AP CSP Stack Journal

5/8- FRQ #4

In the procedure flower_size (size), passing different inputs will cause different segments of our code. In our code, the user will input their ideal flower size that they want. We give them the option to have a “small” or “medium” or “large” flower. Then depending on the input, we will run through the list of flower heights to find ones that match the height that they wanted. For example, if they choose that they want a “small” flower, that will correlate to flowers with max heights below 24 inches. Then, we will show a list of all of the flowers that have that max height. Then we will randomly select one from that list in order to recommend a flower that has a max height of 24 inches. It will be printed with “Here is a list of flowers that match your size requirements: “ before and then “We recommend that you get some “. However, if they don’t input a “small,” the code will run to see if they chose “medium” and if not, then “large.” If the user does not input any of these choices, the code will print “ERROR, please reenter a request.” This shows how different inputs of ideal flower sizes will result in sentences printed that recommend different flowers that match that size.

5/6- Test Part 2

5/2- Logic Gates and Flow Charts

- The **NOT gate** flips the input. If you put in a 1, it gives you a 0. If you put in a 0, it gives you a 1. It’s like saying, “I’m not tired,” which means you’re awake.
- Think of **flow charts** when doing while loops

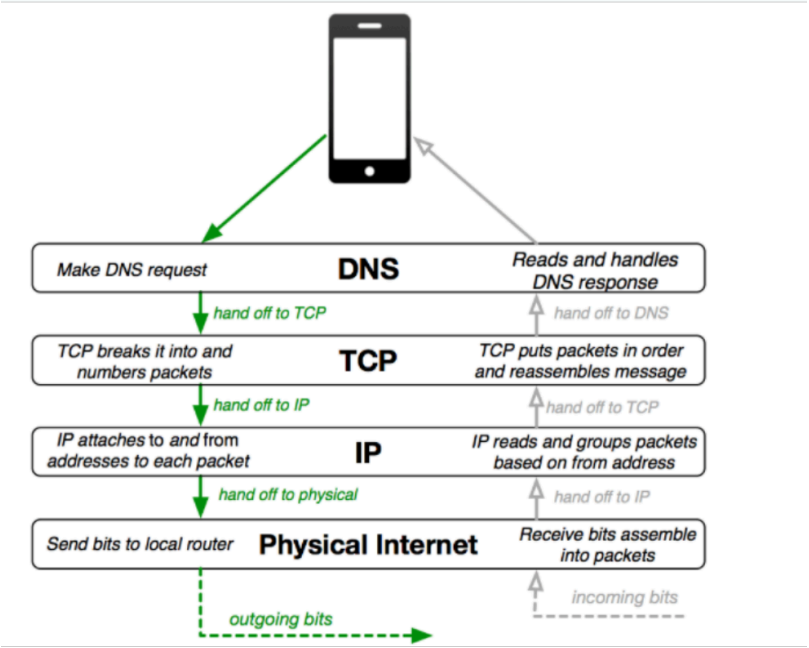
4/30- Test Part 1

4/28- Review

- It's important to use scratch paper for the robot type worksheets

4/24- Layers of the internet and FRQ #3

- **A packet** contains the binary information for an image, text, video, or whatever you are sending
    - It can only last for so long, if they don't arrive on time they will self destruct (you don't want that data just floating around on the internet)
- **A router** is how you connect all of your home devices. It's a mini computer that takes in packets a pushes them up. It figures out the best route to send them to their destination.
- The internet itself is built to be **fault tolerant** so if there is a failure, the whole internet doesn't break down
- **Transmission Control Protocol** handles all of the sending and receiving of your data as packers
- **HTTP** helps transfer data from a client to a server



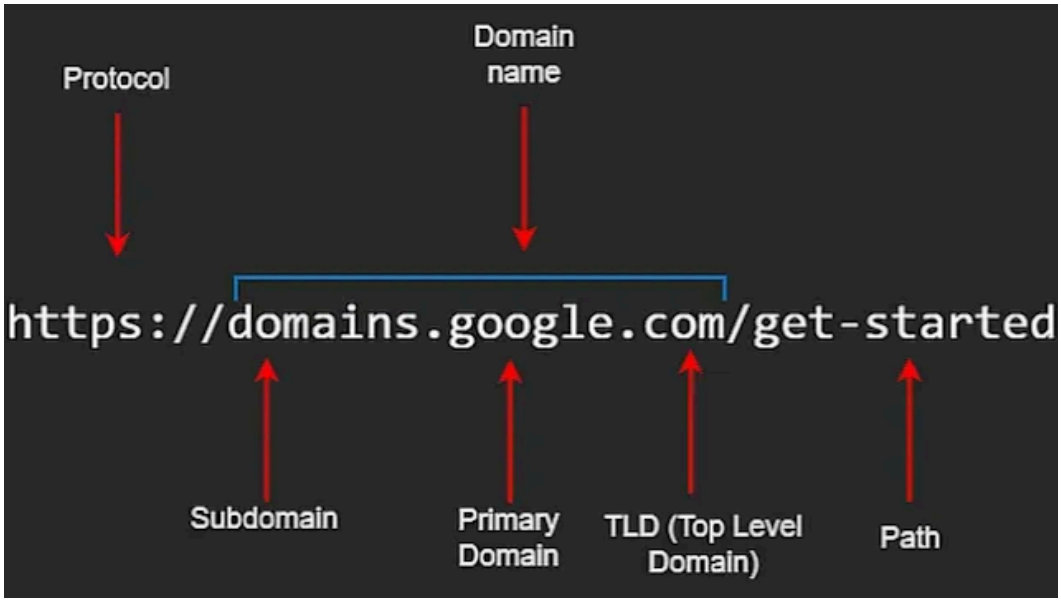
- **SSL-** Secure sockets layer **TLS-** Transport layer security
    - A layer of security around your data

Our list is called flower_choice and it stores 30 names of different flowers. These are the flowers that our code will be choosing from when it is recommending flowers to the user. In this specific code segment each flower has a care level associated with it: low care, medium care, and high

care. It does this through a loop. It uses the len() function to see how many items are in our list. Then that many items is the amount of times that our loop goes through a list. It finds the items in the list typeof_care that match the input. For example if the user inputs the word “low care,” our code will loop through the typeof_care list to find every time that “low care” shows up. It will then take the index number of where “low care” is. Then it will match that index to the same index in our list flower_choice. Using that name of the flower that is in the same index, it will add it to a new list called filtered_list using the append function. We are then able to have a new list (filtered_list) that has all of the flowers (from flower_choice) that have the same index as the flowers from typeof_care that match the type of care that the user wanted. There is a series of if statements in our code depending on the type of care that they want. If they want low care, the code will search through the typeof_care list for the index of “low care.” However if they input “medium care” it will do the same, but looking for the index of “medium care.” In the end, the code produces a list of flowers that match the type of care that the user wants and since it is a separate list, it will be easy to string together and print as a statement for the user to view.

4/22- IP Addresses

- **Protocol-** A well-known set of rules and standards used to communicate between machines
- **IP** means internet protocol
- **Hierarchy** is a ranking system
- IP addresses are 32 bits long
    - The first number is the city
    - The second would be people in the south loop
    - The third would be in Jones
    - Then the fourth and final will be for yourself
- **Domain name system** will take what you write in google, look up the IP address, and make sure that your computer can view it
    - Helps so you don’t have to remember the IP addresses for everything
    - It was created to be an open and public system
    - **DNS spoofing** is when hackers change the IP address aon a website so when people look something up, it will connect them to a different website
- Primary domain is google, yahoo, github, etc



4/17- Intro to the internet and FRQ #2

- **First and Last Mile-** Texts sent, notifications received, and apps used
- **Internet Hub-** Where our messages go to before being sent to next location
- **Internet Backbone-** How we communicate with others across the country

4/15- Submit CREATE Task

Today we submitted all of the parts to our CREATE task to AP Classroom.

4/11- Fifth Work Day of CREATE Task

Today we downloaded our code as a PDF. Here it is. We also recorded the video. Here is the video.

4/3- Fourth Work Day of CREATE Task

Today we worked on our Personalized Project Reference so that it is ready for AP testing day (Audrey Dziedzic- Personalized Project Reference - Google Docs). We also cited our sources at the end of our code for all of our flower pictures. Then we added comments to our code.

Python

```
def flower_size(size): #This function helps users find flowers that fit the
users size requirement and also recommends a flower from that list
    if size == "small": #This is for if the user inputs that they want a small
flower
        for i in range(len(max_height)): #It runs through all of our list of
flowers to
            if max_height[i] <= 24: #This is for the flowers that are under a
height of 24 inches as their max height

def flower_temp(): #This similar to flower_size, but we are using the
temperature of where the user lives instead
    temp = int(input("What is the average temperature during the time when you
plan on planting your flowers?")) #Here, they tell us where they live so that
we can find a flower that works in that climate
    if temp <= 28:
        print("There are no flowers in this list that can survive in that
temperature.")
    elif temp >= 29 and temp <= 44:
        for i in range(len(min_temps)): #This loop here means that it needs to
continue to loop through our min_temps until every single flower in that
min_temp range, has been added to the filtered_list
            if int(min_temps[i]) >= 29 and int(min_temps[i]) <= 45: #We had to
put int() in front becuase we are recieving numbers as our input
                filtered_list.append(flower_choices[i])
```

4/2- Third Work Day of CREATE Task

Today we finished the program by making three new functions and finishing the last function. We finished the last function that we left off on which was the flower temperature function. Now that function works by the user putting in the average temperature when they are expecting to plant the flowers and recommending flowers that will thrive the best in that temperature. We created a function where the user inputs how much they want to have to care for the flowers. It then matches the user with flowers that have that care requirement. We also made a function where the user inputs a flower and then you can see a picture of it. The last function we made was a menu function where we compiled all the functions into a list for the user to choose what they want to do.

Python

```
else:
        print("ERROR, Please reenter request")

def flower_temp():
    temp = int(input("What is the average temperature during the time when you
plan on planting your flowers?"))
    if temp <= 28:
        print("There are no flowers in this list that can survive in that
temperature.")
    elif temp >= 29 and temp <= 44:
        for i in range(len(min_temps)):
            if int(min_temps[i]) >= 29 and int(min_temps[i]) <= 45:
                filtered_list.append(flower_choices[i])
        print("Here are a list of flowers that can survive during the time you
plan on planting them: " + str(filtered_list))
        print("We recommend that you should plant some " +
random.choice(filtered_list) + "s.")
    elif temp >= 45 and temp <= 50:
        for i in range(len(min_temps)):
            if int(min_temps[i]) >= 45 and int(min_temps[i]) <= 50:
                filtered_list.append(flower_choices[i])
        print("Here are a list of flowers that can survive during the time you
plan on planting them: " + str(filtered_list))
        print("We recommend that you should plant some " +
random.choice(filtered_list) + "s.")
    elif temp >= 51 and temp <= 65:
        for i in range(len(min_temps)):
            if int(min_temps[i]) >= 51 and int(min_temps[i]) <= 65:
                filtered_list.append(flower_choices[i])
        print("Here are a list of flowers that can survive during the time you
plan on planting them: " + str(filtered_list))
        print("We recommend that you should plant some " +
random.choice(filtered_list) + "s.")
    elif temp >= 66 and temp <= 75:
        for i in range(len(min_temps)):
            if int(min_temps[i]) >= 66 and int(min_temps[i]) <= 75:
                filtered_list.append(flower_choices[i])
        print("Here are a list of flowers that can survive during the time you
plan on planting them: " + str(filtered_list))
        print("We recommend that you should plant some " +
random.choice(filtered_list) + "s.")
    elif temp > 75:
        print("There are no flowers in this list that can survive in that
temperature.")
```

```
    else:
        print("ERROR, Please reenter request")

def Care_Type():
    Care = input("How much do you want to have to care for your flowers? (Low
care, Standard care, Complex care)")
    if Care == "Low care":
        for i in range(len(typeof_care)):
            if typeof_care[i] == "Low care":
                filtered_list.append(flower_choices[i])
        print("Here are a list of flowers that match your type of care you
want: " + str(filtered_list))
        print("We recommend that you should plant some " +
random.choice(filtered_list) + "s.")
    elif Care == "Standard care":
        for i in range(len(typeof_care)):
            if typeof_care[i] == "Standard care":
                filtered_list.append(flower_choices[i])
        print("Here are a list of flowers that match your type of care you
want: " + str(filtered_list))
        print("We recommend that you should plant some " +
random.choice(filtered_list) + "s.")
    elif Care == "Complex care":
        for i in range(len(typeof_care)):
            if typeof_care[i] == "Complex care":
                filtered_list.append(flower_choices[i])
        print("Here are a list of flowers that match your type of care you
want: " + str(filtered_list))
        print("We recommend that you should plant some " +
random.choice(filtered_list) + "s.")
    else:
        print("ERROR, Please reenter request")

def flower_image():
    image = input("Enter a flower type.")
    if image in flower_choices:
        flower_choices.index(image)
        print(images_flower[flower_choices.index(image)])
    else:
        print("Your flower is not in the list provided, please enter another
flower type")

def menu():
    print("Welcome to the flower Recommender")
```

```
    while True:
        z = input("""What would you like to do:
1. Display all flower types
2. Find a flower based off height
3. Find a flower based off temperature
4. Find a flower based off care type
5. Find a flower based off image
6. Quit""")
        if z == "1":
            print(flower_choices)
        elif z == "2":
            flower_size()
        elif z == "3":
            flower_temp()
        elif z == "4":
            Care_Type()
        elif z == "5":
            flower_image
        elif z == "6":
            break
        else:
            print("ERROR, Please reenter request")

#Main
menu()
```

3/19- Second Work Day of CREATE Task

Today we really started our code. We filtered through all of the flowers to find the ones that fit the size that the user wants. So we first greeted the user and then asked them what size of flower that they wanted. We used the maximum height for each flower to measure this. And then if the flower were in the range of what they wanted, we filtered through our flower_choices array using max_height. Then we had a print statement that showed the results. For next class, we are going to continue to narrow down the types of flowers that we recommend by asking the temperature of where they live to then see if the flower will survive there.

```
Python
import random

#Functions
```

```
print("Welcome to the flower recommendation application. Please answer the
following questions in order to get the best recommendation.")

def flower_size():#First step is to give a list of flowers that match the users
size requirement of the flower
    size = input("What size of flower would you like? (small, medium, large)")
    if size == "small":
        for i in range(len(max_height)):
            if max_height[i] <= 24:
                filtered_list.append(flower_choices[i])
    if size == "medium":
        for i in range(len(max_height)):
            if max_height[i] > 24 and max_height[i] <= 35:
                filtered_list.append(flower_choices[i])
    if size == "large":
        for i in range(len(max_height)):
            if max_height[i] > 35:
                filtered_list.append(flower_choices[i])
    print("Here are a list of flowers that match your size requirements: " +
str(filtered_list))
    print("We recommend that you should get some " +
random.choice(filtered_list) + "s.")

def flower_temp():
    global flower_size
    temp = input("What is the average temperature during the summer where you
live?")
    if temp <= 28:
        print("There are no flowers in this list that can survive in that
temperature.")
    if temp >= 29 and temp <=

#Main
flower_size()
```

3/17- First Work Day of CREATE Task

Today we gathered all of the data into a spreadsheet. We picked 30 flowers that we are going to have as the options of the flowers that the user may get as their recommendation. For each, we also found the temperatures needed for it to grow successfully, the maximum and minimum height that the flower will grow to, along with if the flower was considered low care, standard care, or complex care based on how often you have to water it. For each flower, we also found

an image and cited our sources of it. Then in GitHub we made an array for each of these categories and made sure that it had the right commas and quotes. For the next class, we plan on getting the skeleton of our code done. We will probably get the code for how we greet the user to the game and ask them what they want in their dog, and then have it filter through our arrays. [CREATE Project - Google Sheets]

```python
#AP Create Task
#Audrey Dziedzic and Amelie Lynde

#Init
from PIL import Image
import requests
from io import BytesIO

flower_choices =
["Sunflower","Carnation","Rose","Pansies","Dahlia","Iris","Lavender","Chrysanth
emum","Tulip","Marigolds","Petunia","Daisy","Lily","Orchid","Poppy","Hyacinth",
"Daffodil","Begonia","Zinnia","Sweet","pea","Cornflower","Amaryllis","Daylily",
"Hibiscus","Hydrangea","Azalea","Dandelions","Alstroemeria","Abutilon"]
max_height =
[180,24,18,12,72,48,36,36,28,48,24,48,72,36,36,12,30,24,36,36,30,36,48,84,72,72
,18,36,120]
min_height =
[12,6,8,6,12,6,10,12,6,6,6,12,12,12,12,6,4,6,12,24,12,18,24,36,36,12,6,12,12]
ideal_temps =
[70-78,50-80,60-75,45-65,60-70,60-85,60-85,60-70,29-60,70-80,60-75,60-75,60-75,
65-80,60-75,60-75,60-65,60-75,74-84,45-68,60-75,65-75,65-75,60-85,65-80,45-80,5
0-77,65-80,60-75]
typeof_care = [Low care","Complex care","Complex care","Complex care","Standard
Care","Standard Care","Complex care","Complex care","Low care","Standard
Care","Standard Care","Complex care","Standard Care","Low care","Low
care","Standard Care","Standard Care","Complex care","Standard Care","Complex
care","Standard Care","Complex care","Standard Care","Standard Care","Standard
Care","Standard Care","Low care","Complex care","Standard Care"]
```

3/11- CREATE Task Rubric

- [Create PT Guidelines - Google Docs]

3/7- Dog Breeds Assignment

Python

```python
#Init
import random
dog_breeds =
dog_weights =
dog_image =
filtered_list = []

#Functions
def dog_size():
    x = input("What size dog are you looking for? (tiny, small, medium,
large)")
    if x == "tiny":
        for i in range(len(dog_weights)):
            if dog_weights[i] <= 10:
                filtered_list.append(dog_breeds[i])
    if x == "small":
        for i in range(len(dog_weights)):
            if dog_weights[i] < 11 and dog_weights[i] > 25:
                filtered_list.append(dog_breeds[i])
    if x == "medium":
        for i in range(len(dog_weights)):
            if dog_weights[i] < 26 and dog_weights[i] > 60:
                filtered_list.append(dog_breeds[i])
    if x == "large":
        for i in range(len(dog_weights)):
            if dog_weights[i] > 60:
                filtered_list.append(dog_breeds[i])
    print("We recommend that you get a " + random.choice(filtered_list))

def information():#allows users to look up information (temperament and image)
about a specific dog breed.
    y = input("Enter a dog breed")
    if y in dog_breeds:
        dog_breeds.index(y)
        print(dog_temperament[dog_breeds.index(y)])
        print(dog_image[dog_breeds.index(y)])
    else:
        print("Your dog breed is not in the list provided, please enter another
dog breed")

def purpose():# filters and displays dog breeds based on user-specified purpose
```

```
    theirchoice = input("What qualities do you want in your dog?")
    for i in range(100):
        if theirchoice in bred_for[i]:
            filtered_list.append(dog_breeds[i])
    print("Here are the dogs with your quality" + str(filtered_list))
    if len(filtered_list) == 0:
        print("There are no dogs with this quality")

def menu():
    print("Welcome to the Dog Breed Recommender")
    while True:
        z = input("""What would you like to do:
1. Display all dog breeds
2. Find a dog based off size
3. Find a dogs temperament and image
4. Find a dog based off its purpose
5. Quit""")
        if z == "1":
            print(dog_breeds)
        elif z == "2":
            dog_size()
        elif z == "3":
            information()
        elif z == "4":
            purpose()
        elif z == "5":
            break
        else:
            print("ERROR, Please re enter request")
#Main
menu()
```

3/3- Parameters Review

Steps to successfully implementing a parameter.

- Step #1
- Create the parameter
- Make sure to name it something simple and descriptive.
- Step #2
- **Supply an argument** whenever you call your function in the future.
- Keep in mind what data type that parameter is expecting.
- Step #3

- Use the parameter inside your function
- Think about what your parameter is changing in your function.
- Understand that the parameter is just holding the value that the person who uses the function is providing.
- If you do not use the parameter inside your function, it does nothing.

Python

```
#Init
import turtle
audrey = turtle.Turtle()

#Functions
def house(size,material):
    if material == "wood":
        audrey.color("brown")
    if material == "brick":
        audrey.color("red")
    if material == "stone":
        audrey.color("gray")
    audrey.begin_fill()
    audrey.left(180)
    for i in range(4):
        audrey.forward(size)
        audrey.left(90)
    audrey.right(45)
    audrey.forward(size*.72)
    audrey.left(90)
    audrey.forward(size*.72)
    audrey.end_fill()

def price_house(size, material):
    global total_price
    price = 50
    if size <= 50:
        price = price*1.25
    elif size >51 and size <100:
        price = price*1.5
    elif size > 150:
        price = price*2
    price = price
    if material == "brick":
        price = price*1.25
    elif material == "wood":
        price = price*1.5
```

```
    elif material == "stone":
        price = price*2
    print(price)

def message(name):
    if name == "amelie":
        print("This is a secret message.")
    else:
        print("Hi.")

#Main
house(10,"wood")
price_house(90,"brick")
message("audrey")
```

2/27- Array Practice Challenges

Python

```
fruitList = ["banana" , "apple" , "grape" , "blueberry" , "banana" , "apple" , "apple"]
otherList = []

#1. Write a process that returns true if an item is contained in a list and false if not
def thisList(item, list):
    if item in list:
        return("Yes")
    else:
        return("No")

#2. Write a process that returns how many times an item appears in a list
def item_count(item, list):
    x = list.count(item)
    return(x)

#3. Write a process that returns true if the list is empty and false if it is not
def is_list_empty(list):
    list = []
```

```
    if not otherList:
        return("The list is empty")
    else:
        return("The list is not empty")

#Main
print(thisList("strawberry", fruitList))
print(item_count("apple", fruitList))
print(is_list_empty(otherList))
```

2/24- Displaying Images

```
Python
#Start by importing the necessary libraries
from PIL import Image
import requests
from io import BytesIO


#Define a function called open_image with a url parameter for the image address
def open_image(url):
    response = requests.get(url)
    img = Image.open(BytesIO(response.content))
    img.show()


#Call your function with an image URL
open_image("https://cdn2.thedogapi.com/images/0LJiOVlxp.jpg")


#DO NOT FORGET TO CITE YOUR SOURCE
#The adorable face of a dog. Image source: The Dog API. 2009. Accessed via
https://cdn2.thedogapi.com/images/0LJiOVlxp.jpg CC BY-NC 2.0.
```

- To site your sources, go to the website and find the **link to the website** and **author** and the **date** that it was published
- To **shorten the links** you can go to [URL Shortener, Branded Short Links & Analytics | TinyURL]

Python

```python
#Init
from PIL import Image
import requests
from io import BytesIO
import random

#List of Desserts
dessertList = ["https://tinyurl.com/58rtmsp6" , #Berry Tart
               "https://tinyurl.com/yc7mpk5v" , #Macaroons
               "https://tinyurl.com/yx4fx8ab" , #Pie
               "https://tinyurl.com/bdeh42u7" , #Ice cream
               "https://tinyurl.com/5n6aejut"] #Cupcake

#Functions
def open_image(url):
    response = requests.get(url)
    img = Image.open(BytesIO(response.content))
    img.show()

#Main
def randomrecommender():
    print("Welcome to the dessert recommendation app.")
    question = input("Do you want your recoomendation (y/n)?")
    recommend = random.choice(dessertList)
    if question == "y":
            if recommend == "https://tinyurl.com/58rtmsp6":
                print("This is a berry tart. This has fruits in it so if you
like fruits it is good. Also, it might be a little healthier for a dessert.")
            elif recommend == "https://tinyurl.com/yc7mpk5v":
                print("This is a macaroon which are originally from France.
They are very expensive usually because they are so hard to make, but have a
unique texture.")
            elif recommend == "https://tinyurl.com/yx4fx8ab":
                print("This is a pie. Pies are good for a holiday like for
Thanksgiving you may want a pumpkin pie and in the spring, maybe you want an
apple pie. ")
            elif recommend == "https://tinyurl.com/bdeh42u7":
                print("This is ice cream, which is nice during the summer if it
a hot day because it is cold. You can get it on a cone or in a bowl.")
            else:
                print("This is a cupcake. This is good because you can hold it
in your hand and don't need a fork to eat it. It comes in a lot of flavors
too.")
            open_image(recommend)
```

```
    else:
            print("Ok.")

randomrecommender()

#Sources
#The berry tart image was found at the Playing With Flour website
(http://www.playingwithflour.com/2012/05/mini-fruit-tarts-for-moms.html) by
Monica on March 13th, 2019
#The macaroon image was found at the Free Dessert Photos website
(https://www.pexels.com/search/dessert/) by an unkwon use and date
#The pie image was found at the recipes.com website
(https://recipepes.com/indx.html?utm_content=savory+pie+recipes) by
y.e.ygt@hotmail.com in 2024
#The ice cream image was found at the H-O-M-E webiste
(https://h-o-m-e.org/does-ice-cream-have-eggs/) by William Armstrong on an
unknown date
#The cupcake image was found at The Busy Baker website
(https://thebusybaker.ca/best-ever-chocolate-cupcakes-from-scratch/) by
Chrissie on 12/19/23
```

- Pay attention to the **order** of the code, so that its runs good

2/20- Copyright and Creative Commons

- **Copyright** is a type of intellectual property that gives its owner the **exclusive legal right** to copy, distribute, adapt, display, and perform a creative work (this is for life or 95 years). You do **not** have to apply for a copyright.
- **Creative Commons** are for when you want people to use and share your work.
    - About CC Licenses - Creative Commons
- The following are different licenses… (they can be combined with eachother)
    - **BY** attached means that you have to give credit the creator (all licenses have this on them)
    - **SA** attached means that adaptations must be shared under the same terms
    - **NC** attached means that only noncommercial uses of the work are permitted
    - **ND** attached means that no derivatives or adaptations of the work are permitted.

2/13- Word Counter

- **Split Method-** Splits a string into a list

Split a string into a list where each word is a list item:

```
txt = "welcome to the jungle"

x = txt.split()

print(x)
```

- When the object is inside of the string, the **len() function** returns the number of characters in the string

Return the number of items in a list:

```
mylist = ["apple", "banana", "cherry"]
x = len(mylist)
```

Python

```
#Functions
def word_counter(text):
    list = text.split()
    print(list)
    x = len(list)
    print("Word Count: " + str(x))

#Main
word_counter("Today, I am sitting in a chair")
```

2/11- Return

- A python **return statement** marks the end of a function and specifies the value or values to pass back from the function
    - They can return data of any type

Python

```
def double_num(num):
    print(num * 2)

VS.

def double_num(num):
    return(num * 2)
```

```
FOR
x = double_num(137)
print(x / 2)

NOW
print(is_even(10))
```

- The solution to a function is printed in the **terminal** but does NOT **return a value**
    - Prints it, but we can't use it again later in code (gives it back to them)
- NOW we have to put **the print** in the **main** section!!!!!!
    - Not just the name of the function… it has to be **print(functionName())**

```
Python
#Functions
def last_item(list):
    return(list[len(list)-1])

#Main
numList = [21,48,92,87]
last_item(numList)
```

- **Len** means **length** of list
    - So len(list)-1 will give the last item in the list

2/7- Pixels

- A pixel is a **picture element** (a little tiny dot on your screen)
- Each pixel is made out of **RGB** (red, green, and blue) lights
- **Resolution** is the amount of pixels on your screen (more pixels = more clear image)
- **Pixel Density-** Objects can have the same resolution, but the **smaller objects** with have tighter pixels = more resolution
    - Bigger the screen size, the more pixels that you need for sharp images
- RGB scale ranges from 0 (not showing up) to 255 (will be saturated) and they can be combined with other amounts of R, G, and B, to create a whole new color for each pixel
- R is the first bit, G is the second bit, and B is the third (24 bits per pixel is the **standard** but each color gets 8 each (R gets the 1-8, G gets 9-16, B gets 17-24))
- **Meta Data-** Data about/describes data
    - Such as image width, image height, and bits per pixel
    - Usually the first thing that you need to do
- **Hexa Decimal** is what we use now

2/5- Tetris Score Analysis

- Make sure to make **that the list is GLOBAL**
    - Everytime you use a score, it is accessing that list
- Saying what is in the array is **defining your variable**
- **Can't name** a variable the **same thing** as a function!!!!
- **On skill check-** To find the number of items in a list…
  `print(len(scoreList))`

Python

```
#Tetris Score Analysis
#Audrey Dziedzic 2/5/25

#Init
maxScore = 0
minScore = 3600460

# List of Tetris scores (unsorted)
scoreList = [3600460, 1388900, 5435960, 4222920, 8063900, 1362703, 6529560,
2043580, 1390000, 1696224,
    1372600, 2535840, 2605320, 2275996, 11966100, 1472600, 1390780, 1768400,
1333731, 1317580,
    1777456, 4899280, 2790920, 1626880, 1476400, 29486164, 4570640, 1649100,
4890220, 6492500,
    1614120, 5157860, 1582980, 1357480, 2382340, 1532660, 2378832, 1316520,
2529080, 13793540,
    1386260, 1352620, 1442340, 1406260, 1705680, 12409180, 2281848, 3222400,
1349060, 2302480,
    1571120, 3067100, 3740500, 1606732, 2153480, 2011360, 1344740, 1371060,
1345420, 2373940,
    1702940, 2077552, 2108820, 1322485, 1404800, 2555620, 1405580, 4213540,
3835120, 6563440,
    1350742, 1537800, 1657560, 1333995, 1632505, 2743060, 1509751, 1554880,
1316540, 1323680,
    2433160, 1340460, 1659860, 1608500, 7220241, 1834120, 7081880, 1483360,
16700760, 1435280,
    1702640, 1371040, 1316900, 2114180, 1374100, 1412260, 1379220, 1319573,
1623160, 6787420]

#Create a function that finds and return the highest score in the list.
def highestScore():
    global maxScore
    global scoreList
    for score in scoreList:
```

```
        if score > maxScore:
            maxScore = score
    return(maxScore)

#Create a function that finds and return the lowest score in the list.
def lowestScore():
    global minScore
    global scoreList
    for score in scoreList:
        if score < minScore:
            minScore = score
    return(minScore)

#Create a function that calculates and return the average score in the list.
def averageScore():
    global scoreList
    sum = 0
    for i in range(len(scoreList)):
        sum = sum + scoreList[i]
    avScore = sum/i
    print(avScore)

#Create a function that allows the user to input a score. If the score is
greater than the lowest score on the Tetris score list, remove the lowest score
and the new score will be added to the list. Reject any scores that are not in
the top 100 scores.
def inputScore():
    global scoreList
    global maxScore
    global minScore
    theirScore = int(input("What is your best score?"))
    if theirScore > maxScore:
        scoreList.append(theirScore)
        scoreList.remove(minScore)
        print("Your score has been added to the list!")
    else:
        print("Score not in the top 100.")

#Main
print(highestScore())
print(lowestScore())
averageScore()
inputScore()
```

2/3- Arrays With Loops

- Array with words need to be in **quotes**
- The **i** helps to print the next thing in our array, going through our list, one by one
- If the amount of things in your array changes, the index doesn't work because there is **nothing there** = error
    - That's why you need to use the **function len (length)**
    - Put the name of the list inside of the parentheses
    - Then it **calculates the number of items in the list**
    - It will then stop printing once it reaches the **end of the list**
- The first variable is whatever you want it to be (be descriptive) and then the second is the function that you want
    - It grabs every item for that **variable**
    - The variable after **for** and the variable in the **print** must match

Python

```
numberList = [1,61,43,23,13,5,68,98]
fruitList = ["cherry","banana","pear","apple","grape","strawberry"]

#Simple for loop: Prints the FIRST 5 numbers in our list
for i in range(5):
    print(numberList[i])

#For loop with len(): Prints ALL of the numbers in our numberList
for i in range(len(numberList)):
    print(numberList[i])

#For loop using elements: Prints ALL of our fruits in our fruitList
for fruit in fruitList:
    print(fruit)
```

- It's good for variables to start **empty** or at **zero**
- Make sure that the **print statement,** is outside of the loop, because otherwise it will print every **iteration** of the sum at each step

Python

```
numberList = [1,61,43,23,13,5,68,98]
fruitList = ["cherry","banana","pear","apple","grape","strawberry"]

#Print the sum of all the numbers in the number list
sum = 0 #This is our variable (good to start empty or at zero)
for i in range(len(numberList)):
    sum = sum + numberList[i]
print(sum)
```

```
#Print the largest number in the numbers list
largestNum = 0
for number in numberList:
    if number > largestNum:
        largestNum = number #MUST BE IN THIS ORDER SO THAT IT SWITCHES WHICH
NUMBER IT'S STORING
print(largestNum)
```

1/30- Slot Machine

Python

```
#Audrey Dziedzic 1/28 - 1/30
#Slot Machine

#Init
import random
slots = ["7" , "❀" , "✿" , "❀" , "✿"] #This has an increased probability for
the flower slots compared to the 7's
credits = 100 #Start amount of credits that they have to play with
bet = 0

#Functions
def slotsGame():
    print("Welcome to the Slot Machine. You have a chance to win if your shapes
match. ❀7✿") #Introduction to the game
    print("You have 100 credits to bet with.")
    def spinSlots():
        global result1
        global result2
        global result3
        global credits
        global bet
        result1 = random.choice(slots) #This allows the computer to randomly
pick a slot from the slot function
        result2 = random.choice(slots)
        result3 = random.choice(slots)
        bet = int(input("How many credits do you want to bet?"))
```

```
        if bet > credits: #Makes sure that they only bet the amount that they
have and then have to rebet if they didn't have enough
            print("You do not have enough credits to play the game. Please
rebet a valid amount.")
            bet = int(input("How many credits do you want to bet?"))
        else:
            print("You bet " + str(bet) + " credits. The slot machine is
spinning!")


    def displaySlots():
        global result1
        global result2
        global result3 #They variables have to be global because we used them
from the spinSlots function that told what the results were
        print("These are yours results: " + str(result1) + " " + str(result2) +
" " + str(result3))

    def checkWin():
        global result1
        global result2
        global result3
        global credits
        global bet
        if result1 == '7' and result2 == '7' and result3 == '7': #Matches the
results of it spinning to weather or not it is a win
            print("You got the Jackpot!")
            credits = credits + 3 * bet
            print("You now have " + str(credits) + " credits to bet with.")
        elif result1 == '❀' and result2 == '❀' and result3 == '❀':
            print("You won! ❀")
            credits = credits + 2 * bet
            print("You now have " + str(credits) + " credits to bet with.")
        elif result1 == '✿' and result2 == '✿' and result3 == '✿':
            print("You won! ✿7")
            credits = credits + 2 * bet
            print("You now have " + str(credits) + " credits to bet with.")
        else:
            print("You lost.")
            credits = credits - bet #Makes them lose the amount of credits by
how much they bet with
            print("You now have " + str(credits) + " credits to bet with.")

    while True:
```

```
        spin = input("Do you want to spin the slotmachine (y/n)?")
        if spin == 'y' and credits >= 0: #Allows them to spin
            spinSlots()
            displaySlots()
            checkWin()
        elif spin == 'y' and credits == 0: #But not if they don't have enough
credits
            print("You ran out of credits to play.")
            break
        else:
            print("Thank's for playing.")
            break #Stops the while True loop

#Main
slotsGame() #Put it into one function
```

1/24- Try and Except

Python

```
try:
    x = int(input("Please enter a number 1-10: "))
except ValueError:
    print("ERROR: Please enter a number!!")
```

- Python string ends with () Method- LINK

Python

```
#Init
import random
magic_list = ["For sure" , "Totally" , "Without a doubt" , "Ya duh" , "Of
course" , "Eh" , "Maybeish" , "Perchance" , "I guess", "50/50", "Ugh no",
"Absolutely not" , "No way" , "Nope" , "Not at all"]

#Functions
def magicball():
    print("Welcome to the magic 8 ball.")
    while True:
```

```
        print("Please ask the magic 8 ball a yes/no question that you would
like answered.")
        question = input("What is your question?")
        if question.endswith("?"):
            print(random.choice(magic_list))
        else:
            print("ERROR: Enter a question.")
        anotherq = input("Do you want to ask another question (y/n)?")
        if anotherq == 'n':
                print("Thanks for playing!")
                break

#Main
magicball()
```

1/22- Arrays

- An **array** is an ordered collection of items, like a **list**
    - Can hold movies, numbers, string, and even other arrays
- When **creating list**, use brackets, then each item is separated with a comma in python
- You can assign an array to a **variable** in order to use it later
    - Store a list inside a named variable and give it a name
- Every item in an array has a **position**
    - The 1st is **index 0**, the 2nd is **index 1**, and the 3rd is **index 2**, etc.
- Resources:
    - Python List/Array Methods
    - Python- Access List Items
- **Append** puts the thing that you want to the end of the list, but for **insert**, you get to pick the location for it

Python

```
#Initialize
#mySchedule stores a list of the classes you are currently taking at Jones  as
strings
mySchedule = ["Dance" , "Physics (A)" , "Comp Sci"]

#Main
#Task 1: Append periods 4 - 7 to the list
mySchedule.append("Calc")
```

```
mySchedule.append("Spanish")
mySchedule.append("CRAC")
mySchedule.append("Physics (B)")

#Task 2: Insert your two lunch periods(A day and B Day) in their appropriate
location
mySchedule.insert(3, "C lunch")
mySchedule.insert(7, "A lunch")

#Task 3: Remove your least favorite class
mySchedule.remove("CRAC")

#Task 4: Print just your 2nd period class by accessing the appropriate index in
your array
print(mySchedule[1])

#Task 5: Print only your A day schedule and then only your B day schedule
print(mySchedule[0:3])
print(mySchedule[4:6])
```

Python

```
#Initialize
themeList = []

#Functions
def makeList():
    print("Welcome to my Theme List Organizer")
    print("Please select one of the following options: ")
    print("1. View \n2. Add \n3. Remove \n4. Quit")

    while True:
        #Collect User Choice
        option = input("Select a number between 1-4: ")
        option = int(option)

        #Process Information
        if option == 1:
            print("Here is your theme list: ")
            print(themeList)

        if option == 2:
            addItem = input("What would you list to add to your list?")
```

```
            themeList.append(addItem)
            print("Successfully added " + addItem)

        if option == 3:
            removeItem = input("What would you list to remove from your list?")
            themeList.remove(removeItem)
            print("Successfully removed " + removeItem)

        if option == 4:
            print("Thanks for letting me make your list.")
            break

#Main
makeList()
```

1/8- Parameter Practice and Multiplication Quiz

- **Parameters** are special variables that a function needs to complete a task. When the function is called, values are passed in as arguments inside the ( )

Python
```
#This function adds two numbers together and prints the rest
def sum(num1,num2):
    print(num1 + num2)

#This function takes a name and prints a welcome message
def welcome(firstName):
    print("Welcome to Jones, " + firstName) #String concatenation

#This function takes in a temp as celcius, and prints the equivalent temp as
farenheit
def celcius_to_farenheit(celsius):
    print("33.8"*celsius)

#Main
celcius_to_farenheit(20) #Need this, so when you use the code, you have to
provide the information in order for it to run properly
```

- A **for loop** is used so that you can repeat a loop in the amount of times that you want. This is a specific number that you can choose or have someone input as the parameter. Whereas a **while loop** will continue for forever, until you break it.

Python

```python
#Area of a Rectangle
#Activity: Write a function calculate_area(length, width)
def area(length, width):
    print(length*width)
#Main
area(2,3)

#Discount Calculator
#Activity: Write a function that takes the original price and a discount
percentage as parameters and prints the discounted price.
def discount(price, percent):
    print(price - price*percent/100)
#Main
discount(100,10)

#Repeated Greeting
#Activity: Write a function that takes a name and a times parameter.
#The function should print a greeting with the name repeated the specified
number of times using a loop.
def greeting(name, times):
    for i in range (times):
        print("Hello," + name)

#Main
greeting("Audrey", 3)
```

- This link tells you how to input a timer- [Create a Timer in Python: Elapsed Time, Decorators, and more – LearnDataSci]

Python

```python
#Init
import random
import time

#Functions
def quiz():
    global score
    score = 0
    print("Welcome to the multiplication quiz!")
    print("How many questions would you like the quiz to be?")
    length = int(input("Enter the number of questions you want:"))
    start_time = time.time()
    for i in range (length):
```

```
        computerone = random.randint(0,10)
        computertwo =  random.randint(0,10)
        print("What is",str(computerone),"times",str(computertwo),"?")
        answer = int(input("Enter what you think that answer is:"))
        print("Your answer:", str(answer))
        correctanswer = (computerone*computertwo)
        if answer == correctanswer:
            print("That is correct!")
            score = score + 1
        else:
            print("That is wrong.")
            score = score
    end_time = time.time()
    elapsed_time = end_time - start_time
    print("You finsished the quiz. You got", str(score), "correct out of",
str(length))
    print("It took you",str(elapsed_time), "seconds.")

#Main
quiz()
```

1/6- Rock, Paper, Scissors

- **=** gets the value of
- **==** is equal to
    - Ex: `if computer == 2:`
      `computer = "scissors"`

Python

```
#Init
import random
player_score = 0 #Global
computer_score = 0 #Global

#Functions
def game():
    global player_score
    global computer_score
    print("Welcome to the Rock, Paper, Scissors Game!")
    while True:
```

```
        #Player's choice
        print("Please select one of the three options.")
        player = input("Selection:")
        #Computer's Choice
        computer = random.randint(1,3)
        if computer == 1:
            computer = "rock"
            print("The computer chose rock")
        elif computer == 2:
            computer = "paper"
            print("The computer chose paper")
        else:
            computer = "scissors"
            print("The computer chose scissors")
        #Determine the outcome
        if player == "rock" and computer == "scissors":
            player_score = player_score + 1
            computer_score = computer_score
            print("You won")
            print("You have" ,player_score, "points and the computer has" ,
computer_score)
        elif player == "rock" and computer == "paper":
            player_score = player_score
            computer_score = computer_score + 1
            print("You lost")
            print("You have" ,player_score, "points and the computer has" ,
computer_score)
        elif player == "rock" and computer == "rock":
            player_score = player_score
            computer_score = computer_score
            print("Tie")
            print("You have" ,player_score, "points and the computer has" ,
computer_score)
        elif player == "paper" and computer == "scissors":
            player_score = player_score
            computer_score = computer_score + 1
            print("You lost")
            print("You have" ,player_score, "points and the computer has" ,
computer_score)
        elif player == "paper" and computer == "paper":
            player_score = player_score
            computer_score = computer_score
            print("Tie")
```

```
            print("You have" ,player_score, "points and the computer has" ,
computer_score)
        elif player == "paper" and computer == "rock":
            player_score = player_score + 1
            computer_score = computer_score
            print("You won")
            print("You have" ,player_score, "points and the computer has" ,
computer_score)
        elif player == "scissors" and computer == "scissors":
            player_score = player_score
            computer_score = computer_score
            print("Tie")
            print("You have" ,player_score, "points and the computer has" ,
computer_score)
        elif player == "scissors" and computer == "rock":
            player_score = player_score
            computer_score = computer_score + 1
            print("You lost")
            print("You have" ,player_score, "points and the computer has" ,
computer_score)
        elif player == "scissors" and computer == "paper":
            player_score = player_score + 1
            computer_score = computer_score
            print("You won")
            print("You have" ,player_score, "points and the computer has" ,
computer_score)
        #Ask player if they want to continue
        playagain = input("Would you like to play again?")
        if playagain == "yes":
            print("Restarting...")
        elif playagain == "no":
            print("Ok, thanks for playing.")
            break

#Main
game()
```

12/16- Conditional Challenges

- Write a program that compares two numbers (x and y) and prints the larger number.
  If x > y, print "x is larger".
  If x < y, print "y is larger".
  Otherwise, print "Both numbers are equal".

Python

```
x = int(input("Pick a number for x: "))
y = int(input("Pick a number for y: "))
if x > y:
    print("x is larger")
elif x < y:
    print("y is larger")
else:
    print("Both numbers are equal")
```

- Write a program that checks if a number (n) is divisible by 3 or 5.
  If n is divisible by 3 or n is divisible by 5, print "Divisible by 3 or 5".
  Otherwise, print "Not divisible by 3 or 5".

Python

```
n = int(input("Pick a number for n: "))
if n % 3 == 0 or n % 5 == 0:
    print("Divsible by 3 or 5")
else:
    print("Not divisible by 3 or 5")
```

12/12- Ticket Generator

Python

```
#Initialize
import turtle
t = turtle.Turtle()

def ticket_generator():
    #1: Greet the person
    print("Welcome to the circus ticket generator.")

    #2. Enter in their information

    name = input("Enter your name:")
    age = int(input("Enter age:"))
    day = input("Enter the day of the week:")
    coupon = input ("Enter in your coupon code (if you don't have one, enter
n/a:)")

    #3. Create an algorithm
    if age <= 3:
```

```
        price = 0
    elif age >= 4 and age <= 17 and day == "monday" or day == "tuesday" or day
== "wednesday" or day == "thursday" or day == "friday":
        price = 50
    elif age >= 4 and age <= 17 and day == "saturday" or day == "sunday":
        price = 100
    elif age > 17:
        price = 1000

    elif coupon == FREEFRIDAY and day == "friday" and age <= 17:
        price = 0
    elif coupon == SUNDAY10 and day == "sunday":
        price = 90


    #4. Print the tickets
    def draw_ticket(name, price, day, y_location):
        t.goto(-50, y_location)
        t.write("Ticket", font=("Arial", 15), align="right")
        t.pendown()
        for i in range(2):
            t.forward(500)
            t.left(90)
            t.forward(250)
            t.left(90)
        t.penup()
        t.goto(50, y_location +215)
        t.write("Admit One", font=("Arial", 15), align="right")
        t.goto(440, y_location +215)
        t.write(day, font=("Arial", 15), align="right")
        t.goto(225, y_location +135)
        t.write(name, font=("Arial", 15), align="right")
        t.goto(225, y_location +15)
        t.write(price, font=("Arial", 15), align="right")

    #Main
    draw_ticket(name, price, day, 1)

#Main
ticket_generator()
```

12/10- MadLibs

- Website for **string concatenation-** Python - String Concatenation
- Website on how to **bold text-** How to print Bold text in Python [5 simple Ways]

Python

```
#Audrey Dziedzic 12/10

#Int

#Functions
ans1 = input("Name a character")
ans2 = input("Name a setting")
ans3 = input("Name a place to travel")
ans4 = input("Enter a number 30-60")
ans5 = input("Enter a sport")

#Main
print("Welcome to Mad Lib Story! Answer the questions to create a story!")
print("Once upon a time", ans1,"lived in an old and smelly",ans2,". They wanted
to go somewhere exciting, so they planned a trip to",ans3,". But the flight was
delayed",ans4," minutes.", ans1," was very upset because they missed the
important ",ans5, "game in ",ans3,".")
```

12/6- AI and Pokemon

- **Large language models** (like **Chat GPT**) are trained on the largest amount of information possible
- AI uses **probabilities** to predict what response you want

Python

```
#Initialize (these are our global variables)
pokemon_level = 0 #Global
pokemon_name = "Squirtle"
day = 1
import random

#Functions
def train():
    global pokemon_level
```

```
    global day
    print("Your pokemon just did 10 push ups.")
    pokemon_level = pokemon_level + 1
    print("After day " + str(day) , "your pokemon is at level: ",
pokemon_level)
    day = day + 1

def gym_battle():
    global pokemon_level
    global day
    print("Your Squirtle is going to compete in a battle with another
Pokemon.")
    outcome = random.randint(1,2) #50% chance to win or lose
    if outcome == 1:
        pokemon_level = pokemon_level + 2
        print("Congrats! Your pokemon won and is now at level: ",
pokemon_level)
    else:
        print("Sorry, but your pokemon lost and is still at level: ",
pokemon_level)
    day = day + 1

def rest():
    global pokemon_level
    global day
    print("Your Pokemon is going to rest.")
    print("After resting, your pokemon is still at level: " +
str(pokemon_level))
    print("So it's name is ", pokemon_name)

def evolve():
    global pokemon_level
    global pokemon_name
    if pokemon_level == 0:
        pokemon_name == "Squirtle"
    elif pokemon_level == 2:
        print("Your pokemon has evolved, the new name: Wartortle.")
        pokemon_name = "Wartortle"
    elif pokemon_level == 4:
        print("Your pokemon has evolved, the new name: Blastoise.")
        pokemon_name = "Blastoise"

#Main
print("Welcome to Pokemon Evolution")
```

```
while True:
    print("Select an activity for day: " + str(day))
    print("""1.Train
2.Gym Battle
3.Rest(Display info)
4.Quit""")
    activity = int(input("1-4) Activity for the day: "))
    if activity == 1:
        train()
        evolve()
    elif activity == 2:
        gym_battle()
        evolve()
    elif activity == 3:
        rest()
    else:
        break
```

12/4- Machine Learning

- For help in programming, you can use the **Duck Debugger** on the side
- Website for python conditions like **if** statements- Python Conditions
- This is my code for the **secure login** assignment…

Python

```
#Functions
def login():
    valid_username = "ADZI"
    valid_password = "miles"
    username = input("Please enter your username: ")
    username = username.upper()  #this allows for case-insensitive comparison
    password = input("Please enter your password: ")
    if username == valid_username and password == valid_password:
        print("Access granted.")
    else:
        print("This is not a valid username and password.")

#Main
login()
```

- **Machine Learning-** How computers recognize patterns and make decisions with being explicitly programmed
    - The more examples that you give it, the more accurate it is

12/2- String Methods

- **String methods-** Are functions that are associated with an object and can manipulate its data or perform actions on it
- Website with python string methods and how you would use them- Python String Methods
- Every character in a string has a **position/index**
    - The first letter is 1, the second letter is 2, and so on

```python
message = "computer science at jones is the best?"

#Complete the following tasks using string methods
#Task 1: Capitilize the first letter
x = message.capitalize()
print(x)

#Task 2: Uppercase the sentence (Use all capital letters)
y = message.upper()
print(y)

#Task 3: Replace the ? with an !
w = message.replace("?" , "!")
print(w)

#Task 4: Find and print the position of the word jones in the string
q = message.find("jones")
print(q)
```

11/21- Local vs Global Variables

- If you create a variable, you can only use it **in that function**
- In the first line of the function, **state which** type of variable you are using
- **Global** is helpful for when you need to use it for the whole time

```python
#Initialize
message = "message" #Global Variables
#Can be used everywhere

#Functions
def drawPikachu():

def greet():
    global message
    mesage = "Hello World" #Global Variable
    #It only exists inside the functions
    print(message)

#Main
greet()
print(message)
```

11/19- While Loops

- **A While loop in Python** is a control flow statement that allows you to repeatedly execute a block of code as long as a **given condition is true.** It''s useful when you don’t know in advance how many times the loop should run, as the number of iterations depends on the condition.
- ----------
  while condition:
  # Code block **to execute**

  How it works:
  The condition is **evaluated before each iteration**

  If it's True, the code **inside the loop runs**
  If it's False, the loop stops, and the program **moves to the next statement after the loop**

To avoid infinite loops, **ensure the condition eventually becomes False** (by updating variables inside the loop)

- When you have an infinite loop, you have to **interrupt it** and close the window
- To exit a loop that is going to go for infinite time (we **don’t** want this), just type in **break**

Python

```python
#Example 1:
i = 0
while i < 5:
    print("i is: " + str(i))
    i = i + 1

#Example 2:
while True: #Forever loop
    print("This will loop forever")
    break
```

- For **multiple line strings,** use **triple quotes**
    - But be sure to delete the white space
- For variables, just do **=** but for a while statement, do **==**

Python

```python
#Function
#Adds num1 and num2 and prints the result
def add(num1,num2):
    result = num1 + num2 #Or print(num1+num2)
    print(result)

def subtraction(num1, num2):
    result = num1 - num2
    print(result)

def division(num1, num2):
    result = num1 / num2
    print(result)

def multiplication(num1, num2):
    result = num1 * num2
    print(result)

def simplecalc():
    print("Welcome Preschoolers to Simple Calculator")
while True:
    print("Enter an operation: ")
    print("""1.Addition
    2.Subtraction
    3.Division
    4.Multiplication
```

```
    5.Quit""")
    operation = int(input("(1-5)Operation: "))
    if operation == 1: #True
        int1 = input("Enter your first number: ")
        int2 = input("Enter your second number: ")
        sum = int(int1) + int(int2)
    print(sum)

    if operation == 2: #True
        int1 = input("Enter the number you want to subtract from: ")
        int2 = input("Enter the amount to subtract: ")
        subtraction = int(int1) - int(int2)
    print(subtraction)

    if operation == 3: #True
        int1 = input("Enter your dividend: ")
        int2 = input("Enter your divisor: ")
        division = int(int1) / int(int2)
    print(division)

    if operation == 4: #True
        int1 = input("Enter your first number: ")
        int2 = input("Enter your second number: ")
        multiplication = int(int1) * int(int2)
    print(multiplication)

    if operation == 5:
        break
#Main
simplecalc()
```

Khan Academy Notes

- Procedure is the same as a **function**
- Binary Numbers…
    - **Odd** numbers end in a 1 for **binary**
    - **C/F** for temperature and **AM/PM** on a clock, can be **represented with 1 bit**
    - 3 ON/OFF options need 3 bits of code
    - A bit can **only** store a 1 (on) or a 0 (off)
- Variables…
    - Code runs from **top to bottom** so if it is changed above, it is now changed when printed

11/13- Robot Drawing

- **To change variables-** highlight, then right click, then change all occurrences (will highlight every time that variable appear in code, finally type in what you want to rename the variables in every single place
    - This is important so that it can describe what it is holding
- **Quick step-** you plug in for other quick steps, so it might be in between other code
    - But keep repeating each time that it appears
- After if, is **If** then **else**
- It will tell you what **n starts as** and then go through and it will tell you what it becomes

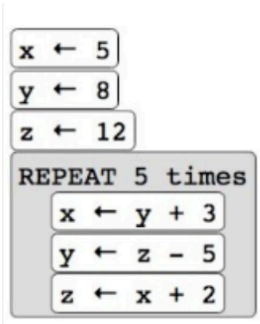
11/11- Random Number

```python
#Import
import random

#Function
print("Welcome to Guessing Game! All you have to do is guess the number 0-10")
print("Guess the number!")
guess = int(input("Enter Guess")) #integer
secrect = random.randint(0,10) #integer
def Guess_Number(secrect):
    print(guess)
    if guess == secrect:
        print("Correct!")
    if guess > secrect:
        print("Your guess is too high!")
    if guess < secrect:
        print("Your guess is too low!")

#Main
Guess_Number(3)
```

11/7- Tracing + Bytes and File Sizes + Programing

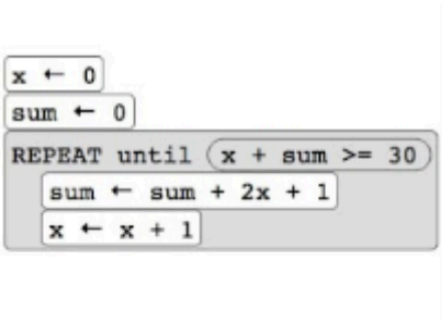
- Once a variable is changed, that is the **new value** of it

Challenge #1



X	Y	Z
11	7	13
10	8	12
11	7	13
10	8	12
11	7	13

Challenge #2



sum	x
1	1
4	2
9	3
16	4
25	5
36	6

- Examples and calculations of **bytes…**

Unit	Number of (Bytes, KB, MB, etc)	Example of File Type or Data Measured in this Unit
Kilobyte (KB)	1,024 bytes	A five page paper might be 100 KB An email without images is about 2 KB
Megabyte (MB)	1,024 kilobytes	A high quality digital picture is about 25 megabytes MP3 audio is about 1 megabyte per minute
Gigabyte (GB)	1,024 megabytes	A DVD disk has a capacity 4.7GB (single layer) A flash drive might hold 32 GB A hard drive might hold 750 GB
Terabyte (TB)	1,024 gigabytes	1 TB = 200,000 5-minute songs; 310,000 pictures; or 500 hours worth of movies
Petabyte (PB)	1,024 terabytes	1 PB = 500 billion pages of standard typed text (or 745 million floppy disks)
Exabyte (EB)	1,024 petabytes	To fill an exabyte, you would need a video call 237,823 years long.

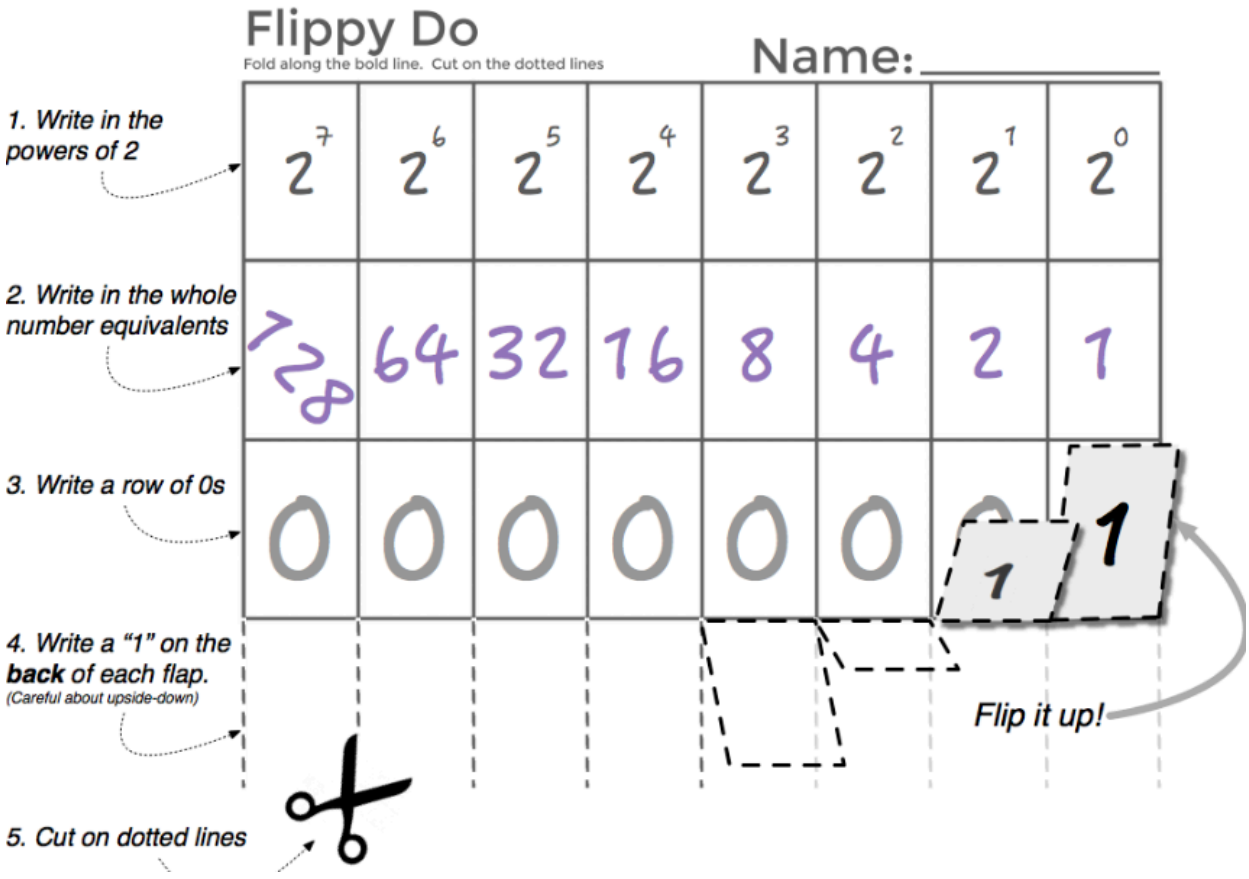
- Mod in code is **% sign**
    - **if year % 400 == 0: #Checking to see if the year is divisible by 400, to get no remainder**
- **Order** is important- do the most **specific rule first**
    - Then check the **general/vague rule last**
- The **parameter** is year. When we **call the function, we supply** the year.

Python

```
#Functions
def leap_year(year): #the parameter is year. when we call the function, we supply the year.
#divisible by 4, leap year *if it is divisible by 100, not a leap year * if it is divisable by 400, it is a leap year
    if year % 400 == 0:
        print("it is a leap year")
    elif year % 100 == 0:
        print("it is not a leap year")
    elif year % 4 == 0:
     print("it is a leap year")

#Main
leap_year(2024)#True
leap_year(1900)#False
leap_year(1600)#True
```

11/1- Binary Number System

- Computers only understand an **ON / OFF state**
    - We had to come up with a number system (which all operate the same way) to help us communicate with computers
- Flowing through the wires is electricity
- More wires = more bits = higher range of numbers = more **numbers you can store**
- Binary number system has 2 digits: **0** (off) and **1** (on)
- Each place value represents **1 bit** (which can be zero or one and is the smallest piece of info that a computer can store)
- Our Flippy Do has 8 bits which together make **1 byte**
- Decimals to binary numbers…
    - 7 = 0000 0111
    - 20 = 0001 0100
- Binary numbers to decimals…
    - 0001 0010 = 18
    - 0001 1111 = 31
- **Modulo Operator-** Divide and it will print what the remainder is
    - count mod 2 = ___ **means that** if you divide the count by 2, means to find the <u>remainder</u>
- **Flippy Do…**



10/30- Random Substitution Cipher

- **Symmetric Encryption-** One key was used to encrypt the information and then you can use that same key to decrypt the information (one single key)
- **Asymmetric Encryption-** Two keys (one public key and one private key)
    - Ex: your email… you can send them to anyone but they are being encrypted by a public key. There is only one person that can read the emails that you read, because they have a private key (they have to log into their email with credentials)

10/28- Cryptography

- **Encryption-** The process of scrambling data into an unreadable, encoded version that only authorized users can access
- **Decryption-** The process of converting an encrypted message back into the original form
- **Cipher-** The algorithm used to encrypt or decrypt a message
- **Key-** A piece of information that unlocks the algorithm for encoding or decoding a message
- **Crack-** Unscrambling an encrypted message without knowing the key

10/24- Variable Swap

- Milk is a number and you can't put it in, so you have to add str (string concatenation)
- Milk is a variable

Python

```
#Audrey Dziedzic
#99 bottles of milk

#Fucctions
def bottles():
    milk = 99
    for i in range (97):
        print(str(milk) + " bottles of milk on the wall, " + str(milk) + "
bottles of milk on the wall. Take one done, pass it around " + str(milk-1) + "
bottles of milk on the wall!")
        milk = milk - 1
    print(str(milk) + " bottles of milk on the wall, " + str(milk) + " bottles
of milk on the wall. Take one done, pass it around " + str(milk-1) + " bottle
of milk on the wall!")
```

```
    print(str(milk-1) + " bottle of milk on the wall, " + str (milk-1) + "
bottle of milk on the wall. Take one done, pass it around. No more bottles of
milk on the wall. Boo Hoo!")
    milk = milk - 1

#Main
bottles()
```

- X, y, and z are the variables
- Computer runs the code in order from top to bottom
- Cant just do y = x and then x = y, because you changed x in the line previously
    - Once you change a variable, it is **changed forever** and can't go back
    - So its a good idea to **create a 3rd variable**

```
Python
#Variable Swap

#Swap the value of x and y without using any numbers using three lines
x = 5
y = 4
z = 0 #Z is useful here

#Your code goes here...
z = x
x = y
y = z

print(x) #Should print 4
print (y) #Should print 5
```

- Errors…
    - **Syntax errors** occur when the program does not follow the rules or grammar of the programming language. (Ex: If you forget a parenthesis for a function or quotation marks for strings. The program will not run and give you error messages)
    - **Logical errors** occur when the program follows the syntax of the programming language, but does not do what you want it to do. (Ex: If you use the wrong variable, operator, or function in your code, you will get a logical error. The program will run and not give you an error message but it will not work as intended)

10/15-10/17- Boolean

Python

```
#Logic 1 (AND Operator)
#num = integer
def is_in_range(num):
    # Check if the number is between 10 and 20 (inclusive)
    if num >= 10 and num <= 20:
        print(True)
    else:
        print(False)
# Example: is_in_range(15) should print True, is_in_range(25) should print
False


#Logic 2 (OR Operator)
#day = string
def is_weekend(day):
    # Check if the day is either Saturday or Sunday
    if day == "Saturday" or day == "Sunday":
        print(True)
    else:
        print(False)
    #print(True)
# Example: is_weekend("Saturday") should print True, is_weekend("Monday")
should print False


#Logic 3 (NOT Operator)
# num = integer
def is_not_negative(num):
    # Check if a number is not negative
    if not num < 0:
        print(True)
    else:
        print(False)


# Example: is_not_negative(5) should print True, is_not_negative(-3) should
print False
```

- **Boolean:** True or False
- **Boolean Expressions:** 17 > 3 Evaluates to a Boolean

- **not** returns the opposite boolean
- If it's a word, put it in **quotes** for Main

10/10- Conditionals

- Age and Exam are the **variables**
- **if** and **elif** and **else:** are important to tell your computer what to do
- **int function-** for text
- **str function-** for numbers
- The following are **logic operators-**
    - **>** is greater than and **<** is less than
    - **>=** is greater than or equal to and **<=** is less than or equal to
    - **!** is not equal to
    - **==** means that it is equal to
    - **and** is for when you need **multiple things (BOTH) need to be true**
    - If one is false, it will NOT print the statement
    - **or** is for when only **ONE THINGS needs to be true**
- max_num() has **3 parameters- a,b,c**
- score_to_grade grade has **1 parameter- score**

```python
#Conditionals Statements
#Init

#Functions

#16 years or older
#Passed your drivers ed exam
def drive_check():
    age = int(input ("Please enter your age: ")) #Input by default collects
strings
    exam = input ("Did you pass your Drivers Ed. Exam?: ")
    #Process with conditional statements
    if age > 15 and exam == "yes": #Evaluate as True or False
        print("You are eligible to obtain your license!")
    else:
        print("You are NOT eligible to obtain your license!")

#18 years or older
#U.S. citizen
def vote_check():
```

```
    age = int(input ("Please enter your age: ")) #Input by default collects
strings
    exam = input ("Are you are U.S. Citizen?: ")
    #Process with conditional statements
    if age >= 18 and exam == "yes": #Evaluate as True or False
        print("You are eligible to vote!")
    else:
        print("You are NOT eligible to vote!")

#a = int
#b = int
#c = int
#Print the largest of the 3 numbers
def max_num(a, b, c):
    #No input needed
    #Process the data with conditional statements
    a = int(input("Please enter a number: "))
    b = int(input("Please enter another number: "))
    c = int(input("Please enter one more number: "))
    if a > b and a > c:
        print("a is the largest. the value of a is: " + str(a))
    if a < b and b > c:
        print ("b is the largest. the value of b is: " +str(b))
    if a < c and b < c:
        print ("c is the largest. the value of c is: " +str(c))

def score_to_grade(score):
    #No input needed
    if score > 89:
        print("A")
    elif score > 79:
        print ("B")
    elif score > 69:
        print ("C")
    elif score > 59:
        print ("D")
    else:
        print ("F")

#Main
drive_check()
vote_check()
max_num(5,10,15)
score_to_grade(75)
```

10/8- Headlines

- **String Concatenation-** Linking strings together (do this using a **+ sign)**
- You can **not** concatenate integers, only strings
    - **str(age)** instead of **age**
    - This function changes an data into a **string**

```python
firstName = "Audrey "
lastName = "Dziedzic"
age = 17
print(firstName + " " + lastName)
print("I am " + str(age) + " years old")
```

- **No quotes** around variables

```python
#Clickbait Headline Generator

#Function
def believe_headline():
    noun = input("Please enter a noun: ")
    ppronoun = input ("Please enter a possessive pronoun: ")
    place = input ("Please enter a place: ")
    print("You won't believe what this " + noun + " found in " + ppronoun + " "
+ place)

def restaurant_sued():
    restaurant = input("Please enter a restaurant name: ")
    money = input("Please enter a really big number in letters: ")
    print("Did you know that " + restaurant + " was sued for " + money + "
dollars last week")

def to_jail():
    name = input("Please enter a name of a person: ")
    city = input("Please enter a city name: ")
    print("Breaking: " + name + " was sent to the " + city + " jail")

#Main
believe_headline()
restaurant_sued()
```

```
to_jail()
```

10/4- Intro To GitHub

- To make it a python file, it has to **end with .py**
- Text has to be in **quotes**
- This function has 1 parameter, which is your first name
- “Welcome to Jones” is the string
- **The one parameter is celsius**

9/30-10/2- Digital Scene Project

- Group Planning Guide- [Design a Digital Scene - Group Planning Guide - Google Docs]

```python
#Init
import turtle
audrey=turtle.Turtle()

#Functions
def drawTshape(): #Audrey code: This gets the first shape of the "T" for the
snowflake
    audrey.color("#66ccff") #This tells the color that the turtle is
    audrey.forward(10)
    audrey.right(90)
    audrey.forward(5)
    audrey.right(180)
    audrey.forward(10)
    audrey.right(180)
    audrey.forward(5)
    audrey.left(90)
    audrey.forward(5)
    audrey.right(180)
    audrey.forward(15)
    audrey.right(90)

def drawSnowflake():
    audrey.penup()
    audrey.goto(random.randint(-350,350),random.randint(-350,350))
```

```
    #Audrey code: This means that the location of the x coordinate and y
coordinate will be random
    #So that the whole snowflake goes to a random spot
    audrey.pendown()
    #This draws the "T" shape 4 times so that the snowflake has 4 "T" shapes
    for i in range(4):
        drawTshape()

def drawAllsnowflakes(): #Audrey Code
    for i in range(10):
        drawSnowflake()
    #Now I have 10 random snowflakes

def drawStump(): #Audrey Code
    audrey.penup()
    audrey.goto(-400,-100)
    audrey.pendown()
    #This makes the color of the turtle and the fill color the same
    audrey.color("#996633","#996633")
    audrey.begin_fill()
    #The turtle will fill now
    for i in range (4):
            audrey.forward(50)
            audrey.left(90)
    #This draws a square
    audrey.end_fill()

def drawLeavesOne(): #Audrey code
    #I now did a rectangles but green, above the stump and longer
    audrey.color("#006600","#006600")
    audrey.begin_fill()
    audrey.penup()
    audrey.goto(-290,-50)
    audrey.pendown()
    audrey.left(90)
    for i in range (2):
        audrey.forward(60)
        audrey.left(90)
        audrey.forward(160)
        audrey.left(90)
    audrey.end_fill()

def drawLeavesTwo(): #Audrey Code
    audrey.color("#339933","#339933")
```

```
    #This is the same as drawLeavesOne() but a lighter color and smaller
    audrey.begin_fill()
    audrey.penup()
    audrey.goto(-310,10)
    audrey.pendown()
    for i in range (2):
        audrey.forward(50)
        audrey.left(90)
        audrey.forward(120)
        audrey.left(90)
    audrey.end_fill()

def drawLeavesThree(): #Audrey Code
    audrey.color("#00cc00","#00cc00")
    #This is even lighter of a color and smaller than drawLeavesTwo()
    audrey.begin_fill()
    audrey.penup()
    audrey.goto(-330,61)
    audrey.pendown()
    for i in range (2):
        audrey.forward(40)
        audrey.left(90)
        audrey.forward(80)
        audrey.left(90)
    audrey.end_fill()

def drawStar(): #Audrey code
    audrey.penup()
    audrey.goto(-355,130)
    audrey.pendown()
    audrey.color("#ffcc00","yellow")
    audrey.begin_fill()
    for i in range (5): #I repeated my triangle 5 times, because a star has 5
points to it
        for i in range(2):
            audrey.forward(20)
            audrey.left(120) #This got me a triangle but not closed off, so
that it looks better
            #I turned it 120 degrees because 360 divided by 3 sides
        audrey.left(192)
        #This positions the turtle so that it could do this again but at the
right location
        #So that they don't overlap
    audrey.end_fill()
```

```
def drawTree(): #Audrey code: To build a tree you need all of these elements so
I called them
    drawStump()
    drawLeavesOne()
    drawLeavesTwo()
    drawLeavesThree()
    drawStar()
#Main
def drawAudreyScene(): #Audrey code:
    #The two things that I made were snowflakes and a tree so I needed them to
show up
    drawAllsnowflakes()
    drawTree()
```

9/25- Loops and Variables

- A **variable** is a container of memory
    - Put before your loop

Python

```
#Functions
def challenge1():
    print("Hello, World")

def challenge2(): #Print 1-10 in ascending order
    num = 1
    for i in range(10):
        print(num)
        num = num + 1
        #Increment our variable by 1

def challenge3(): #Print 1-10 in descending order
    num = 10
    for i in range (10):
        print (num)
        num = num - 1
```

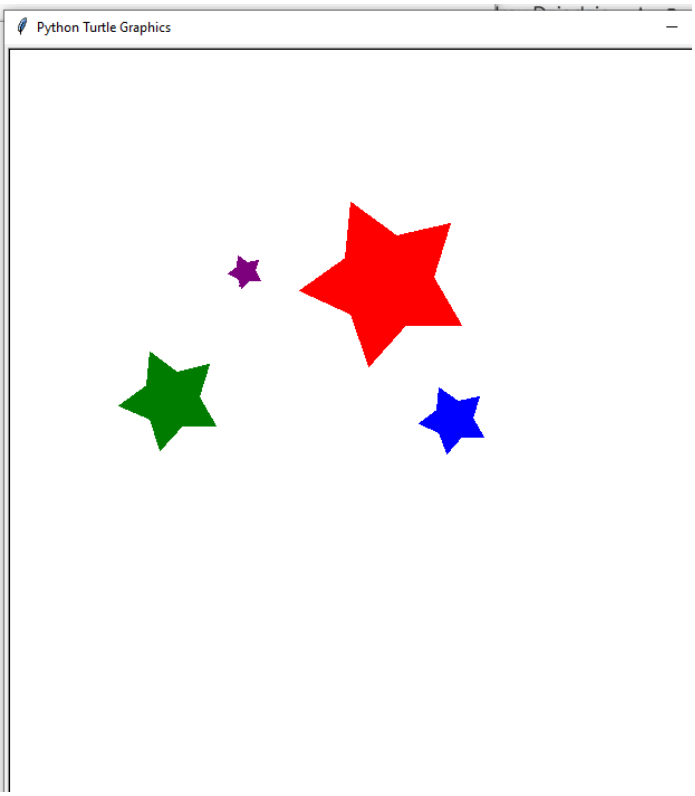
9/23- Parameter Practice

Python

```python
#Init
import turtle
audrey=turtle.Turtle()

#Functions
def drawPolygon(sides):
    for i in range (sides):
        audrey.forward(100)
        audrey.left(360/sides)

#Main
drawPolygon(3)
drawPolygon(5)
drawPolygon(8)
```

File Edit Format Run Options Window Help

```
#Audrey Dziedzic
#Period 3
#9/23/24

#Init
import turtle
audrey=turtle.Turtle()
audrey.speed(50)

#Functions
def drawStarfish(size, color, x,y):
    audrey.penup()
    audrey.goto(x,y)
    audrey.pendown()
    audrey.color(color)
    audrey.begin_fill()
    for i in range (5):
        for i in range(2):
            audrey.forward(size)
            audrey.left(120)
        audrey.left(192)
    audrey.end_fill()

def draw_all_Starfish ():
    drawStarfish(50, "red", 0,100)
    drawStarfish(20, "blue", 50,0)
    drawStarfish(30, "green", -200,10)
    drawStarfish(10, "purple", -140,140)

#Main
draw_all_Starfish()

```



9/19- Parameters

- **Parameters-** A variable (container) that customizes the way a function works
    - The **value** inside the parenthesis
    - The dot function has **2 functions**…the dot **size** and the dot **color**
- Steps..
    - **1.** Create your parameter
    - **2.** Provide an argument (a value/number) to your calls
    - **3.** Use your parameter in the function
- In this case, **size** is the variable and we put it in a case to be **used**
    - Every time we call it, we have to put in a size value
- Multiple parameters are **separated by a comma**

```python
#Init
import turtle
audrey=turtle.Turtle()

#Functions
def square (size, color):
    audrey.color(color)
```

```
    for i in range (4):
        audrey.forward(size)
        audrey.left(90)

#Main
square(20, "blue")
```

9/16-19- Random

Random Dots

- **randint(minimum number, maximum number)** to get the different sizes of dots
- **RGB** you put like **(, ,)** to make the different colors
    - **0** will make the color not exist and **250** is the max of the color
    - Red first, then green, then blue

Python
```
#Init
import turtle
import random
audrey = turtle.Turtle()
#From this point forward we will represent color as RGB triplets
#Not as a string or hexadecimal number (1 for red, 1 for green, 1 for blue)
turtle.colormode(255)

#Functions

#Main
audrey.dot(random.randint(10,500),(random.randint(0,250),random.randint(0,250),
random.randint(0,250)))
```

- Before it should have been**…**
    - So we added the **random.randint** to all of them

Python
```
audrey.dot(random.randint(10,500),(0,0,0))
```

- This is it broken up and why is it the way it is…

```
#main
for i in range(100):
    mrj.dot( random.randint(10,300)#Size
             ,random.randint(0,255)#Red
             ,random.randint(0,255)#Green
             ,random.randint(0,255))#Blue
    mrj.teleport(random.randint(-500,500)#XCor
             ,random.randint(-500,500))#yCor
```

Random Squares

- For **functions** they will remember what you told them to do
    - It stores what it is and using that number or color consistency every time it comes

Finished Code

Python

```
#Audrey Dziedzic
#Period 3
#9/16/23

#Init
import turtle
import random
audrey = turtle.Turtle()
#From this point forward we will represent color as RGB triplets
#Not as a string or hexadecimal number (1 for red, 1 for green, 1 for blue)
turtle.colormode(255)
audrey.speed(50)

#Functions
def circles():
    for i in range(20):

audrey.dot(random.randint(10,500),(random.randint(0,250),random.randint(0,250),
random.randint(0,250)))
        audrey.penup()
        audrey.goto(random.randint(-500,500),random.randint(-500,500))
        audrey.pendown()
```

```
def onesquare():

audrey.color(random.randint(0,250),random.randint(0,250),random.randint(0,250))
#RGB
    audrey.begin_fill()
    #Variable
    square_size=random.randint(10,300)
    for i in range(4): #one square
        audrey.forward(square_size)
        audrey.right(90)
    audrey.end_fill()

def squares():
    for i in range (20):
        onesquare()
        audrey.penup()
        audrey.goto(random.randint(-500,500),random.randint(-500,500))
        audrey.pendown()

#Main
circles()
squares()
```

9/10- Fill colors

- To make it show the output faster do **audrey.speed(50)**

To create **the smile** of the smiley face…

- You have to change the direction that the turtle is **facing like an arrow** so it knows where to go
- The first number is the **radius** and the second is the **angle in degrees** that it is going to turn
- If the radius is positive, it will go **counterclockwise** and if negative, it will go **clockwise**
- To change the **thickness** of it go to **pensize**

Python

```
import turtle
audrey = turtle.Turtle ()

audrey.dot(700, "#ffff66")
```

```
audrey.penup()
audrey.goto(120,150)
audrey.pendown()
audrey.dot(200,"#009933")
audrey.penup()
audrey.goto(-120,150)
audrey.pendown()
audrey.dot(200,"#009933")
audrey.penup()
audrey.goto(-190,0)
audrey.right(90)
audrey.pendown()
audrey.pensize(30)
audrey.circle(200,180)
```

To **fill in a shape**..

- Set the color of the turtle first
- Make it a **sandwich** with **begin_fill** and **end_fill** first

Python
```
audrey.color("blue", "pink")
audrey.begin_fill()
for i in range(6):
    audrey.forward(100)
    audrey.right(60)
audrey.end_fill()
```

9/6- Dots

- To get colors, use **hex** and put it in the **string**

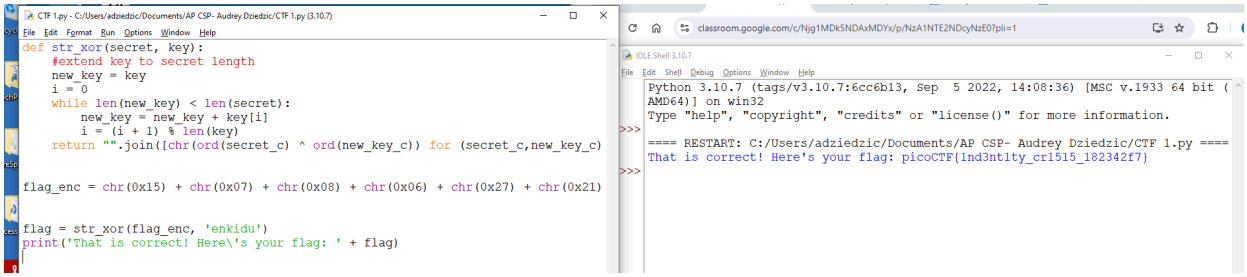
Python
```
def bullseye():
    audrey.dot(150, "#99ff99")
    audrey.dot(100,"#33cc33")
    audrey.dot(50, "#006600")
```

- For dots, make sure that you specify that size and color of the dot, putting **quotations** around the color
- It's like a coordinate plane and where the turtle starts **all** of its code is at **(0,0)**

```python
audrey.penup()
audrey.goto(-150,150)
audrey.pendown()
```

- Website for colors- https://www.w3schools.com/colors/colors_picker.asp

9/4- Code Structure



- Computer reads code from **top to bottom**
- All functions end in parenthesis
- New functions that you create begins with **def**
- If your original code is indented, you have to make sure that it is **indented again** for your new function
- **Call-** Calling the function actually performs the specified actions with the indicated parameters to be used

New function-

```python
def square():
    for i in range (4):
        audrey.forward(100)
        audrey.left(90)
```

- Then to use it write **square ( )**
- Separating your code out..
    - **Initialization-** Importing your turtle
    - **Functions-** To define your new functions
    - **Main-** Put all of your code here

```python
#Initialization
import turtle
audrey = turtle.Turtle( )
audrey.left(90)

#Functions
def drawStep():
    audrey.forward(40)
    audrey.right(90)
    audrey.forward(40)
    audrey.left(90)

def drawSide():
    for i in range(4):
        drawStep ()
    audrey.forward(40)
    audrey.left(90)

def drawDiamond():
    for i in range (4):
        drawSide()

#Main
drawDiamond()
```

8/30- Loops

- **Integers**- Whole numbers
- **String**- A sequence of characters inside of quotes
- **Variable**- A container or a box that you can store data in (does not need quotes) can be letter or numbers
- **Loop**- used to repeat a sequence of commands (where i is a variable) it exits the loop when the variables equals the loop number
- : - tells you what to loop so you have to **indent it**
- Python runs in order, from top to bottom so you have to define your **variables** first
- Name your variables something descriptive
- For **shapes**, Angle= 360/# of sides

```python
name = "audrey"

print("Welcome to my website" + name)
```

To make a square-

```python

for i in range(4):

    audrey.forward(100)

    audrey.left(90)
```

8/27- Turtle

- **import turtle-** this line creates a turtle and stores it inside of a variable (box)
- To call it your name… **audrey = turtle.Turtle( )**
- = means to get the value of
- **audrey.forward (100)** means to move the turtle **forward**
- **audrey.left (90)** means to turn the turtle 90 degrees to the **left**
- Want to do it in the most efficient way

```python
import turtle
audrey = turtle.Turtle()
```

8/26- Hello, World

```python
print("Hello, World")
```

- Text that begins with a **#** is a **comment**
- print(“ “) will print something in the **shell**
- **Shell-** interprets the code that you write
- **Bug-** error in code
- **Function-** sequence of commands that can be named and used to accomplish a task (Defining a function does not execute it. Defining it names the function and specifies what to do when the function is called)
- **Comment-** not a command to the computer, it’s ignored