Листинг 1.1. Импорт модулей и присваивание констант, используемых в программе bayes

import sys   
import random   
import itertools   
import numpy as np   
import cv2 as cv   
  
MAP\_FILE = 'cape\_python.png'  
SA1\_CORNERS = (130, 265, 180, 315) # (UL-X, UL-Y, LR-X, LR-Y)   
SA2\_CORNERS = (80, 255, 130, 305) # (UL-X, UL-Y, LR-X, LR-Y)   
SA3\_CORNERS = (105, 205, 155, 255) # (UL-X, UL-Y, LR-X, LR-Y)

Листинг 1.2. Определение класса Search и метода init ()

class Search():   
 """Байесовская игра "Поиск и спасение" с 3 областями поиска."""

def \_\_init\_\_(self, name):   
 self.name = name  
 self.img = cv.imread(MAP\_FILE, cv.IMREAD\_COLOR)   
 if self.img is None:   
 print('Could not load map file {}'.format(MAP\_FILE),   
 file=sys.stderr)   
 sys.exit(1)   
 self.area\_actual = 0  
 self.sailor\_actual = [0, 0] # "локальные" координаты в области поиска

self.sa1 = self.img[SA1\_CORNERS[1] : SA1\_CORNERS[3],   
 SA1\_CORNERS[0] : SA1\_CORNERS[2]]  
 self.sa2 = self.img[SA2\_CORNERS[1] : SA2\_CORNERS[3],   
 SA2\_CORNERS[0] : SA2\_CORNERS[2]]   
 self.sa3 = self.img[SA3\_CORNERS[1] : SA3\_CORNERS[3],   
 SA3\_CORNERS[0] : SA3\_CORNERS[2]]   
 self.p1 = 0.2   
 self.p2 = 0.5   
 self.p3 = 0.3  
 self.sep1 = 0   
 self.sep2 = 0  
 self.sep3 = 0

Листинг 1.3. Определение метода для отображения базовой карты  
  
 def draw\_map(self, last\_known):  
 """Отображаем базовую карту с масштабом, последними известными координатами xy и областями поиска."""   
 cv.line(self.img, (20, 370), (70, 370), (0, 0, 0), 2)  
 cv.putText(self.img, '0', (8, 370), cv.FONT\_HERSHEY\_PLAIN, 1, (0, 0, 0))   
 cv.putText(self.img, '50 Nautical Miles', (71, 370),   
 cv.FONT\_HERSHEY\_PLAIN, 1, (0, 0, 0))   
  
 cv.rectangle(self.img, (SA1\_CORNERS[0], SA1\_CORNERS[1]),  
 (SA1\_CORNERS[2], SA1\_CORNERS[3]), (0, 0, 0), 1)   
 cv.putText(self.img, '1',  
 (SA1\_CORNERS[0] + 3, SA1\_CORNERS[1] + 15),  
 cv.FONT\_HERSHEY\_PLAIN, 1, 0)  
 cv.rectangle(self.img, (SA2\_CORNERS[0], SA2\_CORNERS[1]),  
 (SA2\_CORNERS[2], SA2\_CORNERS[3]), (0, 0, 0), 1)  
 cv.putText(self.img, '2',   
 (SA2\_CORNERS[0] + 3, SA2\_CORNERS[1] + 15),  
 cv.FONT\_HERSHEY\_PLAIN, 1, 0)  
 cv.rectangle(self.img, (SA3\_CORNERS[0], SA3\_CORNERS[1]),  
 (SA3\_CORNERS[2], SA3\_CORNERS[3]), (0, 0, 0), 1)  
 cv.putText(self.img, '3',   
 (SA3\_CORNERS[0] + 3, SA3\_CORNERS[1] + 15),  
 cv.FONT\_HERSHEY\_PLAIN, 1, 0)   
 cv.putText(self.img, '+', (last\_known),  
 cv.FONT\_HERSHEY\_PLAIN, 1, (0, 0, 255))  
 cv.putText(self.img, '+ = Last Known Position', (274, 355),   
 cv.FONT\_HERSHEY\_PLAIN, 1, (0, 0, 255))  
 cv.putText(self.img, '\* = Actual Position', (275, 370),  
 cv.FONT\_HERSHEY\_PLAIN, 1, (255, 0, 0))  
 cv.imshow('Search Area', self.img)  
 cv.moveWindow('Search Area', 750, 10)  
 cv.waitKey(500)

Листинг 1.4. Определение метода для случайного выбора фактического местоположения моряка  
def sailor\_final\_location(self, num\_search\_areas):   
 """Возвращаем координаты x,y потерявшегося моряка."""   
 # Поиск координат моряка в отношении любого подмассива области поиска  
 Search Area.  
self.sailor\_actual[0] = np.random.choice(self.sa1.shape[1])  
self.sailor\_actual[1] = np.random.choice(self.sa1.shape[0])

area = int(random.triangular(1, num\_search\_areas + 1))

if area == 1:  
 x = self.sailor\_actual[0] + SA1\_CORNERS[0]   
 y = self.sailor\_actual[1] + SA1\_CORNERS[1]   
 self.area\_actual = 1  
elif area == 2:   
 x = self.sailor\_actual[0] + SA2\_CORNERS[0]   
 y = self.sailor\_actual[1] + SA2\_CORNERS[1]  
 self.area\_actual = 2  
 elif area == 3:   
 x = self.sailor\_actual[0] + SA3\_CORNERS[0]  
 y = self.sailor\_actual[1] + SA3\_CORNERS[1]  
 self.area\_actual = 3  
 return x, y

Листинг 1.5. Определение методов для случайного выбора эффективности поиска и выполнения самого поиска

def calc\_search\_effectiveness(self):  
 """Устанавливаем десятичное значение эффективности поиска для каждой области поиска."""  
 self.sep1 = random.uniform(0.2, 0.9)  
 self.sep2 = random.uniform(0.2, 0.9)  
 self.sep3 = random.uniform(0.2, 0.9)  
def conduct\_search(self, area\_num, area\_array, effectiveness\_prob):  
 """Возвращаем результаты поиска и список просмотренных координат."""   
 local\_y\_range = range(area\_array.shape[0])  
 local\_x\_range = range(area\_array.shape[1])  
 coords = list(itertools.product(local\_x\_range, local\_y\_range))  
 random.shuffle(coords)  
 coords = coords[:int((len(coords) \* effectiveness\_prob))]  
 loc\_actual = (self.sailor\_actual[0], self.sailor\_actual[1])  
 if area\_num == self.area\_actual and loc\_actual in coords:  
 return 'Found in Area {}.'.format(area\_num), cords  
 else:  
 return 'Not Found', cords

Листинг 1.6. Определение способов применения теоремы Байеса и отрисовка меню в оболочке Python

def revise\_target\_probs(self):  
 """Обновляем вероятности целей в области на основе эффективности поиска."""  
 denom = self.p1 \* (1 - self.sep1) + self.p2 \* (1 - self.sep2) \  
 + self.p3 \* (1 - self.sep3)  
 self.p1 = self.p1 \* (1 - self.sep1) / denom  
 self.p2 = self.p2 \* (1 - self.sep2) / denom  
 self.p3 = self.p3 \* (1 - self.sep3) / denom

def draw\_menu(search\_num):  
 """Выводим меню выбора для проведения поиска в области."""  
 print('\nSearch {}'.format(search\_num))  
 print(  
 """  
 Choose next areas to search:

0 - Quit   
1 - Search Area 1 twice  
2 - Search Area 2 twice  
3 - Search Area 3 twice   
4 - Search Areas 1 & 2   
5 - Search Areas 1 & 3   
6 - Search Areas 2 & 3   
7 - Start Over  
"""  
)

Листинг 1.7. Определение начала функции main(), используемой для запуска программы

def main():  
 app = Search('Cape\_Python')  
 app.draw\_map(last\_known=(160, 290))  
 sailor\_x, sailor\_y = app.sailor\_final\_location(num\_search\_areas=3)  
 print("-" \* 65)  
 print("\nInitial Target (P) Probabilities:") print("P1 = {:.3f}, P2 = {:.3f}, P3 = {:.3f}".format(app.p1, app.p2, app.p3))  
 search\_num = 1

Листинг 1.8. Использование цикла для выбора пунктов меню и запуска игры

while True:  
 app.calc\_search\_effectiveness()  
 draw\_menu(search\_num)  
 choice = input("Choice: ")

if choice == "0":  
 sys.exit()  
 elif choice == "1":  
 results\_1, coords\_1 = app.conduct\_search(1, app.sa1, app.sep1)  
 results\_2, coords\_2 = app.conduct\_search(1, app.sa1, app.sep1)  
 app.sep1 = (len(set(coords\_1 + coords\_2))) / (len(app.sa1)\*\*2)  
 app.sep2 = 0  
 app.sep3 = 0  
  
 elif choice == "2":  
 results\_1, coords\_1 = app.conduct\_search(2, app.sa2, app.sep2)  
 results\_2, coords\_2 = app.conduct\_search(2, app.sa2, app.sep2)  
 app.sep1 = 0  
 app.sep2 = (len(set(coords\_1 + coords\_2))) / (len(app.sa2)\*\*2)  
 app.sep3 = 0

elif choice == "3":  
 results\_1, coords\_1 = app.conduct\_search(3, app.sa3, app.sep3)  
 results\_2, coords\_2 = app.conduct\_search(3, app.sa3, app.sep3)  
 app.sep1 = 0  
 app.sep2 = 0  
 app.sep3 = (len(set(coords\_1 + coords\_2))) / (len(app.sa3)\*\*2)

elif choice == "4":  
 results\_1, coords\_1 = app.conduct\_search(1, app.sa1, app.sep1)  
 results\_2, coords\_2 = app.conduct\_search(2, app.sa2, app.sep2)  
 app.sep3 = 0

elif choice == "5":  
 results\_1, coords\_1 = app.conduct\_search(1, app.sa1, app.sep1)  
 results\_2, coords\_2 = app.conduct\_search(3, app.sa3, app.sep3)  
 app.sep2 = 0

elif choice == "6":  
 results\_1, coords\_1 = app.conduct\_search(2, app.sa2, app.sep2)  
 results\_2, coords\_2 = app.conduct\_search(3, app.sa3, app.sep3)  
 app.sep1 = 0

elif choice == "7":  
 main()  
  
 else:  
 print("\nSorry, but that isn't a valid choice.", file=sys.stderr)  
 continue

Листинг 1.9. Завершение и вызов функции main()

app.revise\_target\_probs() # формула Байеса для обновления вероятностей нахождения

print("\nSearch {} Results 1 = {}"  
 .format(search\_num, results\_1), file=sys.stderr)  
 print("Search {} Results 2 = {}\n"  
 .format(search\_num, results\_2), file=sys.stderr)  
 print("Search {} Effectiveness (E):".format(search\_num))  
 print("E1 = {:.3f}, E2 = {:.3f}, E3 = {:.3f}"  
 .format(app.sep1, app.sep2, app.sep3))

if results\_1 == 'Not Found' and results\_2 == 'Not Found':  
 print("\nNew Target Probabilities (P) for Search {}:"  
 .format(search\_num + 1))  
 print("P1 = {:.3f}, P2 = {:.3f}, P3 = {:.3f}"  
 .format(app.p1, app.p2, app.p3))  
 else:  
 cv.circle(app.img, (sailor\_x, sailor\_y), 3, (255, 0, 0), -1)  
 cv.imshow('Search Area', app.img)  
 cv.waitKey(1500)  
 main()  
 search\_num += 1

if \_\_name\_\_ == '\_\_main\_\_':  
 main()