INTRODUCTION TO COMPUTER PROGRAMMING IN ENGINEERING AND SCIENCE

Lee wilkins

Office : 6C.9

Contact: MIO

Today's Topics:

- Review functions
- If / else / and / not / in
- Review string methods

Comp Sci assignments (All tests and assignments occur Wednesdays in our lab block)

Test 1 week 7 (15%) Wednesday March 6)

Assignment 2 week 8 (10%)

Assignment 3 week 11 (10%)

Test 2 week 13 (15%)

Physics assignments

Assignments (4 x 2%) 8% Date communicated by the Physics teacher

Project 1: Solving differential equations 10% Week 11

Project 2: Applying programming in science 22% Week 15

GRADE BREAKDOWN REVIEW

For: Loops through a range of items, or for each item in a set of items

While: Loops while a condition is true (be careful!)

LOOPS

While loops occur while a condition is true

```
While condition is true :
    #do task
```

In this example, the loop will run until count is greater than 4

```
i = 1
while i <= 10:
    print(i)
    i += 1
```

LOOPS: WHILE

It is possible to create a loop that never stops, in fact, its pretty easy to do, so be careful.

```python
while True:
    print("oopsie")
```

Crtl + c to stop it!

CAREFUL: INFINITE WHILE LOOPS!!

While loops can happen until the condition is no longer true OR there is a break.

Here, we are asking a question until

```
secret_number = 42
while True:
    user_guess = int(input("Guess a number between 1 and 100: "))
    if user_guess == secret_number:
        print(" Congratulations! You guessed the correct number.")
        break
    else:
        print("Try again!")
```

LOOPS: WHILE

For / in loop can be used to iterate through each item in a list or in a range.

```
for item in a_list:
    #do task
```

In this example, item is the name we are calling the individual item. a_list is a reference to an actual list.

```
fruits = ['apple', 'banana', 'cherry']
for fruit in fruits:
    print(len(fruit))
```

LOOPS: FOR IN

Before we can really see the power of for loops, we need to talk about lists.

Lists are a way of storing many things in a single variable. You can access them like we do with string indexes, remembering 0 is the first item in a list. A list can be many o

```
my_string_list = ["apple", "oranges", "bananas"]
print(my_string_list[0]) # apples
my_int_list = [2, 3, 10]
print(my_int_list[1]) #3
my_float_list = [2.4, 502.4, 2.5]
print(my_float_list[2]) #2.5
my_list_list = [ [1,4,5], [3,5,4], [4,2,5]]
print(my_list_list[1][1]) #5
```

LISTS

Lists can contain multiple data types. List is the entire structure (with its own methods) and each item can be accessed and on its own. Its important to pay attention to data types if your list is like this!

```
my_mixed_list = [2.4, "502.4", 2]
print(type(my_mixed_list)) # <class 'list'>
print(type(my_mixed_list[1])) # <class ‘str'>
```

LISTS

Similar to strings, there are list methods. You can find them here https://www.w3schools.com/python/python_ref_list.asp

List methods can only be performed on variables that have the data type list.

```
fruits = ['apple', 'banana', 'cherry']
print(fruits) # ['apple', 'banana', 'cherry']
fruits.reverse()
print(fruits)# ['cherry', 'banana', 'apple']
```

LISTS METHODS

Append will add a new item to the list (to the end)

```
new_fruit = input("What is another fruit?")
fruits.append(new_fruit)
print(fruits)
```

LISTS: APPEND

We can turn a string into a list if it has a delineator (something to separate the items of a list.

```
# Convert a string into a list
my_string_to_convert = "apples, oranges, bananas"
print(my_string_to_convert) # apples, oranges, bananas
my_string_to_convert = my_string_to_convert.split(“,") # ['apples', '
oranges', ' bananas’]
print(my_string_to_convert)
```

This string is split by the comma, but it has spaces! We can handle this two ways:

- Make a string that has no spaces (“apples, oranges, bananas”) or trip the white space with replace

```
my_string_to_convert = my_string_to_convert.replace(" ", "")
print(my_string_to_convert)
```

STRING TO LIST

Delineators can be anything! “this|is|my|string” or even “this is my string” where the delineator is a space.

This can be useful when getting data from a larger file that ou need to clean up. For example, data from a study or collection!

STRING TO LIST

Fruit represents a single item in the list. It changes as we iterate through the list. Every loop, we’re looking at the next item of fruit inside fruits.

```
fruits = ['apple', 'banana',
‘cherry']
```

Fruits references the specific list

```
for fruit in fruits:
    print(len(fruit))
```

Fruit can be named anything, but this is typical naming convention.

BACK TO FOR / IN LOOPS

LAB TIME

Coming up: Review, loops, advanced selection statements

NEXT CLASS: