**DAY 15**

# Setting up Pycharm Development Environment

#Coffee Project

MENU = {

"espresso": {

"ingredients": {

"water": 58,

"coffee": 18,

},

"cost": 1.5,

},

"latte": {

"ingredients": {

"water": 200,

"milk": 150,

"coffee": 24,

},

"cost": 2.5,

},

"cappuccino": {

"ingredients": {

"water": 250,

"milk": 100,

"coffee": 24,

},

"cost": 3.0,

}

}

profit = 0

resources = {

"water": 300,

"milk": 200,

"coffee": 100,

}

def is\_resource\_sufficient(order\_ingredients):

for item in order\_ingredients:

if order\_ingredients[item] >= resources[item]:

print(f"Sorry, there is not {item}.")

return False

return True

def process\_coins():

"""Returns the total calculated from coins inserted"""

print("Please insert coins.")

total = int(input("how many quarters?: ")) \* 0.25

total += int(input("how many dimes?: ")) \* 0.1

total += int(input("how many nickles?: ")) \* 0.05

total += int(input("how many pennies?: ")) \* 0.01

return total

def is\_transaction\_successful(money\_received, drink\_cost):

"""Return true when the payment is accepted, or False if money is insufficient."""

if money\_received >= drink\_cost:

change = round(money\_received - drink\_cost, 2)

print(f"Here is ${change} in change.")

global profit

profit += drink\_cost

return True

else:

print("Sorry, there is not enough money. Money refunded")

return False

def make\_coffee(drink\_name, order\_ingredients):

"""Deduct the required ingredient from the resources."""

for item in order\_ingredients:

resources[item] -= order\_ingredients[item]

print(f"Here is your {drink\_name}")

is\_on = True

while is\_on:

choice = input("What would you like? (espresso/latte/cappuccino): ")

if choice == "off":

is\_on = False

elif choice == "report":

print(f"Water: {resources['water']}ml")

print(f"Milk: {resources['milk']}ml")

print(f"Coffee: {resources['coffee']}g")

print(f"Money: ${profit}")

else:

drink = MENU[choice]

if is\_resource\_sufficient(drink["ingredients"]):

payment = process\_coins()

if is\_transaction\_successful(payment, drink["cost"]):

make\_coffee(choice, drink["ingredients"])

**DAY 16**

#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 17**

from Day17 import Question

from Data import question\_data

from Quiz\_brain import QuizBrain

question\_bank = []

for question in question\_data:

question\_text = question["text"]

question\_answer = question["answer"]

new\_question = Question(question\_text, question\_answer)

question\_bank.append(new\_question)

quiz = QuizBrain(question\_bank)

while quiz.still\_has\_questions():

quiz.next\_question()

print("You have completed the quiz")

print(f"Your final score was: {quiz.score}/{quiz.question\_number}")

**DAY 18**

#Day 18

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.pensize(15)

tim.speed(1)

for \_ in range(200):

tim.color(random.choice(colors))

tim.forward(30)

tim.setheading(random.choice(direction))

**DAY 19**

#Day19 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 20**

#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.bgcolor("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 21**

#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")

nemo = Fish()

nemo.swim()

nemo.breathe()

print(nemo.num\_eyes)