day 17

```
#Object Oriented Programming

from turtle import Turtle, Screen

timmy = Turtle()
print(timmy)
timmy.shape("turtle")
timmy.color("pink")
timmy.forward(100)
timmy.left(200)
timmy.forward(100)
timmy.right(200)
timmy.left(200)
timmy.right(100)
timmy.forward(100)
timmy.right(200)
timmy.forward(100)
timmy.right(200)
timmy.forward(100)
timmy.left(200)

my_screen = Screen()
print(my_screen.canvheight)
```

day 18

```
#Creating a class in python
class User:
def __init__(self, user_id, username):
    self.id   = user_id
    self.username = username
    self.followers = 0
    self.following = 0
def follow(self, user):
    user.followers += 1
    self.following += 1
user_1 = User("001", "DeRay")
user_2 = User("002", "Morgan")
print(user_1.followers)
user_1.follow(user_2)
print(user_1.followers)
print(user_1.following)
print(user_2.followers)
```

```
print(user_2.following)
```

day 19

```
import turtle as t
import random
tim = t.Turtle()
t.colormode(255)
def random_color():
r = random.randint(0, 255)
g = random.randint(0, 255)
b = random.randint(0, 255)
random_color = (r, g, b)
return random_color
colors = ["red","purple","black","green","yellow","gray","brown"]
direction = [0, 90, 180, 360]
tim.pen-size(15)
tim.speed(1)

for _ in range(200):
tim.color(random.choice(colors))
tim.forward(30)
tim.set-heading(random.choice(direction))

#Turtle Square
# print(another.another_variable)
from turtle import Turtle, Screen
kate = Turtle()
print(kate)
kate.shape("turtle")
kate.color("red")
#turtle will move in square dimension
my_screen = Screen()
kate.forward(200)
kate.right(90)
kate.forward(200)
kate.right(90)
kate.forward(200)
kate.right(90)
kate.forward(200)
kate.right(90)
for a in range(4):
 Kate.forward(200)
 Kate.right(90)
```

```
print(my_screen.canvwidth)
```

 # day 20

```
#Turtle race game

from turtle import Turtle, Screen
tim = Turtle()
screen = Screen()

def move_forwards():
    tim.forward(10)

def move_backwards():
    tim.backward(10)

def turn_left():
    new_heading =tim.heading() + 10
    tim.setheading(new_heading)

def turn_right():
    new_heading = tim.heading() - 10
    tim.setheading(new_heading)
def clear():
    tim.clear()
    tim.home()
screen.listen()
screen.onkey(move_forwards, "w")
screen.onkey(move_backwards, "s")
screen.onkey(turn_left, "l")
screen.onkey(turn_right, "r")
# screen.onkey(key="space", fun= move_forwards)
screen.exitonclick()
```

 # day 21

```
#Snake Game - Controlling the snake
from turtle import   Screen
from snake import Snake
from food import Food
from scoreboard import Scoreboard
import time
```

```
screen = Screen()
screen.setup(width=600, height=600)
screen.bicolor("black")
screen.title("My Snake Game")
screen.tracer(0)
snake = Snake()
food = Food()
scoreboard = Scoreboard()
screen.listen()
screen.onkey(snake.up, "Up")
screen.onkey(snake.down, "Down")
screen.onkey(snake.left, "Left")
screen.onkey(snake.right, "Right")


game_is_on = True
while game_is_on:
    screen.update()
    time.sleep(0.1)
    snake.move()

#Detect collision with food
if snake.head.distance(food) < 10:
        food.refresh()
        snake.extend()
    scoreboard.increase_score()

#Detect collision with wall.
if snake.head.xcor() > 288 or snake.head.xcor() < -280 or snake.head.ycor() > 280
or snake.head.xcor() < -280:
        game_is_on = False
    scoreboard.game_over()

#Detect collision with tail
for segment in snake.segments[1:]:
    if snake.head.distance(segment) < 10:
            game_is_on = False
        scoreboard.game_over()


screen.exitonclick()
```

day 22

```
#Class Inheritance
class Animal:
def __init__(self):
self.num_eyes = 2
def breathe(self):
print("Inhale, exhale")

class Fish(Animal):
def __init__(self):
super().__init__()

def breathe(self):
super().breathe()
print("Doing this underwater")

def swim(self):
print("moving in water")
nem = Fish()
nem.swim()
nem.breathe()
print(nem.num_eyes)
```

 # day 23

```
#Building the Pong Arcade Game
from turtle import Screen, Turtle
from paddle import Paddle
from ball import Ball
from scoreboard import Scoreboard
import time
screen = Screen()
screen.bgcolor("blue")
screen.setup(width=800, height = 600)
screen.title("Pong Game")
screen.tracer(0)

r_paddle = Paddle((350, 0))
l_paddle = Paddle((-350, 0))
ball = Ball()
scoreboard = Scoreboard()

screen.listen()
```

```
screen.onkey(r_paddle.go_up, "Up")
screen.onkey(r_paddle.go_down, "Down")
screen.onkey(l_paddle.go_up, "w")
screen.onkey(l_paddle.go_down, "s")

game_is_on = True
while game_is_on:
time.sleep(ball.move_speed)
screen.update()
ball.move()
if ball.ycor() > 280 or ball.ycor() < -280:
ball.bounce_y()
if ball.distance(r_paddle) < 50 and ball.xcor() > 320 or ball.distance(l_paddle) < 50
and ball.xcor() < -320:
ball.bounce_x()

if ball.xcor() > 380:
ball.reset_position()
scoreboard.l_point()
if ball.xcor() < -380:
ball.reset_position()
scoreboard.r_point()

screen.exitonclick()
```