Question 1:

```
# ArrayMax
# RunTime: O(n)

import numpy as np
def ArrayMax(arr) -> int:
  curr_max = arr[0]

  for i in arr:
    if i > curr_max:
      curr_max = i

  return curr_max

arr = np.random.randint(0, high=1000, size=100)
print(f'Max in Arr: {ArrayMax(arr)}')
```

This program will run in O(n) time. This is because the algorithms loops through the contents of the array only once.

Question 2:

```
#PrefixAverages1
#Runtime: O(n^2)

import numpy as np
def PrefixAverages1(arr):
  a = []
  for i in range(len(arr)):
    s = arr[0]
    for j in range(1, i):
      s = s + arr[j]
    a.append(round(s / (i + 1)))
  return a


arr = np.random.randint(0, high=1000, size=100)
print(f'{PrefixAverages1(arr)}')
```

This program has a runtime of $O(n^2)$ due to the nested loop. While the second loop will not always run the length of n, when i reaches the final length of n, j will follow suit and run the whole range as well, making it n^2.

Question 3:

```
#PrefixAverages2
#Runtime: O(n)
import numpy as np

def PrefixAverages2(arr):
  n = len(arr)
  a, s = [0] * n, 0

  for i in range(n):
    s += arr[i]
    a[i] = round(s / (i + 1))
  return a

arr = np.random.randint(0, high=1000, size=100)
print(f'{PrefixAverages2(arr)}')
```

This algorithm will run at O(n) because it runs the length of the array once.

Questions 4 & 5: