String testing methods=
.isalnum() is the string alphanumeric .isspace() is the string all spaces
.lstrip() strip spaces from left end .replace(old,new)
.isalpha() is it alphabetical .startswith(substr) does the string start with subs
.rstrip() strip spaces from right end
.isdigit() is the string all numerical .endswith(substr)
.strip() strip spaces from both ends
.islower() is te string all lowercase] .lower() converts characters to lowercase
.split() split string into list of words
.isupper() is all uppercase .upper() converts all to uppercase
.find/rfind(substr) index where substr starts or -1

```
age = int(input("Please enter person's age: "))
if age < 2:
        category = "infant"
elif age >= 2 and age < 4:
        category = "toddler"
elif age >= 4 and age < 13:
        category = "child"
elif age >= 13 and age < 18:
        category = "teenager"
elif age >= 18 and age < 65:
        category = "adult"
else:
        category = "senior"
print("This person's age category:", category)
# Currency conversion program
print("""Convert currency to U.S. dollars
        e = Euros
        c = Chinese Yuan
        r = Indian Rupees
        j = Japanese Yen
        b = Bitcoin""")
currency = input("Which currency to convert: ")
if currency=="e":
        amount=float(input('Amount of Euros to convert: '))
        US_dollars=(amount*1.21)
elif currency=="c":
        amount=float(input('Amount of Yuan to convert: '))
        US_dollars=(amount*.16)
elif currency=="r":
        amount=float(input('Amount of Rupees to convert: '))
        US_dollars=(amount*.014)
elif currency=="j":
        amount=float(input('Amount of Yen to convert: '))
        US_dollars=(amount*.0096)
elif currency=="b":
        amount=float(input('Amount of Bitcoin to convert: '))
        US_dollars=(amount*46284.1)
else:
        print("Invalid currency code.")
        exit()
print("In U.S. dollars, that is $", format(US_dollars, '.2f'), sep="")

        import math
        nSquares = int(input("Number of squares to draw: "))
        screen = turtle.Screen()
        screen.bgcolor("yellow")
        ted = turtle.Turtle()  # Create a turtle; call it ted
        sideLength = 300
        for square in range(nSquares):
           for i in range(4):   # draw a square: do the following 4 times
              ted.forward(sideLength)      # draw a line
              ted.left(90)          # turn left to be ready for next line
           ted.forward(sideLength/2)
           ted.left(45)
           sideLength = sideLength / math.sqrt(2)
        screen.exitonclick() # close screen when user clicks on it

        numDiamond = 0
        numRuby = 0
        numEmerald = 0
        numSapphire = 0
        collect = True
        print("Welcome to the Gemstones Shop!\n")
        while collect == True:
                print("Which gemstone would you like to collect?")
                print("\t1 - diamond")
                print("\t2 - ruby")
                print("\t3 - emerald")
                print("\t4 - sapphire")
                print("\t5 - Exit\n")
                userChoice = int(input("Please enter your choice: "))
                if userChoice == 1:
                        numDiamond += int(input("How many diamonds would you like to col
")) 
                elif userChoice == 2:
                        numRuby += int(input("How many rubies would you like to collect? "))
                elif userChoice == 3:
                        numEmerald += int(input("How many emeralds would you like to collect? "))
                elif userChoice == 4:
                        numSapphire += int(input("How many sapphires would you like to collect? "))
                elif userChoice == 5:
                        pokemon = False
                else:
                        print("Invalid choice; please try again.")
print("Total diamonds collected is", numDiamond)
print("Total rubies collected is", numRuby)
print("Total emeralds collected is", numEmerald)
print("Total sapphires collected is", numSapphire)


import math  # Need so global variable math.pi is defined
def circumference(r):
        circumference = 2 * math.pi * r
        return circumference
def area(r):
        area = math.pi * (r**2)
        return area
def main():
        again = True
        while again:
                r = float(input("Enter the radius: "))
                if r == 0:
                        print("Goodbye!")
                        again = False
                elif r < 0:
                        print("Invalid radius!")
                else:
                        print("A circle of radius", r, "has circumference",
circumference(r), "and area", area(r))
main()

import turtle  #import the turtle
def draw_spiral(turtle):
        sideLength = 2 # Initiate sideLength to 2
        numSides = 100  # We want 100 sides
        for side in range(numSides):
           turtle.forward(sideLength)
           turtle.right(72)
           sideLength += 2

def main():  # define main() function
        screen = turtle.Screen()     # Open a turtle screen
        ted = turtle.Turtle()
        draw_spiral(ted)
        screen.exitonclick()
main()  # call main() function
```

import math # import math module

```
def distance (x1, y1, x2, y2):  # define distance(x1, y1, x2, y2) to calculate radius with arguments
x1, y1, x2, y2
        return math.sqrt((x2 - x1)**2 + (y2 - y1)**2)  # return result of distance formula
sqrt((x2-x1)^2 + (y2-y1)^2)
def perim (r):
        return 2 * math.pi * r
def main():
        x_center = float(input("Enter x for center point: "))  # get float from user for x1
        y_center = float(input("Enter y for center point: "))  # get float from user for y1
        x_perim = float(input("Enter x for perimeter point: "))  # get float from user
        y_perim = float(input("Enter y for perimeter point: "))  # get float from user f

        radius = distance(x_center, y_center, x_perim, y_perim)  # call distance() function
with x1, y1, x2, y2 passed as arguments
        perimeter = perim(radius)  # call radius() function with radius from distance() function
passed as argument

        print(f"Perimeter of circle centered at ({x_center} , {y_center}) with radius {radius}
is {perimeter}") # print with proper formatting

main()  # call main() function


def test_pwd(p): # define test_pwd(p) that takes in a password (string) as an argument
        if len(p) >= 9 :
                if ' ' not in p and '\t' not in p :  # if p does not contain spaces or tabs
                        hasUpper = False
```

```
                hasLower = False
                hasDigit = False
                hasSpecial = False
                for char in p:  # iterate through each character in p
                        if char.isupper():
                                hasUpper = True

                        if char.islower():
                                hasLower = True
                        if char.isdigit():
                                hasDigit = True
                        if not char.isalnum():  # if char is N
alphanumeric (aka a symbol), set hasSpecial to True
                                hasSpecial = True  # n
condition alone would pass with a space or tab, but since we checked for it earlier, if it contains a space or tab, it wouldn't have made it this far
                if hasUpper and hasLower and hasDigit and has
p passes all conditions, return True
                        return True
                else:
                        return False  # if p does not have an uppersc
digit, and special character, return False
        else:
                return False  # if p contains a space or tab, return False
    else:
        return False # if length of p <= 9, return False

def main(): # define main() function
    # print password requirements
    print("\nPassword must have:")
    print("• at least 9 characters")
    print("• no spaces or tabs")
    print("• at least 1 upper-case letter")
    print("• at least 1 lower-case letter")
    print("• at least 1 digit")
    print("• at least 1 special character (not letter, space, digit, or tab)")
    password_attempts = 0  # keep track of password attempts
    valid_passwords = 0  # keep tack of number of valid passwords
    notDone = True  # set to True while user still wants to enter more passwor
    while notDone: # while notDone is True (aka user wants to enter more pas
        password = input("\nEnter password: ") # get password from

        if test_pwd(password) == False:  # if user entered invalid pas
test_pwd returns False)
                print("Sorry, that is not a valid password.")

        else: # if user entered valid password (aka test_pwd returns T
                print("Password OK.")
                valid_passwords += 1  # incrememnt number of
passwords by 1
        password_attempts += 1 # always increment password_attem
after every time test_pwd is called
        invalidInput = True  # True if user enters something other tha
Initiate to True after every password attempt in order to enter while loop below
        while invalidInput :
                again = input("Want to try another password? (y
ask user if they want to try another password
                if again == 'n' : # if user enters 'n', break out of v
and end program by setting notDone and invalidInput to False
                        notDone = False
                        invalidInput = False
                elif again == 'y': # if they want to enter another p
get out of this nested while loop by setting invalidInput to False
                        invalidInput = False
                else:  # user entered something other than 'n' or '
                        print("Please enter y or n")
    # print number of valid passwords and total attempts
    print(f"\nYour score: {valid_passwords} valid passwords in {password_at
tries.")

main() # call main() function
```

algorithm= A description of the steps in a process to compute a solution.

program=coded sequence of steps to be followed exactly. computer=hardware+software. Outside hardware=keyboard,mouse,monitor.inside=CPU,(RAM=volatile storage with faster access usually located off processor chip),bluetooth.software=OS,apps,browser. Use binary to store data, base 2, [N cards count to 0 to 2^N - 1]

Negative numbers in binary are represented - one bit of the number represents the sign

pwd=print working directory, ls=list files in current directory. cd=change directory. cd..=go up to parent directory.//integer divide(no decimals) mix of float and integer gives you float.\n=newline.\t=tab.\"doublequotes.syntax error=mis-formatting.semantic/logical error=no warning message but result doesn't make sense. Runtime error=runs but some other error happens. Parse error=invalid character.Name error=variable not yet defined. Value error=gave function invalid input. If statements check and performs if something is true. A variable is a name that represents a value stored in the computer's memory. For loops:

continue statement=to end current iteration and force python to loop back.

For i in range(7): continue brings it back to for break=to break out a loop and go to code after loop any for loop can be written as a while but not every while can be for

```
print(afjei;)
Continue
print(this statement never happens)
```

```
For i in range(7):
    print(afjei;)
        break
    print(this statement never happens)
```

Turtles: ted=turtle.Turtle() spiral: for i in range(100): ted.forward(5+i) ted.right(15) hex=6.

Name boolean functions with a question word (is)

module=code library that includes functions you might want to use in your code (turtle is a module). Should have functions that are related, follow module theme name. Doesn't normally have main functions but it needs to use the weird syntax if it does.

Strings: treat like lists, index string[6] [0] is first element. Concatenation is when you join two strings next to each other with a +. repetition=(string*3)+(string2*2)=stringstringstringstring2string2

Slicing strings=string[start,end,step]

argument=a value provided to a function during a function call