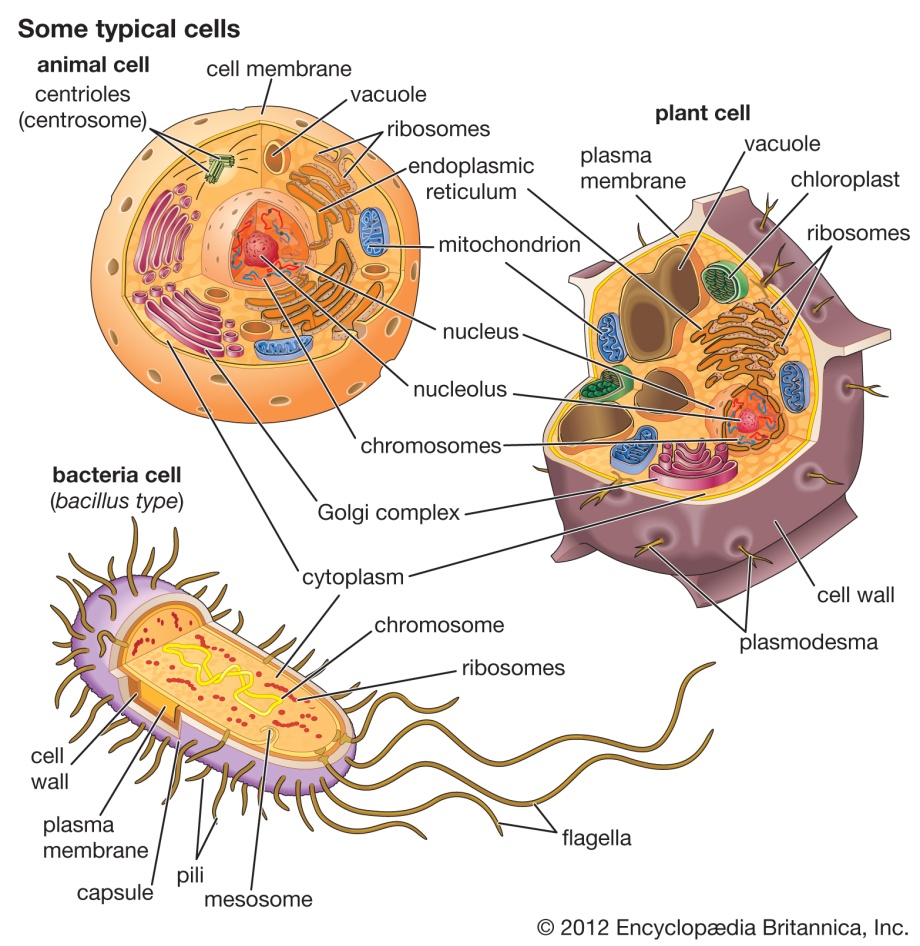
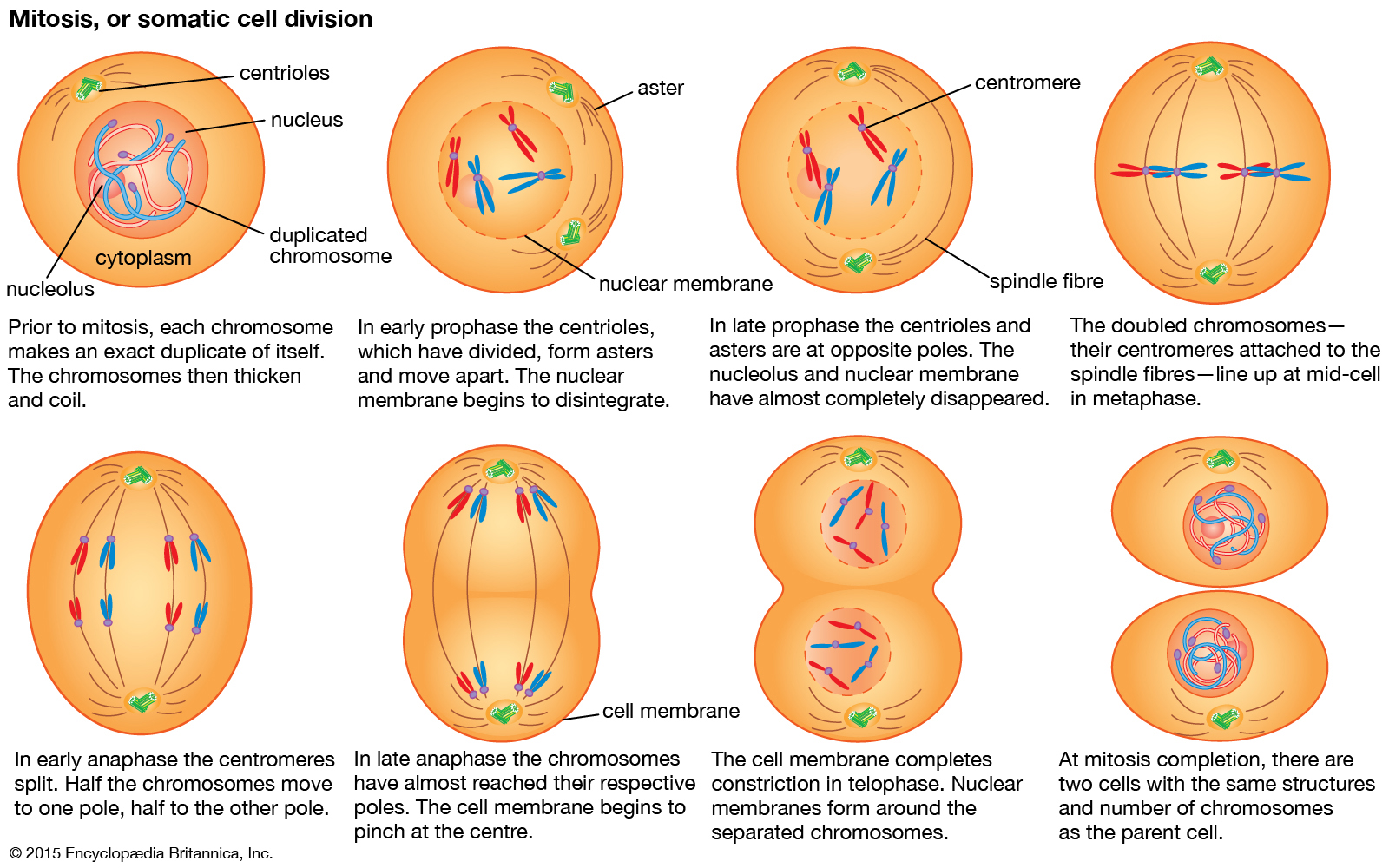
Приложение к проекту по теме: «Моделирование жизни одноклеточных организмов»

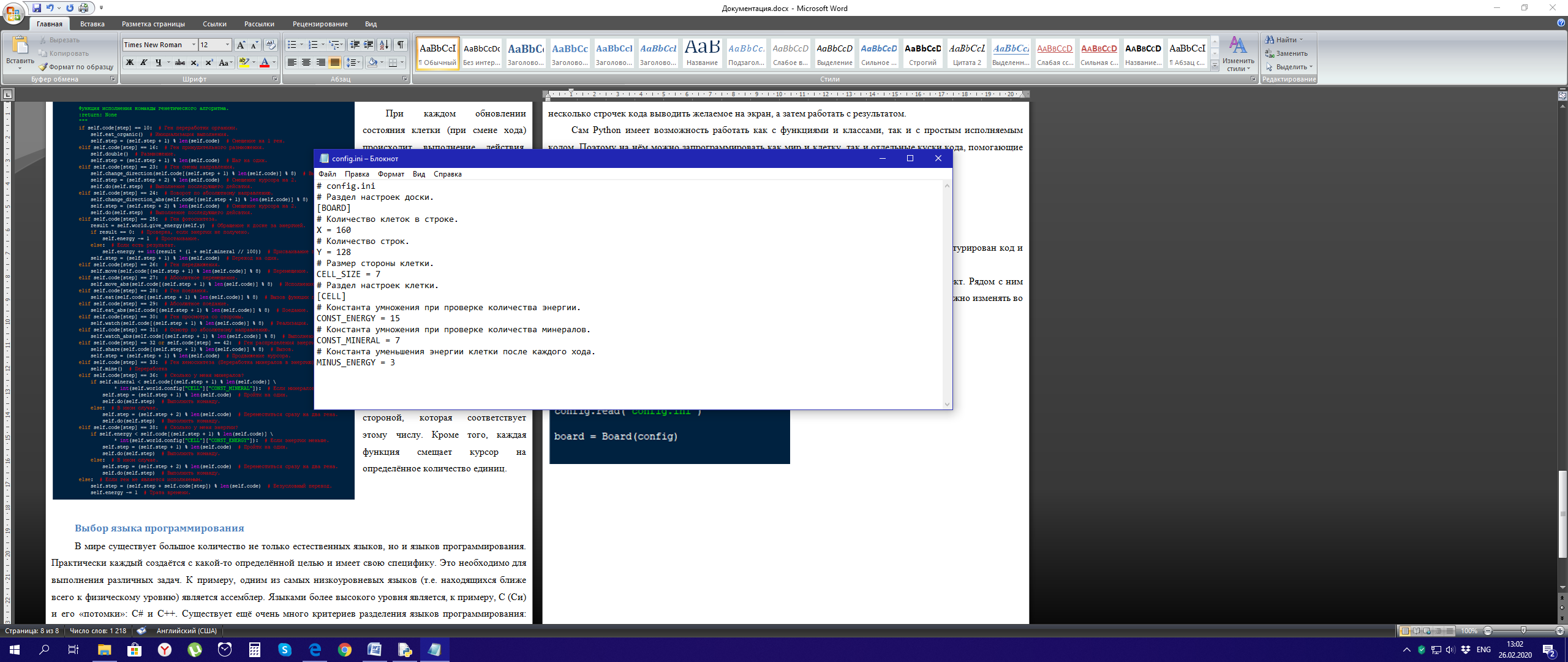


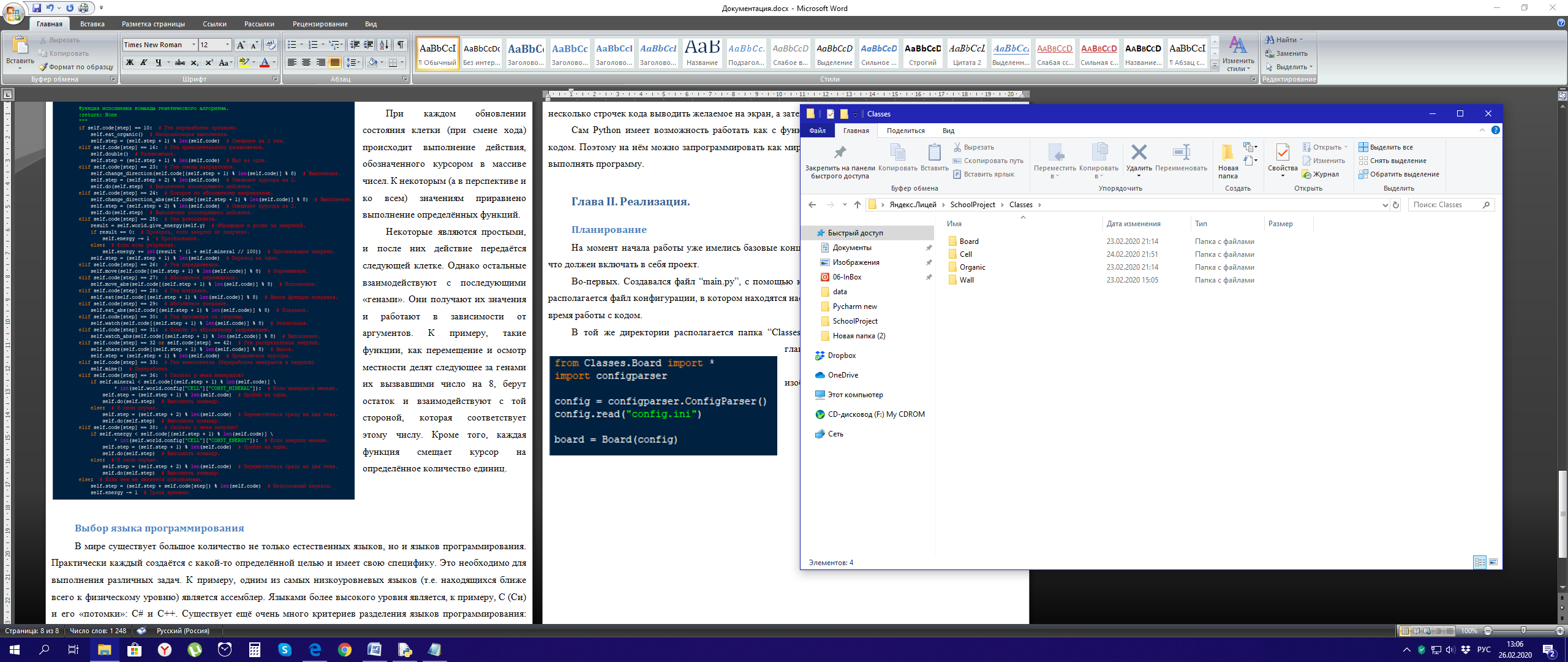
Разные виды клеток: животная, растительная и бактериальная.

# 

Функция, вызывающаяся при создании экземпляра класса бота.

Процесс деления клетки. Процесс схож с тем, что происходит и у бактерий.

Код функции выполнения действий.

Файл config.ini с переменными и настройками для программы.

Папка с классами приложения: доска, клетка, органика и стена.

Файл main.py, содержащий считывание конфигурационного файла и запуск проекта.

# Деление клетки:

def double(self) -> None:

"""

Отпочкование потомка.

:return: None

"""

directions = self.directions.copy() # Копирование списка направлений для работы с ним.

if self.energy < 152: # Если недостаточно энергии для деления и минимального разделения энергии.

self.board[self.y][self.x] = Organic(self.x, self.y, self.board) # Смерть.

else: # Если всего достаточно.

self.energy -= 150 # Штраф за размножение.

for \_ in range(8): # Перебор всех возможных направлений.

direction = choice([i for i in directions.keys()]) # Выбор случайного направления.

result = self.check\_direction(direction)

if result is None and result is not False: # Если в этом направлении пусто.

if directions[direction] == "UP": # Направление вверх.

self.board[self.y - 1][self.x] = Cell(self.world, self.x,

self.y - 1) # Создание клетки.

self.board[self.y - 1][self.x].energy = self.energy // 2 # Отдача энергии.

self.board[self.y - 1][self.x].mineral = self.mineral // 2 # Минералов.

self.board[self.y - 1][self.x].code = self.code.copy() # Передача генов.

self.energy //= 2 # Остаток в половину энергии.

self.mineral //= 2 # Остаток в половину минералов.

if randint(1, 4) == 4: # Шанс мутации равняется 25%.

self.board[self.y - 1][self.x].mutate() # Мутация потомка.

break # Окончание размножения.

if directions[direction] == "UP-RIGHT": # Направление ввеох и вправо.

self.board[self.y - 1][self.x + 1] = Cell(self.world, self.x + 1,

self.y - 1) # Создание клетки.

self.board[self.y - 1][

self.x + 1].energy = self.energy // 2 # Отдача трети энергии.

self.board[self.y - 1][self.x + 1].code = self.code.copy() # Передача генов.

self.board[self.y - 1][self.x + 1].mineral = self.mineral // 2 # Минералов.

self.mineral //= 2 # Остаток в половину минералов.

self.energy //= 2 # Остаток в половину энергии.

if randint(1, 4) == 4: # Шанс мутации равняется 25%.

self.board[self.y - 1][self.x + 1].mutate() # Мутация потомка.

break # Окончание размножения.

if directions[direction] == "RIGHT": # Направление вправо.

self.board[self.y][self.x + 1] = Cell(self.world, self.x + 1,

self.y) # Создание клетки.

self.board[self.y][

self.x + 1].energy = self.energy // 2 # Отдача трети энергии.

self.board[self.y][self.x + 1].mineral = self.mineral // 2 # Минералов.

self.mineral //= 2 # Остаток в половину минералов.

self.board[self.y][self.x + 1].code = self.code.copy() # Передача генов.

self.energy //= 2 # Остаток в половину энергии.

if randint(1, 4) == 4: # Шанс мутации равняется 25%.

self.board[self.y][self.x + 1].mutate() # Мутация потомка.

break # Окончание размножения.

if directions[direction] == "RIGHT-DOWN": # Направление вниз и вправо.

self.board[self.y + 1][self.x + 1] = Cell(self.world, self.x + 1,

self.y + 1) # Создание клетки.

self.board[self.y + 1][

self.x + 1].energy = self.energy // 2 # Отдача трети энергии.

self.board[self.y + 1][self.x + 1].mineral = self.mineral // 2 # Минералов.

self.mineral //= 2 # Остаток в половину минералов.

self.board[self.y + 1][self.x + 1].code = self.code.copy() # Передача генов.

self.energy //= 2 # Остаток в половину энергии.

if randint(1, 4) == 4: # Шанс мутации равняется 25%.

self.board[self.y + 1][self.x + 1].mutate() # Мутация потомка.

break # Окончание размножения.

if directions[direction] == "DOWN": # Направление вниз.

self.board[self.y + 1][self.x] = Cell(self.world, self.x,

self.y + 1) # Создание клетки.

self.board[self.y + 1][self.x].energy = self.energy // 2 # Отдача энергии.

self.board[self.y + 1][self.x].mineral = self.mineral // 2 # Минералов.

self.mineral //= 2 # Остаток в половину минералов.

self.board[self.y + 1][self.x].code = self.code.copy() # Передача генов.

self.energy //= 2 # Остаток в половину энергии.

if randint(1, 4) == 4: # Шанс мутации равняется 25%.

self.board[self.y + 1][self.x].mutate() # Мутация потомка.

break # Окончание размножения.

if directions[direction] == "DOWN-LEFT": # Направление вниз и влево.

self.board[self.y + 1][self.x - 1] = Cell(self.world, self.x - 1,

self.y + 1) # Создание клетки.

self.board[self.y + 1][

self.x - 1].energy = self.energy // 2 # Отдача трети энергии.

self.board[self.y + 1][self.x - 1].mineral = self.mineral // 2 # Минералов.

self.mineral //= 2 # Остаток в половину минералов.

self.board[self.y + 1][self.x - 1].code = self.code.copy() # Передача генов.

self.energy //= 2 # Остаток в половину энергии.

if randint(1, 4) == 4: # Шанс мутации равняется 25%.

self.board[self.y + 1][self.x - 1].mutate() # Мутация потомка.

break # Окончание размножения.

if directions[direction] == "LEFT": # Направление влево.

self.board[self.y][self.x - 1] = Cell(self.world, self.x - 1,

self.y) # Создание клетки.

self.board[self.y][self.x - 1].energy = self.energy // 2 # Отдача энергии.

self.board[self.y][self.x - 1].mineral = self.mineral // 2 # Минералов.

self.mineral //= 2 # Остаток в половину минералов.

self.board[self.y][self.x - 1].code = self.code.copy() # Передача генов.

self.energy //= 2 # Остаток в половину энергии.

if randint(1, 4) == 4: # Шанс мутации равняется 25%.

self.board[self.y][self.x - 1].mutate() # Мутация потомка.

break # Окончание размножения.

if directions[direction] == "LEFT-UP": # Направление вверх и влево.

self.board[self.y - 1][self.x - 1] = Cell(self.world, self.x - 1,

self.y - 1) # Создание клетки.

self.board[self.y - 1][

self.x - 1].energy = self.energy // 2 # Отдача трети энергии.

self.board[self.y - 1][self.x - 1].code = self.code.copy() # Передача генов.

self.board[self.y - 1][self.x - 1].mineral = self.mineral // 2 # Минералов.

self.mineral //= 2 # Остаток в половину минералов.

self.energy //= 2 # Остаток в половину энергии.

if randint(1, 4) == 4: # Шанс мутации равняется 25%.

self.board[self.y - 1][self.x - 1].mutate() # Мутация потомка.

break # Окончание размножения.

else: # Если не пусто.

del directions[direction] # Удаление проверенного направления.

if len(directions) == 0: # Окончание всех проверок.

self.board[self.y][self.x] = Organic(self.x, self.y, self.board) # Смерть.

# Органика:

class Organic:

"""

Класс органики.

"""

def \_\_init\_\_(self, x: int, y: int, world):

"""

Инициализация органики.

:param x: положение органики по ширине.

:param y: положение органики оп высоте.

:param world: родительский мир.

"""

self.x = x # Положение органики по ширине.

self.y = y # Положение органики по высоте.

self.world = world # Родительский мир.

self.check = True # Переключатель для проверки совршения хода.

def update(self) -> None:

"""

Обновление положения. Летит вниз.

:return: None

"""

if self.y < len(self.world) - 1 and self.world[self.y + 1][self.x] is None: # Если пусто.

self.y += 1 # Изменение координаты своего положение на 1 вниз.

self.world[self.y][self.x] = self # Копирование себя.

self.world[self.y - 1][self.x] = None # Уничтожение своей прошлой версии.

# Класс стены:

class Wall:

pass