, value in dictionary.items():  
        result.append(key - value)  
    return result
```

Complete the loop table below with the corresponding values of **key**, **value**, and **result** for each loop interaction the following function call:

```
difference_list({5: 0, 4: 1, 3: 1, 6: 4})
```

RESPONSE FOR LOOP 3:

key	value	.....result.....
5	0	[5]
4	1	[5, 3]
3	1	[5, 3, 2]
6	4	[5, 3, 2, 2]

## Loop table 4

Give the function below:

```
def make_set(dictionary):  
    result = set()  
    for key, value in dictionary.items():  
        result.add(key)  
        result.add(value)  
    return result
```

Complete the loop table below with the corresponding values of `key`, `value`, and `result` for each loop interaction the following function call:

```
make_set({5: 0, 4: 1, 3: 1, 6: 4})
```

RESPONSE FOR LOOP 4:

key	value	.....result.....
5	0	{5, 0}
4	1	{5, 4, 0, 1}
3	1	{5, 4, 0, 1, 3}
6	4	{5, 4, 0, 1, 3, 6}

## Conditionals

### Evaluate the code

Evaluate the code below (when the code throws an error, the answer should be ERROR)

```
x = 4
if x < 0:
    x += 1
else:
    x *= 10

x # what's the value of x?
```

## 40

```
x = 10
if x > 10 and x < 0:
    x = 20
x # what's the value of x?
```

## 10

```
x = 10
if x < 10 or x > 0:
    x = 20
x # what's the value of x?
```

## 20

## For loops

### Evaluate the code

Evaluate the code below (when the code throws an error, the answer should be ERROR)

```
my_dict = {2: 0, 3: 1, 1: 0}
for key in my_dict:
    my_dict.pop(key)
my_dict
```

ERROR

```
my_dict = {2: 0, 3: 1, 1: 0}
my_set = {2, 3}
for value in my_set:
    my_dict.pop(value)
```

## 0

## 1

```
my_dict
```

```
## {1: 0}
```

```
my_set = {2, 1, 4, 5, 7, 10, 23, 44}
for value in my_set:
    my_set.discard(value)
my_set
```

```
ERROR
```

```
names = ["Ann", "Philipp", "Beatrice", "Paul"]
double_letter = []
for name in names:
    for i in range(len(name)-1):
        if name[i] == name[i+1]:
            double_letter.append(name)
double_letter
```

```
## ['Ann', 'Philipp']
```

```
names = ["Ann", "Philipp", "Beatrice", "Paul"]
two_vowels = []
for name in names:
    for i in range(len(name)-1):
        if name[i] in "aeiou" and name[i+1] in "aeiou":
            two_vowels.append(name)
two_vowels
```

```
## ['Beatrice', 'Paul']
```

## Lists

### Evaluate the code

Evaluate the code below (when the code throws an error, the answer should be **ERROR**)

```
numbers = [1, 2]
numbers.append(5)
numbers.append(0)
numbers
```

```
## [1, 2, 5, 0]
```

```
numbers = [1, 2]
numbers.remove(1)
numbers
```

```
## [2]
```

```
numbers = [1, 2]
numbers.pop(1)
```

```
## 2
```

```
numbers
```

```
## [1]
```

```
numbers = [1, 2]
numbers[1]
```

```
## 2
```

```
numbers
```

```
## [1, 2]
```

```
numbers = [2, 3, 20, 1, 2, 40, 2]
numbers[7]
```

```
ERROR
```

```
numbers = [2, 3, 20]
numbers[3] = 10
numbers
```

```
ERROR
```

```
numbers = [2, 3, 20]
numbers.insert(0, 1)
numbers
```

```
## [1, 2, 3, 20]
```

## Dictionaries

### Evaluate the code

Evaluate the code below (when the code throws an error, the answer should be **ERROR**)

```
counts = {"a": 1, "b": 2}
counts[1] = "c"
counts
```

```
## {'a': 1, 'b': 2, 1: 'c'}
```

```
counts = {"a": 1, "b": 2}
counts["a"] = "c"
counts
```

```
## {'a': 'c', 'b': 2}
```

```
counts = {"a": 1}
counts.append("b": 2)
counts
```

ERROR

## Tuples

### Evaluate the code

Evaluate the code below (when the code throws an error, the answer should be **ERROR**)

```
names = ("Ann", "Philipp", "Beatrice", "Paul")
names[3]
```

```
## 'Paul'
```

```
names = ("Ann", "Philipp", "Beatrice", "Paul")
names[4]
```

ERROR

```
names = ("Ann", "Philipp", "Beatrice", "Paul")
names[1]
```

```
## 'Philipp'
```

## Files

### Evaluate the code

Evaluate the code below (when the code throws an error, the answer should be **ERROR**)

*names.txt*

```
Marcellin Burke
Albrecht Libby
Niki Zoey
Braxton Marvin
Marvin Brown
Ann Brown
```



```
f = open("names.txt", "r")
f.write("Ann\n")
f.close()
```

ERROR

```
f = open("names.txt", "r")
names = []
for line in f:
    names.append(line)
f.close()
names
```

```
## ['Marcellin Burke\n', 'Albrecht Libby\n', 'Niki Zoey\n', 'Braxton Marvin\n', 'Marvin Brown\n', 'Ann Brown\n']
```

```
f = open("names.txt", "r")
names = []
for line in f:
    name = line.strip()
    names.append(name)
f.close()
names
```

```
## ['Marcellin Burke', 'Albrecht Libby', 'Niki Zoey', 'Braxton Marvin', 'Marvin Brown', 'Ann Brown']
```

```
f = open("names.txt", "r")
names = []
for line in f:
    line_names = line.strip().split(" ")
    for n in line_names:
        names.append(n)
f.close()
names
```

```
## ['Marcellin', 'Burke', 'Albrecht', 'Libby', 'Niki', 'Zoey', 'Braxton', 'Marvin', 'Marvin', 'Brown', 'Ann']
```

```
f = open("names.txt", "r")
names = []
for line in f:
    line_names = line.split(" ")
    for n in line_names:
        names.append(n)
f.close()
names
```

```
## ['Marcellin', 'Burke\n', 'Albrecht', 'Libby\n', 'Niki', 'Zoey\n', 'Braxton', 'Marvin\n', 'Marvin', 'Brown', 'Ann']
```

```
f = open("names.txt", "r")
names = {}
for line in f:
```

```

line_names = line.strip().split(" ")
for n in line_names:
    if n in names:
        names[n] += 1
    else:
        names[n] = 1
f.close()
names

```

```
## {'Marcellin': 1, 'Burke': 1, 'Albrecht': 1, 'Libby': 1, 'Niki': 1, 'Zoey': 1, 'Braxton': 1, 'Marvin': 1}
```

## Writing code

### Strings

Write a python function named `sensor_vowels` that takes a string as argument and returns a string that instead of vowels has \*. (strings are not mutable)

```

def sensor_vowels(string):
    new_string = ""
    for char in string:
        if char in "aeiou":
            new_string += "*"
        else:
            new_string += char
    return new_string

def main():
    assert sensor_vowels("banana") == "b*n*n*"
    assert sensor_vowels("shirt") == "sh*rt"

main()

```

### Lists

Write a python function named `replace_vowels` that takes a list of strings of length one as argument, and **mutates** the argument list by replacing the vowels with \*.

```

def replace_vowels(character_list):
    for i in range(len(character_list)):
        if character_list[i] in "aeiou":
            character_list[i] = "*"
    return character_list

def main():
    original_list = ["b", "a", "n", "a", "n", "a"]
    replace_vowels(original_list)
    assert original_list == ["b", "*", "n", "*", "n", "*"]
    assert replace_vowels(["s", "h", "i", "r", "t"]) == ["s", "h", "*", "r", "t"]

main()

```