Day - 20

**Snake Game: Part 1**

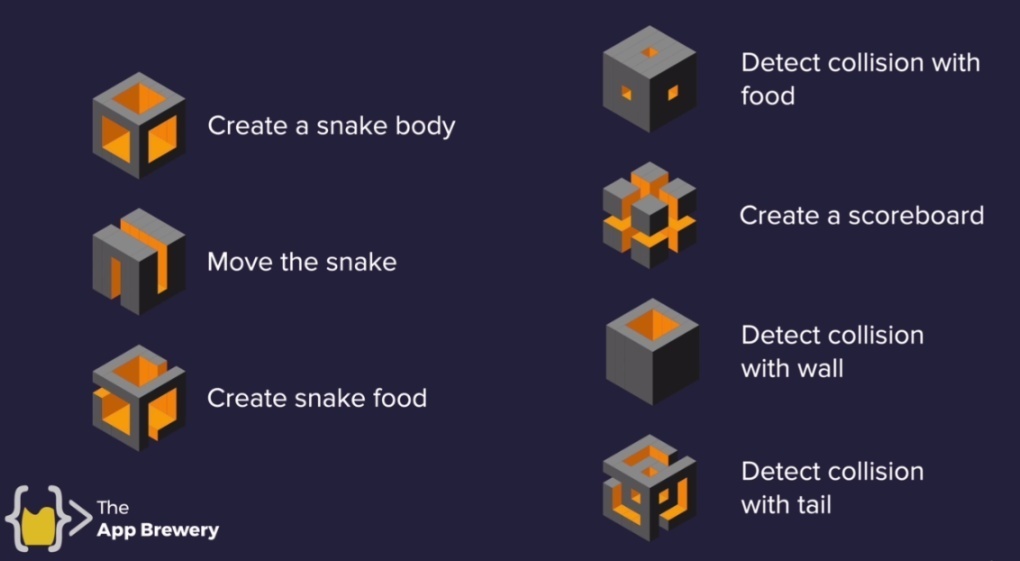
**Animation & Coordinates**

Data Types, Numbers, Operators, Type Conversion, f-strings

for loop with start, stop and step attributes

**20.1 The NOKIA snake game**

* Divided into 7 parts. Day 20: 3 parts and day 21: 4 parts



|  |  |
| --- | --- |
| Day 20 | Day 21 |
| 1. Creating a Snake Body by "adding 3 squares" 2. Move the snake 3. Create snake food | 1. Detect collision with food 2. Create a scoreboard 3. Detect collision with wall 4. Detect collision with tail |

**20.2 Screen Setup and Creating a Snake Body**

|  |  |
| --- | --- |
|  | **import** turtle  scr = turtle**.Screen**()  #*resizing screen*  scr**.setup**(width = 800, height = 700)  #*Setting screen color*  scr**.bgcolor**("#a8c64e")  #*Showing title of the sreen*  scr**.title**(" ---- The famous NOKIA 1100 snake game ---- ")  star\_positions = [(10, 0), (0, 0), (-10, 0)]  snake = []  **for** pos **in** star\_positions:      sqrs = turtle**.Turtle**(shape = "square")      sqrs**.shapesize**(0.4, 0.4, 0)      sqrs**.color**("#3c412c")      sqrs**.penup**()      sqrs**.goto**(pos)          #*By default turtle size is 20 X 20.*          #*So ours will be 8 X 8.*          #*Distance from each other will be 10*      snake**.append**(sqrs)  #*Screen doesn't disappear autometically*  scr**.exitonclick**()  #*python snake.py* |

**20.3 Animation controlling**

* ***tracer(): turtle.tracer***(***n***=None, ***delay***=None). Both parameters are ***int***.
* Turn turtle animation on/off and set delay for update drawings. If n is given, only each ***n-th regular screen update*** is really performed. (Can be used to accelerate the drawing of complex graphics.) When called without arguments, returns the currently stored value of ***n***. Second argument sets delay value (see ***delay()***).
* ***delay(): turtle.delay(delay=None)*** delay is positive integer
* Set or return the drawing delay in milliseconds.
* ***update():*** **turtle.update()** Perform a ***TurtleScreen*** update. To be used when tracer is turned off.
* NOTE: Although above methods are acting on " ***turtle*** " but it actually woks on "***turtle.Screen()"***.
* The Time module:
* To control the animation time : ***import time***
* And then use for 1 sec or 0.1 or any others, ***time.sleep(1)***

**import** turtle

#*time module for delay the time*

**import** time

scr = turtle**.Screen**()

#*resizing screen*

scr**.setup**(width = 800, height = 700)

#*Setting screen color*

scr**.bgcolor**("#a8c64e")

#*Showing title of the sreen*

scr**.title**(" ---- The famous NOKIA 1100 snake game ---- ")

star\_positions = [(10, 0), (0, 0), (-10, 0)]

#*Turnig off the animation by "tracer" off with "delay" setting*

scr**.tracer**(n = 0)

#*delay animation*

#*scr.delay(30)*

snake = []

**for** pos **in** star\_positions:

    sqrs = turtle**.Turtle**(shape = "square")

    sqrs**.shapesize**(0.4, 0.4, 0)

    sqrs**.color**("#3c412c")

    sqrs**.penup**()

    sqrs**.\_tracer**(flag=**None**)

    sqrs**.goto**(pos)

        #*By default turtle size is 20 X 20.*

        #*So ours will be 8 X 8.*

        #*Distance from each other will be 10*

    snake**.append**(sqrs)

#*update the screen after loading the whole snake*

scr**.update**()

game\_on = **True**

**while** game\_on:

    #*moving the snake segments*

    #*first it will move like a Caterpillar.*

    #*Need to use "update" and disable "tracer"*

    #*update screen here when al snake is moved*

    scr**.update**()

    time**.sleep**(1)

**for** sq\_r **in** snake:

        sq\_r**.forward**(5)

        #*Shows one by one move "scr.update()"*

        #*set higher value with higher "delay" to understand*

        #*Shows one by one move slowly: time.sleep(1)*

        #*stop the snake at boundary*

**if** snake[0]**.position**()[0] **>=** 380:

        game\_on = **False**

        #*update "turtle-object/square-segments" for animation when "tracer" is off*

        #*scr.update()*

#*Screen doesn't disappear autometically*

scr**.exitonclick**()

#*python snake.py*

**20.4 Advanced for – loop: Turning the turtle (modify lopp)**

Modifying the above code for better performance and animation for TURN

* FOR loop with Start, Stop and Step:

**for** i **in** **range**(start\_value, stop\_value, step):

* However for loop doesn't support key-vale arguments. So **for** i **in** **range**(start= 3, stop= 0, step=-1): not goanna work.

**Animating the Snake Segments on Screen**

No-OOP yet : Turn the SNAKE

**import** turtle

#*time module for delay the time*

**import** time

scr = turtle**.Screen**()

#*resizing screen*

scr**.setup**(width = 800, height = 700)

#*Setting screen color*

scr**.bgcolor**("#a8c64e")

#*Showing title of the sreen*

scr**.title**(" ---- The famous NOKIA 1100 snake game ---- ")

start\_positions = [(10, 0), (0, 0), (-10, 0)]

#*Turnig off the animation by "tracer" off with "delay" setting*

scr**.tracer**(n = 0)

snake = []

**for** pos **in** start\_positions:

    sqrs = turtle**.Turtle**(shape = "square")

    sqrs**.shapesize**(0.4, 0.4, 0)

    sqrs**.color**("#3c412c")

    sqrs**.penup**()

    sqrs**.\_tracer**(flag=**None**)

    sqrs**.goto**(pos)

        #*By default turtle size is 20 X 20. So ours will be 8 X 8. Distance from each other will be 10.*

    snake**.append**(sqrs)

#*update the screen after loading the whole snake*

scr**.update**()

game\_on = **True**

**while** game\_on:

    #*moving the snake segments. first it will move like a Caterpillar. Need to use "update" and disable "tracer". Update screen here when all snake is moved*

    scr**.update**()

    time**.sleep**(1)

    #*Reverse for loop. Move following/backward segments to forward.*

**for** i **in** **range**(len(snake)-1, 0, -1):

        new\_x = snake[i-1]**.xcor**()

        new\_y = snake[i-1]**.ycor**()

        snake[i]**.goto**(new\_x, new\_y)

        #*Shows one by one move "scr.update()". set higher value with higher "delay" to understand. Shows one by one move slowly: time.sleep(1).*

    #*moving the first segment*

    snake[0]**.forward**(10)

    # snake[0]**.left**(90)

    # snake[0]**.forward**(10)

    #*Stop the snake at boundary.*

**if** snake[0]**.position**()[0] **>=** 380:

        game\_on = **False**

#*Screen doesn't disappear autometically*

scr**.exitonclick**()

#*python snake.py*

**20.5 Create a Snake Class & Move to OOP**

|  |  |
| --- | --- |
| **import** turtle  **import** snake  #*time module for delay the time*  **import** time  scr = turtle**.Screen**()  #*resizing screen. Setting screen color. Showing title of the sreen*  scr**.setup**(width = 800, height = 700)  scr**.bgcolor**("#a8c64e")  scr**.title**(" ---- The famous NOKIA 1100 snake game ---- ")  #*Turnig off the animation by "tracer" off with "delay" setting*  scr**.tracer**(n = 0)  #*create a snake object*  snAKe = snake**.Snake**()  snAKe**.create\_sanke**()  #*update the screen after loading the whole snake*  scr**.update**()  game\_on = **True**  **while** game\_on:      #*moving the snake segments. first it will move like a Caterpillar. Need to use "update" and disable "tracer". Update screen here when all snake is moved*      scr**.update**()      time**.sleep**(1)        snAKe**.move**()      #*snake[0].left(90)*      #*Stop the snake at boundary.*  **if** snAKe**.**snake[0]**.position**()[0] **>=** 380:          game\_on = **False**    #*Screen doesn't disappear autometically*  scr**.exitonclick**()  #*python*snake\_main.py | **import** turtle  STARTING\_POSITION = [(10, 0), (0, 0), (-10, 0)]  MOVE\_DISTANCE = 10  **class** Snake:  **def** **\_\_init\_\_**(self):  **self.**snake = []  **self.create\_sanke**()    **def** **create\_sanke**(self):  **for** pos **in** STARTING\_POSITION:              sqrs = turtle**.Turtle**(shape = "square")              sqrs**.shapesize**(0.4, 0.4, 0)              sqrs**.color**("#3c412c")              sqrs**.penup**()              sqrs**.goto**(pos)                  #*By default turtle size is 20 X 20. So ours will be 8 X 8. Distance from each other will be 10.*  **self.**snake**.append**(sqrs)  **def** **move**(self):          #*Reverse for loop. Move following/backward segments to forward.*  **for** i **in** **range**(len(**self.**snake)-1, 0, -1):              new\_x = **self.**snake[i-1]**.xcor**()              new\_y = **self.**snake[i-1]**.ycor**()  **self.**snake[i]**.goto**(new\_x, new\_y)              #*Shows one by one move "scr.update()". set higher value with higher "delay" to understand. Shows one by one move slowly: time.sleep(1).*          #*moving the first segment*  **self.**snake[0]**.forward**(MOVE\_DISTANCE)          #*snake[0].left(90)*  #*python*snake.py |

**20.5 How to Control the SNAKE with a Key-press**

|  |  |
| --- | --- |
| **import** turtle  **import** snake  #*time module for delay the time*  **import** time  scr = turtle**.Screen**()  #*resizing screen. Setting screen color. Showing title of the sreen*  scr**.setup**(width = 800, height = 700)  scr**.bgcolor**("#a8c64e")  scr**.title**(" ---- The famous NOKIA 1100 snake game ---- ")  #*Turnig off the animation by "tracer" off with "delay" setting*  scr**.tracer**(n = 0)  #*create a snake object*  snAKe = snake**.Snake**()  #*update the screen after loading the whole snake*  scr**.update**()  scr**.listen**()  scr**.onkey**(snAKe**.**up, "Up")  scr**.onkey**(snAKe**.**down, "Down")  scr**.onkey**(snAKe**.**left, "Left")  scr**.onkey**(snAKe**.**right, "Right")  game\_on = **True**  **while** game\_on:      #*moving the snake segments. first it will move like a Caterpillar. Need to use "update" and disable "tracer". Update screen here when all snake is moved*      scr**.update**()      time**.sleep**(0.2)        snAKe**.move**()      #*snake[0].left(90)*      #*Stop the snake at boundary.*  **if** snAKe**.**snake[0]**.position**()[0] **>=** 380:          game\_on = **False**    #*Screen doesn't disappear autometically*  scr**.exitonclick**()  #*python snake\_main.py* | **import** turtle  STARTING\_POSITION = [(10, 0), (0, 0), (-10, 0)]  MOVE\_DISTANCE = 10  UP = 90  DOWN = 270  LEFT = 180  RIGHT = 0  **class** Snake:  **def** **\_\_init\_\_**(self):  **self.**snake = []  **self.create\_sanke**()  **self.**head = **self.**snake[0] #*This line is called after create\_sanke(), unless error occurs*    **def** **create\_sanke**(self):  **for** pos **in** STARTING\_POSITION:              sqrs = turtle**.Turtle**(shape = "square")              sqrs**.shapesize**(0.4, 0.4, 0)              sqrs**.color**("#3c412c")              sqrs**.penup**()              sqrs**.goto**(pos)                  #*By default turtle size is 20 X 20. So ours will be 8 X 8. Distance from each other will be 10.*  **self.**snake**.append**(sqrs)  **def** **move**(self):          #*Reverse for loop. Move following/backward segments to forward.*  **for** i **in** **range**(len(**self.**snake)-1, 0, -1):              new\_x = **self.**snake[i-1]**.xcor**()              new\_y = **self.**snake[i-1]**.ycor**()  **self.**snake[i]**.goto**(new\_x, new\_y)              #*Shows one by one move "scr.update()". set higher value with higher "delay" to understand. Shows one by one move slowly: time.sleep(1).*          #*moving the first segment*  **self.**head**.forward**(MOVE\_DISTANCE)  **def** **up**(self):  **if** **self.**head**.heading**() **!=** DOWN:  **self.**head**.setheading**(UP)  **def** **down**(self):  **if** **self.**head**.heading**() **!=** UP:  **self.**head**.setheading**(DOWN)  **def** **left**(self):  **if** **self.**head**.heading**() **!=** RIGHT:  **self.**head**.setheading**(LEFT)  **def** **right**(self):  **if** **self.**head**.heading**() **!=** LEFT:  **self.**head**.setheading**(RIGHT)  #*python snake.py* |