# Adv. Strings

David H Smith IV

University of Illinois Urbana-Champaign

Tues, Oct 4 2021

# Reminders

# Reminders

# Strings Slicing

# Poll Question: Slicing

What is the result of running this code?

```
1 my_str = "CS 105"
2 print(my_str[1:2])
```

A. 'C'
B. 'CS'
C. 'CS '
D. 'S'
E. 'S '

# Poll Question: Slicing

What is the result of running this code?

```
1 my_str = "CS 105"
2 print(my_str[-4:-2])
```

A. 'S 1'

B. 'S 10'

C. ' 1'

D. ' 10'

# Poll Question: Slicing

What is the result of running this code?

```
1 my_str = "CS 105"
2 print(my_str([::2]))
```

A. 'C'

B. 'CS'

C. 'S'

D. 'C 0'

# Slicing

A. `string[start:stop:interval]`
B. Like range, start is inclusive stop is exclusive.
C. Interval default is 1
D. Interval is optional

# Split

# Poll Question: Splitting

What is the result of running this code?

```
1 my_str = "CS 105 rox"
2 result = my_str.split()
```

A. `("CS", "105", "rox")`

B. `["CS 105 rox"]`

C. `["CS", "105 rox"]`

D. `["CS", "105", "rox"]`

# Poll Question: Splitting

What is the result of running this code?

```
1 csv = "1, 2, 3, 4"
2 result = csv.split(",")
```

A. ['1']
B. ['1, 2, 3, 4']
C. ['1', '2', '3', '4']
D. ['1,', '2,', '3,', '4']

# Join

# Poll Question: Joining

What is the result of running this code?

```
1 numlist = [1, 2, 3, 4]
2 result = ",".join(numlist)
```

A. '1234'

B. '1,2,3,4'

C. '1, 2, 3, 4

D. TypeError

# A Common Pattern

The generic pattern:

```
mylist = input_data.split(<separator>)
... data processing ...
outputstring "<separator>".join(my_list)
```

An example of this being done on one line:

```
output = ",".join(input.split(",")[::2])
```

# Pattern Practice

Write some code that takes a string with comma separated integers that converts the string into the square of each original value.

```
numlist = "1, 2, 3, 4"

#your code here
```

# Pattern Practice

```
numlist = "1, 2, 3, 4"

squaredlist = []
for num in numlist.split(","):
  squaredlist.append(int(num) ** 2)
squared_csv = ",".join(squredlist)
```

# General Loop Practice

# Task: Validate User Input

**Problem Statement:** Create a function that gets 10 words that contain the letter "e", stores them in a list, then returns them. Note that this problems uses nested loops but not break or enumerate.

# Task: Validate User Input

**Problem Statement:** Create a function that gets 10 words that contain the letter "e", stores them in a list, then returns them. Note that this problems uses nested loops but not break or enumerate.

```
1 def no_e():
2   l = []
3   for i in range(0, 10):
4     word = input("Enter a word with the letter e: ")
5     while "e" not in word:
6       word = input("Enter a word with the letter e: ")
7     l.append(word)
8   return l
```

# Task: Validate User Input

**Problem Statement:** Create a function that keeps asking the user for strings of an even length and adding them to a list until the user enters a string of an odd length. Then return the final list. You'll want to use a "while True:" loop here.

# Task: Validate User Input

**Problem Statement:** Create a function that keeps asking the user for strings of an even length and adding them to a list until the user enters a string of an odd length. Then return the final list. You'll want to use a "while True:" loop here.

```
1 def get_even_words():
2   l = []
3   while True:
4     user_in = input("Enter a word with an even number of vowels: ")
5     if len(user_in) % 2 != 0:
6       print("That word has an odd number of letters. Terminating!!")
7       break
8     l.append(user_in)
```