# Ministry of Education of Republic of Moldova Technical University of Moldova

Report

## on Discrete Math

Laboratory Work Nr. 1



**Performed by** b. **Cristian Brinza FAF-212**

**Verified by** pr.univ **Cristifor Fiștic**

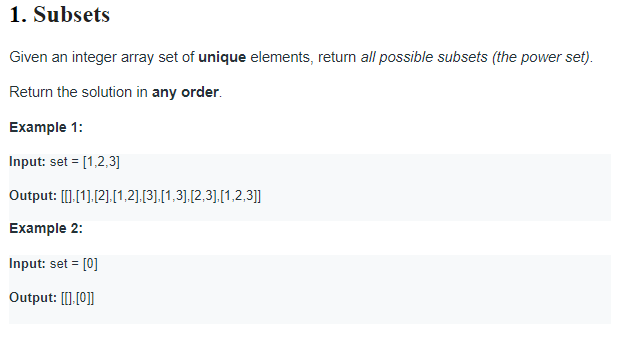
# Chișinău 2021

*Topic:*  Programming discrete math theme in Python programming language.

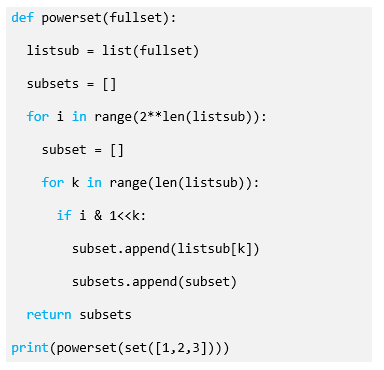
**Purpose of the laboratory work:** Accumulation of practical skills for developing and programming computational processes and program testing skills.

1. **Subsets**

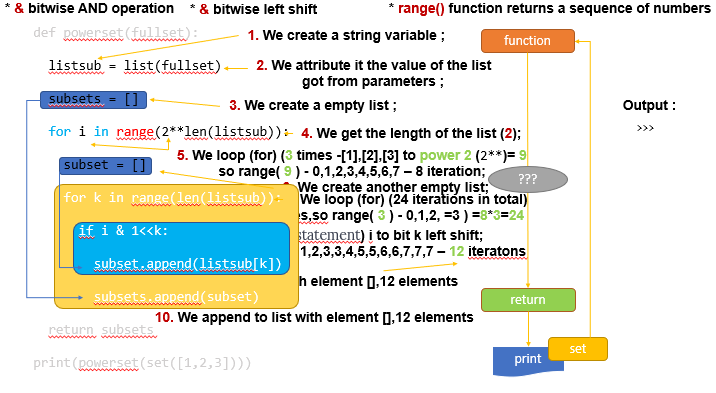
* **Condition**

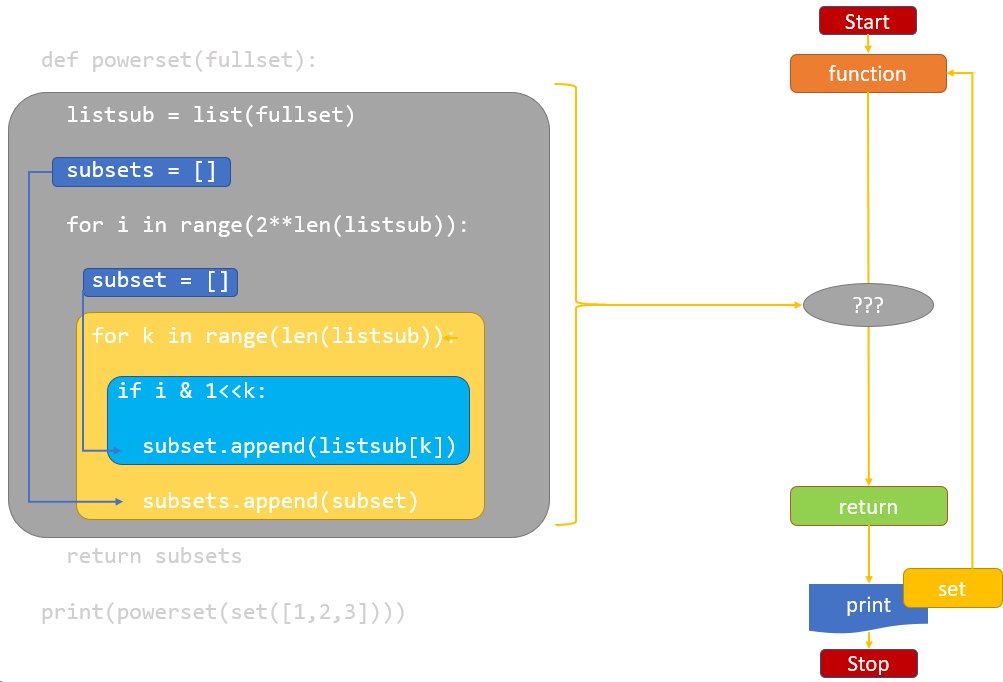


* **Code:**

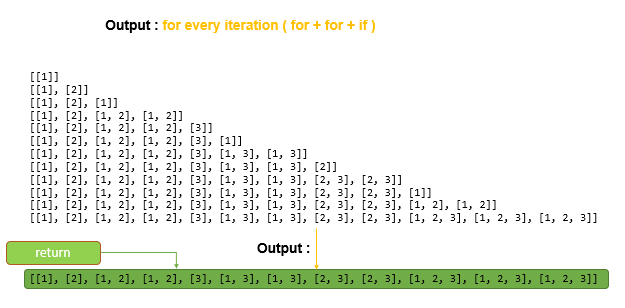


* **Comented code:**



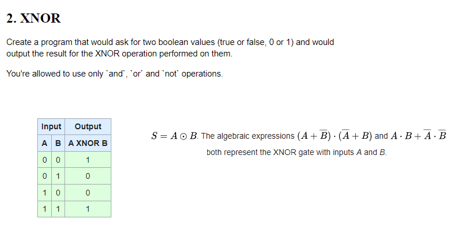


* **Output:**

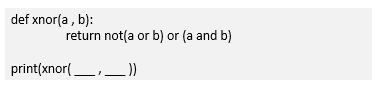


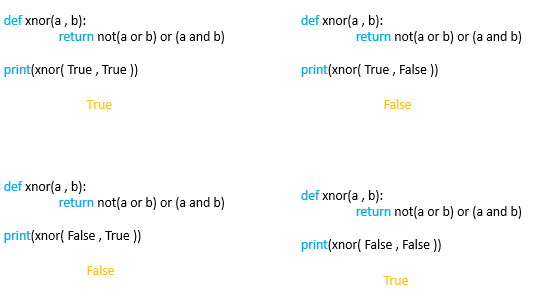
1. **XNOR**

* **Condition**



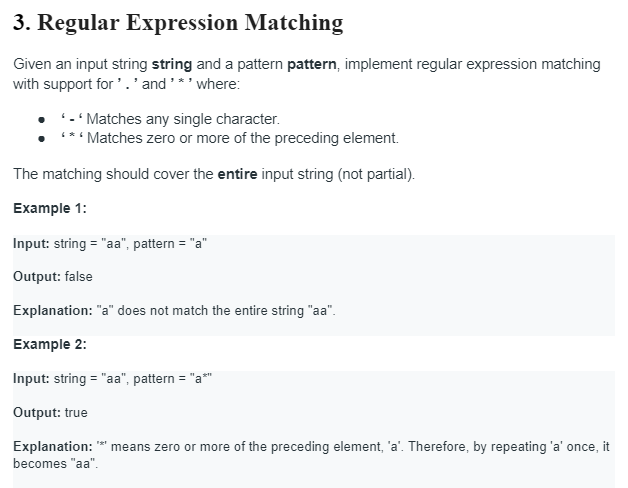
* **Code:**



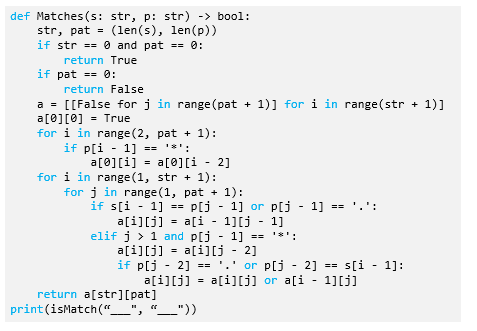


1. **REM**

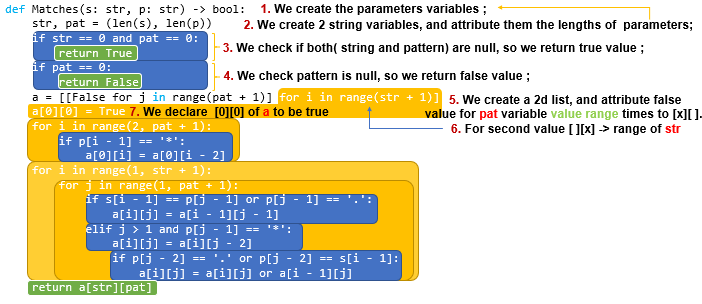
* **Condition**



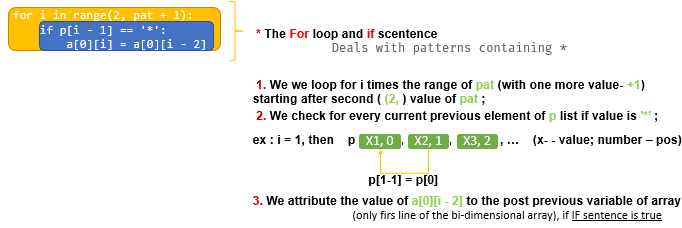
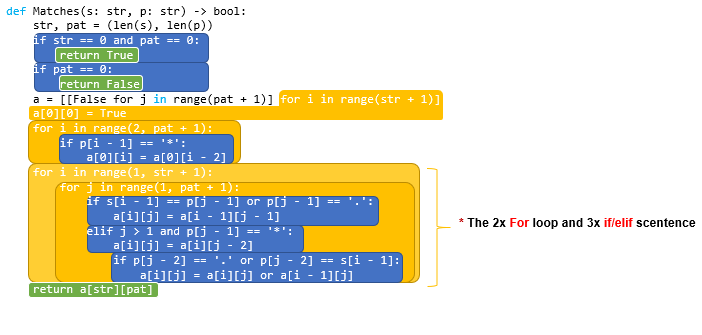
* **Code:**

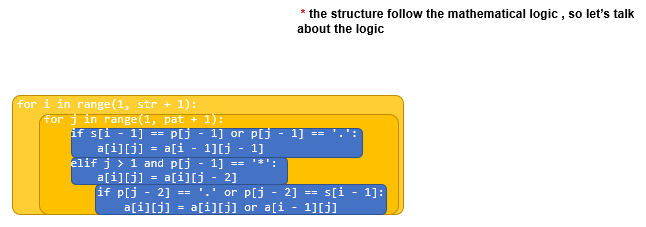
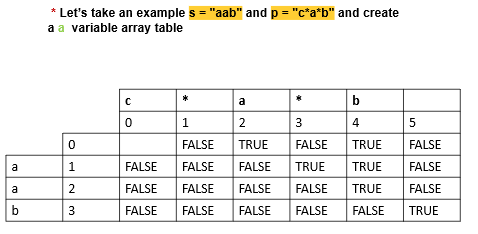


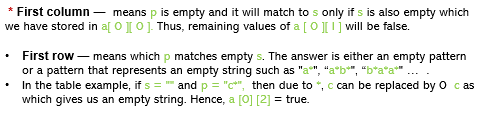
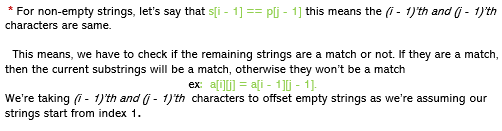
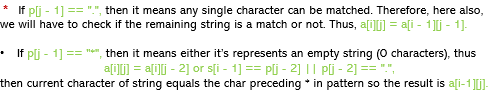
* **Comented code:**



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

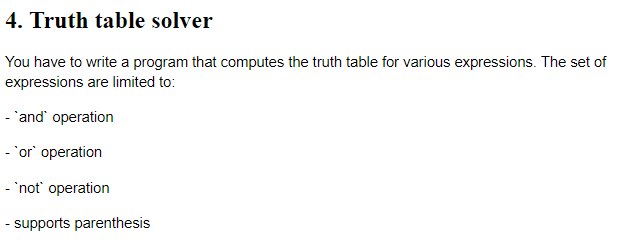
 

1. **TTS**

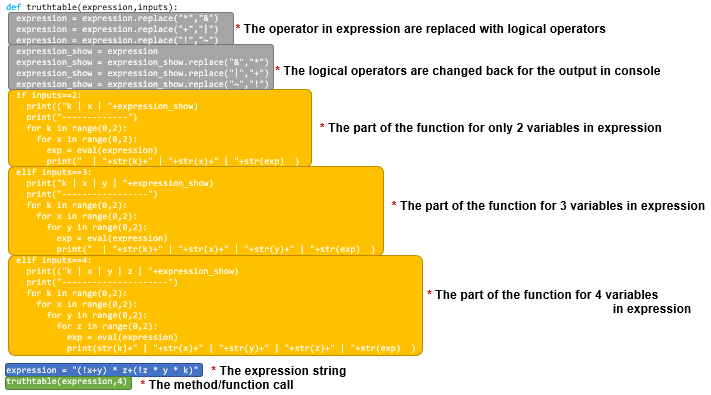
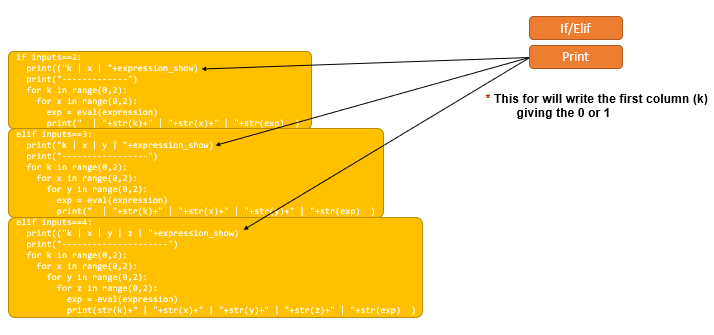
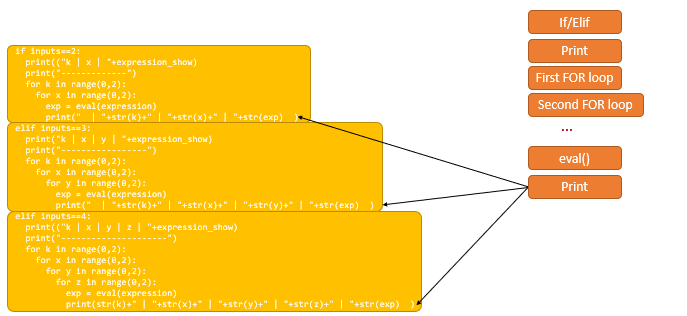
* **Condition**



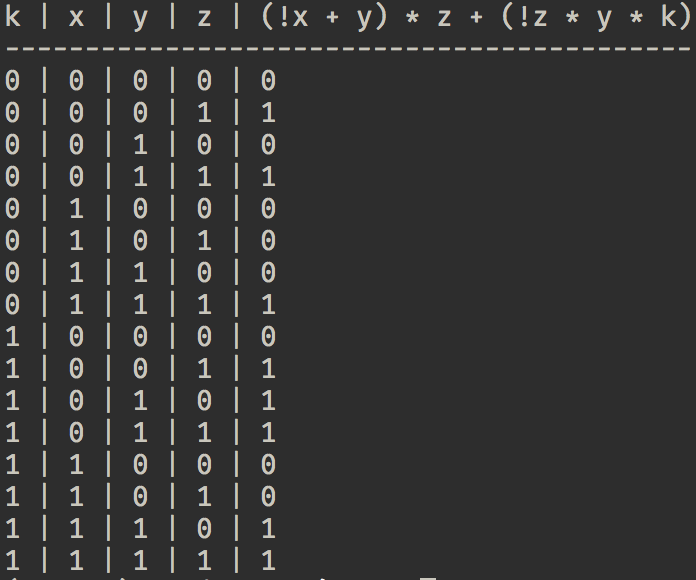
* **Code:**



* **Comented code:**

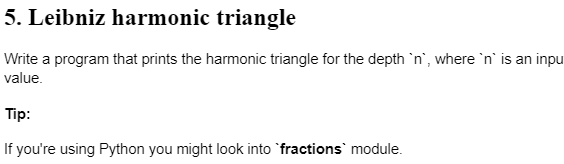
  

* **Output:**

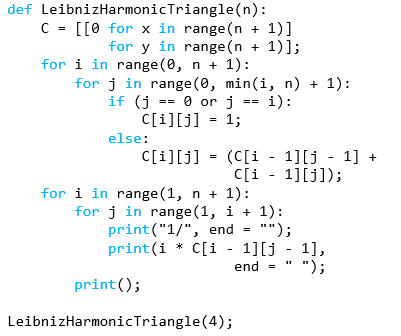


1. **LHT**

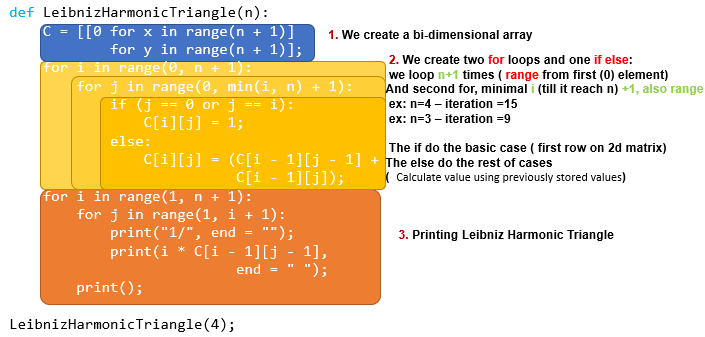
* **Condition**



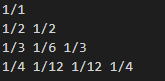
* **Code:**



* **Comented code:**

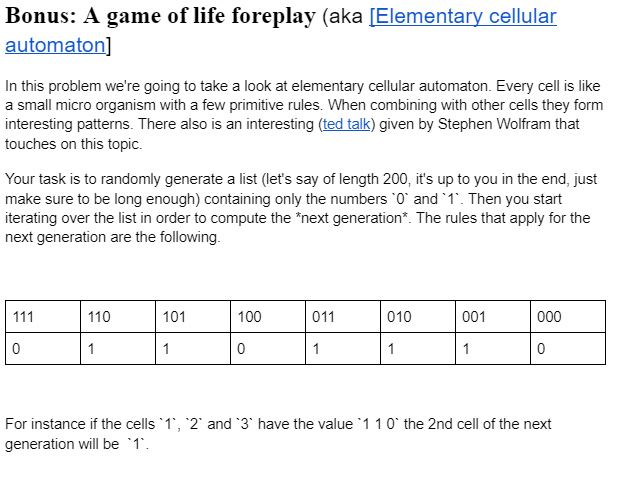


* **Output:**



1. **Bonus**

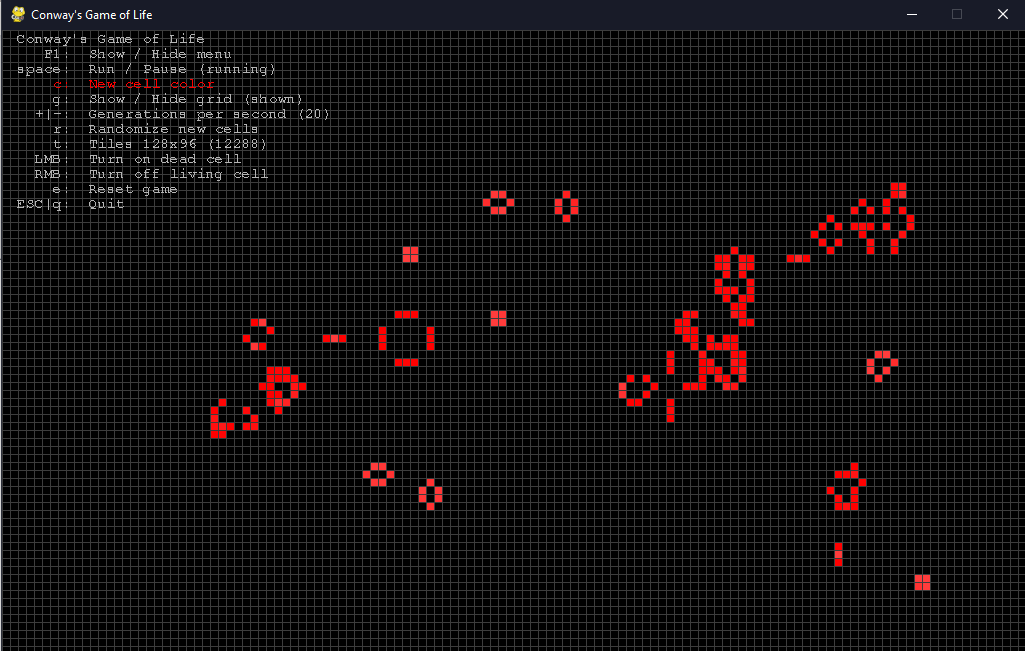
* **Condition**



* **Code:**
* import sys
* from itertools import cycle
* from random import random
* import pygame as pg
* # Colors
* BLACK = (0, 0, 0)
* GREY = (60, 60, 60)
* LIGHT\_GREY = (180, 180, 180)
* RED = (255, 0, 0)
* GREEN = (0, 255, 0)
* YELLOW = (255, 255, 0)
* BLUE = (0, 0, 255)
* MAGENTA = (255, 0, 255)
* CYAN = (0, 255, 255)
* WHITE = (255, 255, 255)
* CELL\_COLORS = [RED, GREEN, YELLOW, BLUE, MAGENTA, CYAN]
* # Fonts
* FONTSIZE = 15
* FONT = 'courier new'
* # Game settings
* WIDTH = 1024
* HEIGHT = 768
* FPS = 60
* TITLE = "Conway's Game of Life"
* TILESIZES = cycle([8, 16, 32, 64])
* TILESIZE = next(TILESIZES)
* GENERATIONS\_PER\_SECOND = 20
* RANDOM\_CHANCE\_TO\_ALIVE\_CELL = 0.3
* # Mouse buttons
* LEFT = 0
* RIGHT = 2
* class Cell(pg.sprite.Sprite):
* def \_\_init\_\_(self, game, x, y):
* self.groups = game.all\_sprites
* pg.sprite.Sprite.\_\_init\_\_(self, self.groups)
* self.image = pg.Surface((TILESIZE, TILESIZE))
* self.rect = self.image.get\_rect()
* self.rect.x = x \* TILESIZE
* self.rect.y = y \* TILESIZE
* self.off()
* def off(self, color=BLACK):
* self.alive = False
* self.image.fill(color)
* def on(self, color=WHITE):
* self.alive = True
* self.image.fill(color)
* self.color = color
* def survive(self):
* r, g, b = self.color
* if r != 255:
* r += 5
* if g != 255:
* g += 5
* if b != 255:
* b += 5
* self.color = (r, g, b)
* self.image.fill(self.color)
* class Game:
* def \_\_init\_\_(self):
* pg.init()
* pg.font.init()
* self.screen = pg.display.set\_mode((WIDTH, HEIGHT))
* pg.display.set\_caption(TITLE)
* self.clock = pg.time.Clock()
* pg.key.set\_repeat(500, 100)
* self.new\_generation\_event = pg.USEREVENT+1
* pg.time.set\_timer(self.new\_generation\_event, int(1000/GENERATIONS\_PER\_SECOND))
* self.menu\_font = pg.font.SysFont(FONT, FONTSIZE)
* def new(self):
* self.gridwidth = int(WIDTH / TILESIZE)
* self.gridheight = int(HEIGHT / TILESIZE)
* self.pause = True
* self.show\_menu = True
* self.show\_grid = True
* self.colors = cycle(CELL\_COLORS)
* self.color = next(self.colors)
* self.gps = GENERATIONS\_PER\_SECOND
* self.all\_sprites = pg.sprite.Group()
* self.cells = []
* for x in range(self.gridwidth):
* self.cells.append([])
* for y in range(self.gridheight):
* self.cells[x].append(Cell(self, x, y))
* self.previous\_click, self.previous\_x, self.previous\_y = None, None, None
* def run(self):
* while True:
* self.clock.tick(FPS)
* self.events()
* self.draw()
* def quit(self):
* pg.quit()
* sys.exit()
* def draw\_grid(self):
* for x in range(0, WIDTH, TILESIZE):
* pg.draw.line(self.screen, GREY, (x, 0), (x, HEIGHT))
* for y in range(0, HEIGHT, TILESIZE):
* pg.draw.line(self.screen, GREY, (0, y), (WIDTH, y))
* def blit(self, position, text, color=LIGHT\_GREY):
* self.screen.blit(self.menu\_font.render(text, False, color), (FONTSIZE, FONTSIZE\*position))
* def draw\_menu(self):
* self.blit(0, f"{TITLE}")
* self.blit(1, f"   F1:  Show / Hide menu")
* self.blit(2, f"space:  Run / Pause {'(paused)' if self.pause else '(running)'}")
* self.blit(3, f"    c:  New cell color", self.color)
* self.blit(4, f"    g:  Show / Hide grid {'(shown)' if self.show\_grid else '(hidden)'}")
* self.blit(5, f"  +|-:  Generations per second ({self.gps})")
* self.blit(6, f"    r:  Randomize new cells")
* self.blit(7, f"    t:  Tiles {self.gridwidth}x{self.gridheight} ({self.gridwidth\*self.gridheight})")
* self.blit(8, f"  LMB:  Turn on dead cell")
* self.blit(9, f"  RMB:  Turn off living cell")
* self.blit(10, f"    e:  Reset game")
* self.blit(11, f"ESC|q:  Quit")
* self.blit(int(HEIGHT/FONTSIZE) - 1, f"Developed by Yaniv Rozenboim")
* def draw(self):
* self.all\_sprites.draw(self.screen)
* if self.show\_grid:
* self.draw\_grid()
* if self.show\_menu:
* self.draw\_menu()
* pg.display.flip()
* def randomize(self, chance\_for\_alive\_cell=RANDOM\_CHANCE\_TO\_ALIVE\_CELL):
* for x in range(self.gridwidth):
* for y in range(self.gridheight):
* if random() < chance\_for\_alive\_cell:
* self.cells[x][y].on(self.color)
* else:
* self.cells[x][y].off()
* def new\_generation(self):
* temp = []
* for x in range(self.gridwidth):
* temp.append([])
* for y in range(self.gridheight):
* prev\_x = x-1
* prev\_y = y-1
* next\_x = (x+1) % self.gridwidth
* next\_y = (y+1) % self.gridheight
* value = \
* self.cells[prev\_x][prev\_y].alive + \
* self.cells[prev\_x][y].alive + \
* self.cells[prev\_x][next\_y].alive + \
* self.cells[x][prev\_y].alive + \
* self.cells[x][next\_y].alive + \
* self.cells[next\_x][prev\_y].alive + \
* self.cells[next\_x][y].alive + \
* self.cells[next\_x][next\_y].alive
* if self.cells[x][y].alive:
* if value < 2 or value > 3:
* temp[x].append(False)
* else:
* temp[x].append(True)
* else:
* if value == 3:
* temp[x].append(True)
* else:
* temp[x].append(False)
* for x in range(self.gridwidth):
* for y in range(self.gridheight):
* if temp[x][y]:
* if self.cells[x][y].alive:
* self.cells[x][y].survive()
* else:
* self.cells[x][y].on(self.color)
* else:
* if self.cells[x][y].alive:
* self.cells[x][y].off(BLACK)
* def events(self):
* for event in pg.event.get():
* if event.type == pg.QUIT:
* self.quit()
* if event.type == pg.KEYDOWN:
* if event.key == pg.K\_ESCAPE or event.key == pg.K\_q:
* self.quit()
* if event.key == pg.K\_SPACE:
* self.pause = not(self.pause)
* if event.key == pg.K\_F1:
* self.show\_menu = not(self.show\_menu)
* if event.key == pg.K\_g:
* self.show\_grid = not(self.show\_grid)
* if event.key == pg.K\_r:
* self.randomize()
* if event.key == pg.K\_c:
* self.color = next(self.colors)
* if event.key == pg.K\_e:
* self.new()
* if event.key == pg.K\_t:
* global TILESIZE
* TILESIZE = next(TILESIZES)
* self.new()
* if event.unicode == "+":
* if self.gps < FPS/2:
* self.gps += 1
* pg.time.set\_timer(self.new\_generation\_event, int(1000/self.gps))
* if event.unicode == "-":
* if self.gps > 1:
* self.gps -= 1
* pg.time.set\_timer(self.new\_generation\_event, int(1000/self.gps))
* click = pg.mouse.get\_pressed()
* x, y = pg.mouse.get\_pos()
* x = int(x / TILESIZE)
* y = int(y / TILESIZE)
* if (click, x, y) != (self.previous\_click, self.previous\_x, self.previous\_y):
* self.previous\_click, self.previous\_x, self.previous\_y = click, x, y
* if click[LEFT] and not self.cells[x][y].alive:
* self.cells[x][y].on(self.color)
* elif click[RIGHT] and self.cells[x][y].alive:
* self.cells[x][y].off(BLACK)
* if event.type == self.new\_generation\_event and not self.pause:
* self.new\_generation()
* g = Game()
* while True:
* g.new()
* g.run()
* **Comented code:**



* **Output:**





***Conclusion:***

It’s very interesting to see what you are creating by writing code. It’s like art based on code. Skills were developed to compile, run and test a simple program in the Python programming language.

As a result of the elaboration of the given paper, the basis was applied for the practical application of the theoretical knowledge.

The structures/concepts/algorithms used in this problem, after writing, compiling the program ,, several times.

Thus one can judge about the wide possibilities offered by the Python language regarding data manipulation. In this practical work I realized the knowledge accumulated during the theoretical and practical classes, I consolidated the material and in some places I learned new things. It allowed the assessment of knowledge in writing style both for the grade and personally. It allowed us to correct mistakes and possible future misunderstandings. Under the guidance of the teacher, we conducted the first individual study on this subject, this facilitating the adaptation to the knowledge of the use of theoretical material.

The verification of the results confirms that the elaborated program works correctly.

Linear algorithms can be used to calculate mathematical expressions.

Where drawn conclusions about Python programming language. As in the end I can say that the study had a positive impact on my personal education.

## **Bibliography:**

1. <https://www.geeksforgeeks.org/python-classes-and-objects/>
2. <https://developers.google.com/edu/python>
3. <https://towardsdatascience.com/a-comprehensive-guide-for-classes-in-python-e6bb72a25a5e>
4. <https://www.learnpythonwithrune.org/birthday-paradox-and-hash-function-collisions-by-example/>
5. <https://auth0.com/blog/image-processing-in-python-with-pillow/>
6. <https://www.geeksforgeeks.org/birthday-attack-in-cryptography/>
7. <https://stackoverflow.com/questions/52484901/using-monte-carlo-method-in-python>
8. <https://www.codegrepper.com/code-examples/python/frameworks/file-path-in-python/pillow+png+image>
9. <https://www.youtube.com/watch?v=PMISi2ZyuR8>
10. <https://towardsdatascience.com/monte-carlo-simulations-with-python-part-1-f5627b7d60b0>
11. <https://support.microsoft.com/en-us/office/converting-documents-to-a-newer-format-34ec742e-f0f9-4d95-bbe3-3ee8e30a86fa?ns=winword&version=21&syslcid=1033&uilcid=1033&appver=zwd210&helpid=185764&ui=en-us&rs=en-us&ad=us>