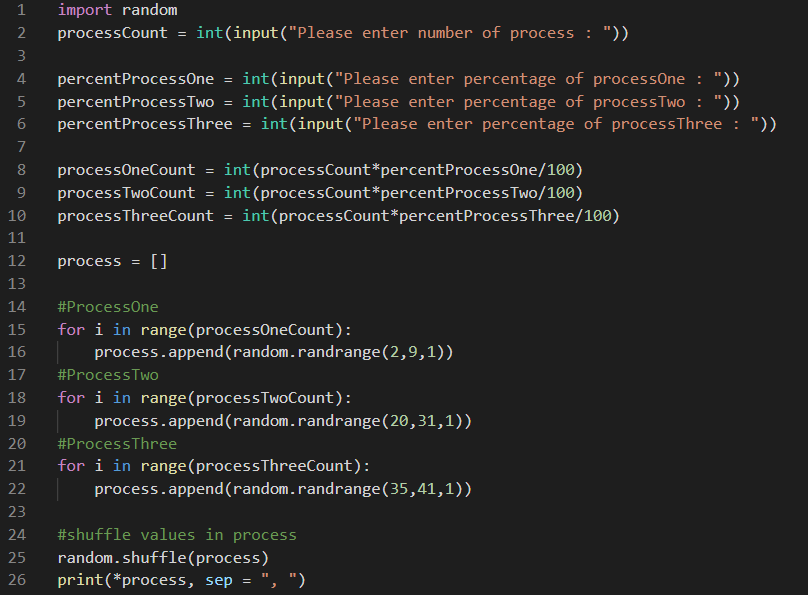
**Scheduling Algorithm**

นายอิทธิพร แก้วอำไพ 590610681



**Concept of Code**

**ส่วนของการรับค่า**

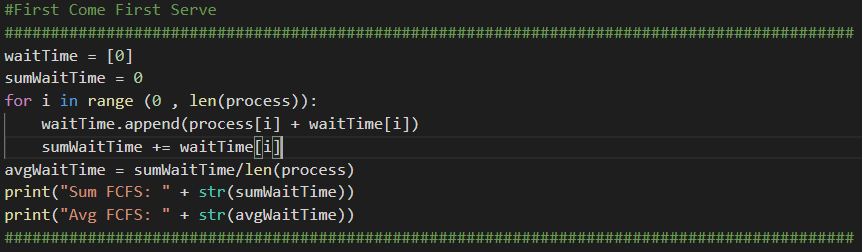
เริ่มต้นโปรแกรมด้วยรอรับค่าจำนวน process ที่ต้องการเป็นจำนวนที่ต้องการ หลังจากนั้นจะทำการแบ่งเป็น จะทำการแบ่งเป็น 3 ส่วน คือ process ที่ใช้เวลาน้อย process ที่ใช้เวลาปานกลางและ process ที่ใช้เวลานาน ซึ่งจะเปลี่ยนแปลงไปตามที่เราต้องการกำหนด โดย code ที่อยู่ในส่วนของ #ProcessOne คือ code ที่จะสุ่มค่าระหว่างเลข 2-8 ในส่วนของ #ProcessTwo คือ code ที่จะสุ่มค่าระหว่าง 20-30 และสุดท้าย #ProcessThree คือ code ที่จะสุ่มระหว่าง 35-40 หลังจากสุ่มจำนวนของข้อมูลทั้งหมดแล้วตัว code จะทำการสลับค่าทั้งหมดที่อยู่ใน process

**ส่วนของอัลกอริทึม**

1. **First Come First Serve (FCFS)**

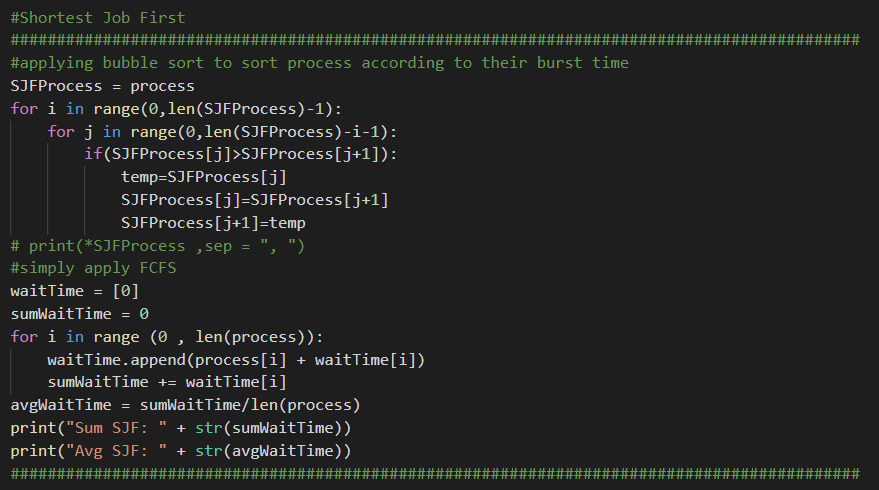
เป็นอัลกอริทึ่มที่ process จะทำงานตามลำดับ มาก่อนได้ก่อน สมมติว่ามีงานกำลังทำอยู่ แล้วมีอีกงานหนึ่งเข้ามางานนั้นจะต้องรอให้งานแรกเสร็จก่อน ดังนั้นถ้ามีงานต่อๆไปเข้ามา งานนั้นก็ต้องรอให้งานก่อนหน้าเสร็จก่อน จึงจะทำได้สามารถเขียนอธิบายอย่างง่ายได้ดังนี้

Process1 -> Process2 -> Process3 -> … -> ProcessN

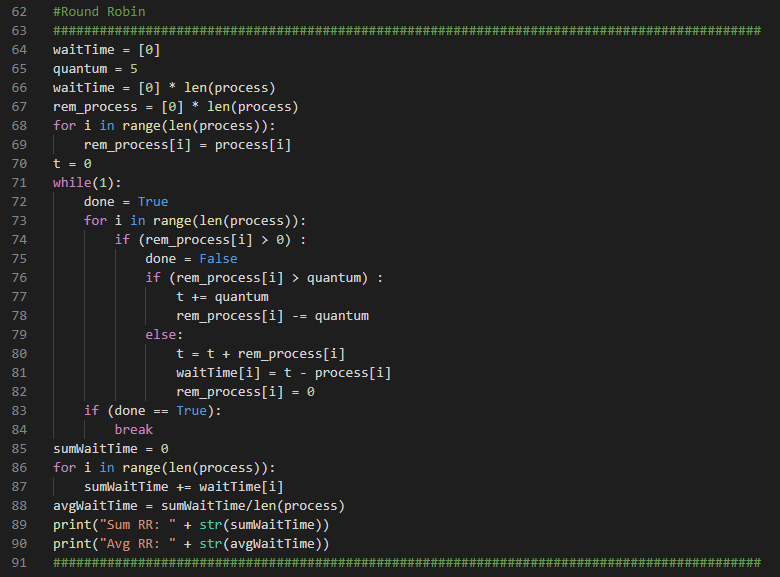


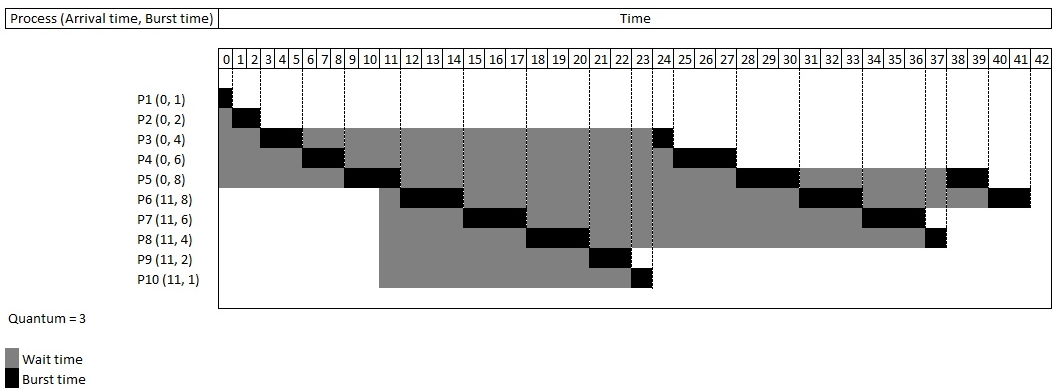
1. **Shortest Job First (SJF)**

เป็นอัลกอริทึ่มที่ process execution น้อยๆ จะได้เข้าคิวก่อน โดยในที่นี้ยังไม่ได้ใช้ arrive time ในการพิจารณา โดยหลักการทำงานก็คือเมื่อมี process เข้ามาทางอัลกอรึทึ่มจะทำการสลับ process ที่มีใช้เวลาน้อยมาทำก่อน



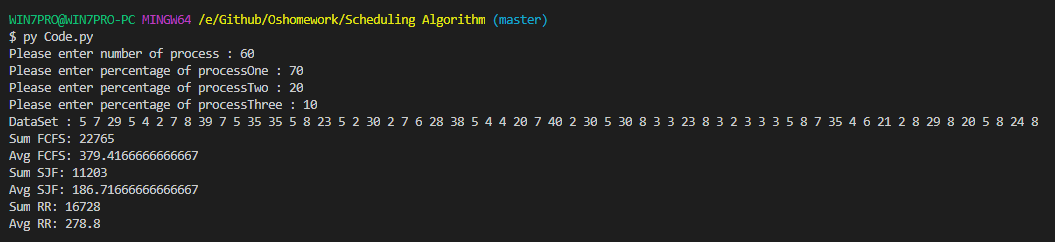
1. **Round Robin (RR)**

เป็นอัลกอริทึ่มที่ทุก process จะได้ทำในเวลาที่กำหนดให้ ซึ่งตัวที่กำหนดเวลาให้จะเรียกว่า Quantum time ไม่สน priority และมี Limit ของ Response Time ที่ชัดเจนจะรอแบบมีเวลาจำกัด ทำให้ไม่มี Starvation เกิดขึ้น ซึ่ง Quantum Time จะเป็นตัวกำหนด Wait Time ว่าจะมากหรือจะน้อย

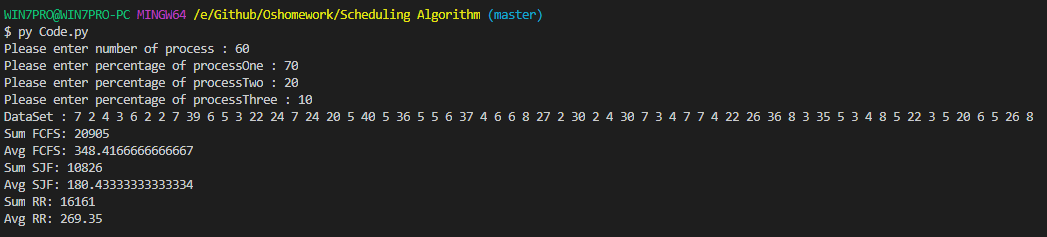


**ผลการทดลอง**

**สมมติฐานที่ 1** ถ้ามีชุดโปรเซส ที่ต้องการใช้ CPU ความยาวไม่เท่ากันในการทำงาน เข้ามาเพื่อให้จัดคิวจำนวน 60 โปรเซส โดยให้นักศึกษาสร้างชุดโปรเซสเองโดยทำการสุ่มโปรเซสที่ใช้เวลา (2 ถึง 8 milisec) จำนวน 70 %, โปรเซสที่ใช้เวลา (20 ถึง 30 milisec) จำนวน 20 %  , โปรเซสที่ใช้เวลา (35 ถึง 40 milisec) จำนวน 10 % และเมื่อได้โปรเซสครบแล้วให้สุ่มลำดับที่เข้ามาในคิว

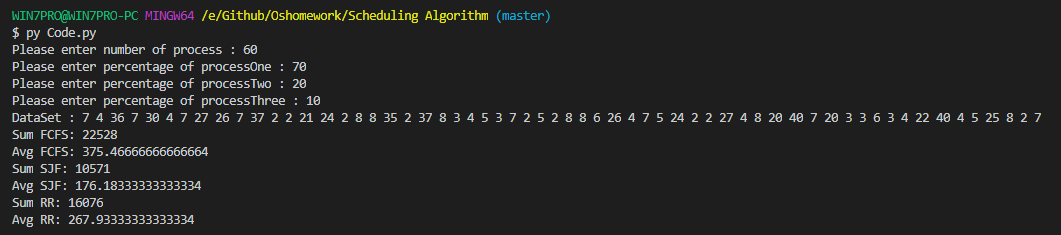
* 1. การรันครั้งที่ 1 ได้ผลดังภาพที่ 1.1

ภาพที่ 1.1

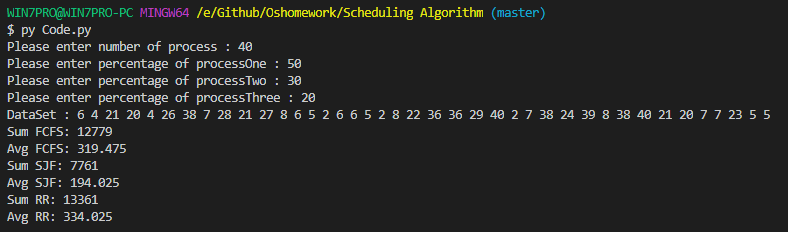
* 1. การรันครั้งที่ 2 ได้ผลดังภาพที่ 1.2

ภาพที่ 1.2

* 1. การรันครั้งที่ 3 ได้ผลดังภาพที่ 1.3

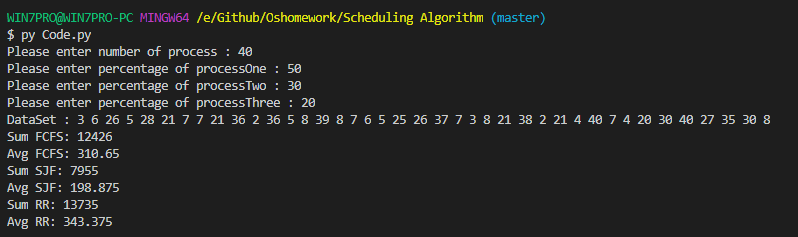


ภาพที่ 1.3

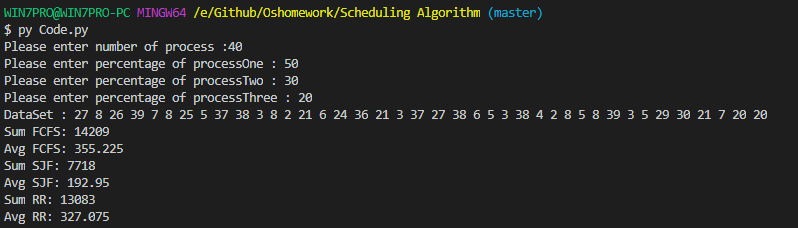
 **สมมติฐานที่ 2** ถ้ามีชุดโปรเซส ที่ต้องการใช้ CPU ความยาวไม่เท่ากันในการทำงาน เข้ามาเพื่อให้จัดคิวจำนวน 40 โปรเซส โดยให้นักศึกษาสร้างชุดโปรเซสเองโดยทำการสุ่มโปรเซสที่ใช้เวลา (2 ถึง 8 milisec) จำนวน 50 %, โปรเซสที่ใช้เวลา (20 ถึง 30 milisec) จำนวน 30 %  , โปรเซสที่ใช้เวลา (35 ถึง 40 milisec) จำนวน 20 % และเมื่อได้โปรเซสครบแล้วให้สุ่มลำดับที่เข้ามาในคิว

a. การรันครั้งที่ 1 ได้ผลดังภาพที่ 2.1

ภาพที่ 2.1

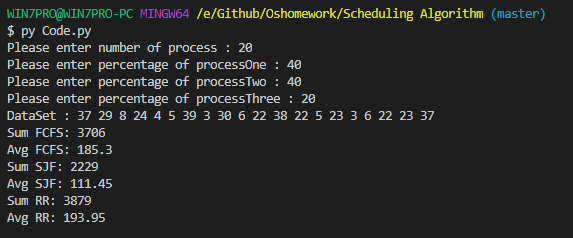
b. การรันครั้งที่ 2 ได้ผลดังภาพที่ 2.2

ภาพที่ 2.2

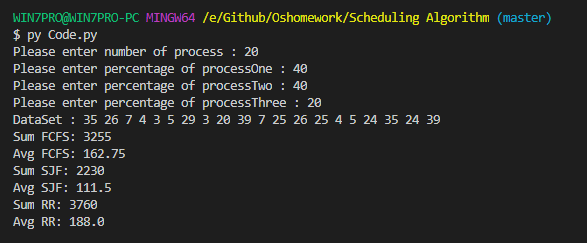
c. การรันครั้งที่ 3 ได้ผลดังภาพที่ 2.3

ภาพที่ 2.3

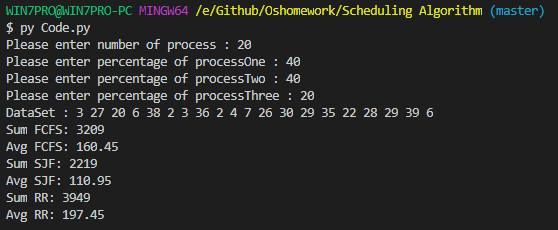
**สมมติฐานที่ 3** ถ้ามีชุดโปรเซส ที่ต้องการใช้ CPU ความยาวไม่เท่ากันในการทำงาน เข้ามาเพื่อให้จัดคิวจำนวน 20 โปรเซส โดยให้นักศึกษาสร้างชุดโปรเซสเองโดยทำการสุ่มโปรเซสที่ใช้เวลา (2 ถึง 8 milisec) จำนวน 40 %, โปรเซสที่ใช้เวลา (20 ถึง 30 milisec) จำนวน 40 %  , โปรเซสที่ใช้เวลา (35 ถึง 40 milisec) จำนวน 20 % และเมื่อได้โปรเซสครบแล้วให้สุ่มลำดับที่เข้ามาในคิว

a. การรันครั้งที่ 1 ได้ผลดังภาพที่ 3.1

ภาพที่ 3.1

b. การรันครั้งที่ 2 ได้ผลดังภาพที่ 3.2

ภาพที่ 3.2

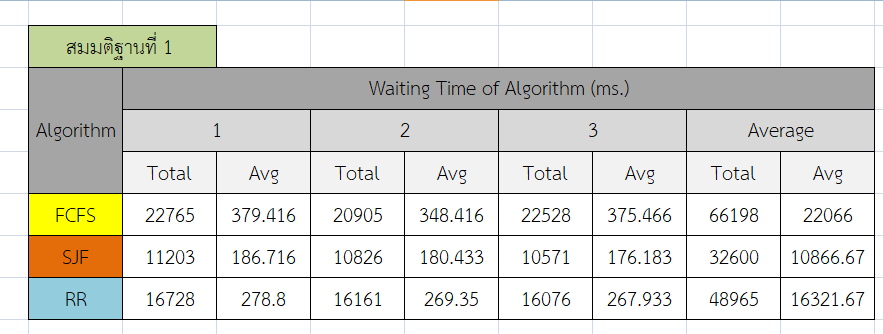
c. การรันครั้งที่ 3 ได้ผลดังภาพที่ 3.3

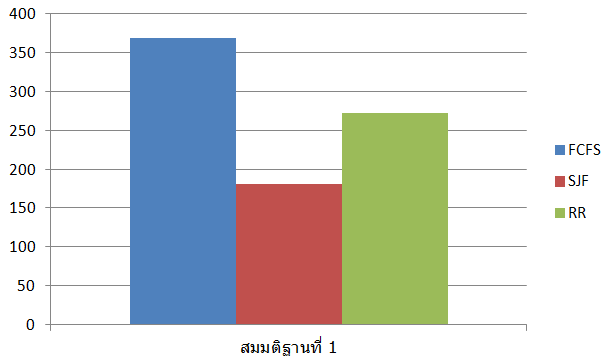
ภาพที่ 3.3

**ผลการทดลอง**

จากการทดลองสมมติฐาน 3 แบบ สามารถนำไปแสดงเป็นกราฟได้ดังนี้

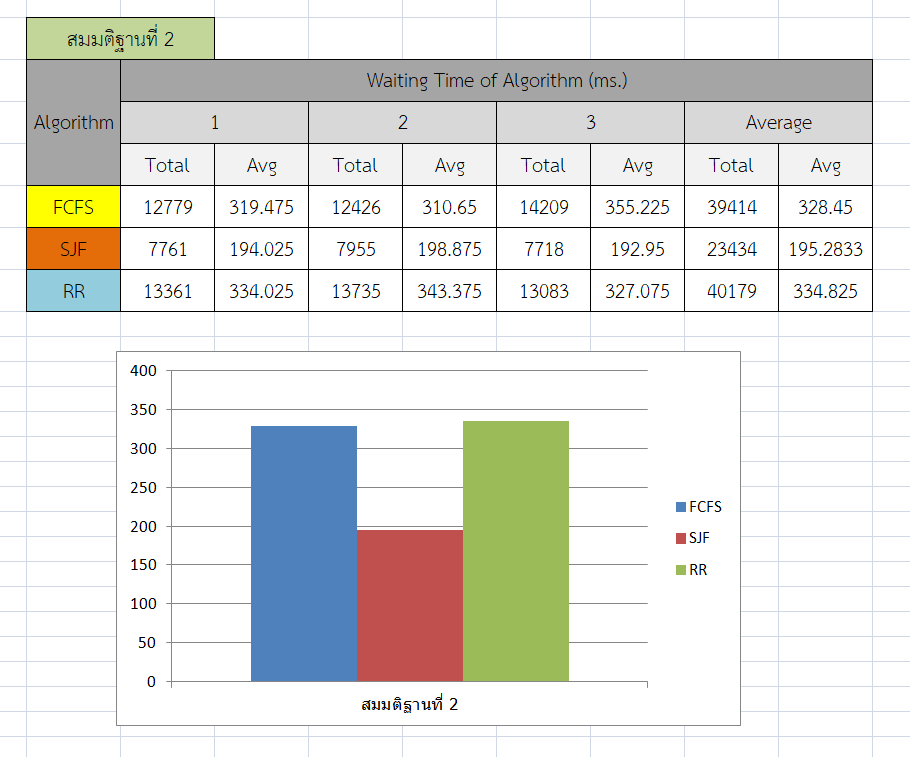
**สมมติฐานที่ 1**





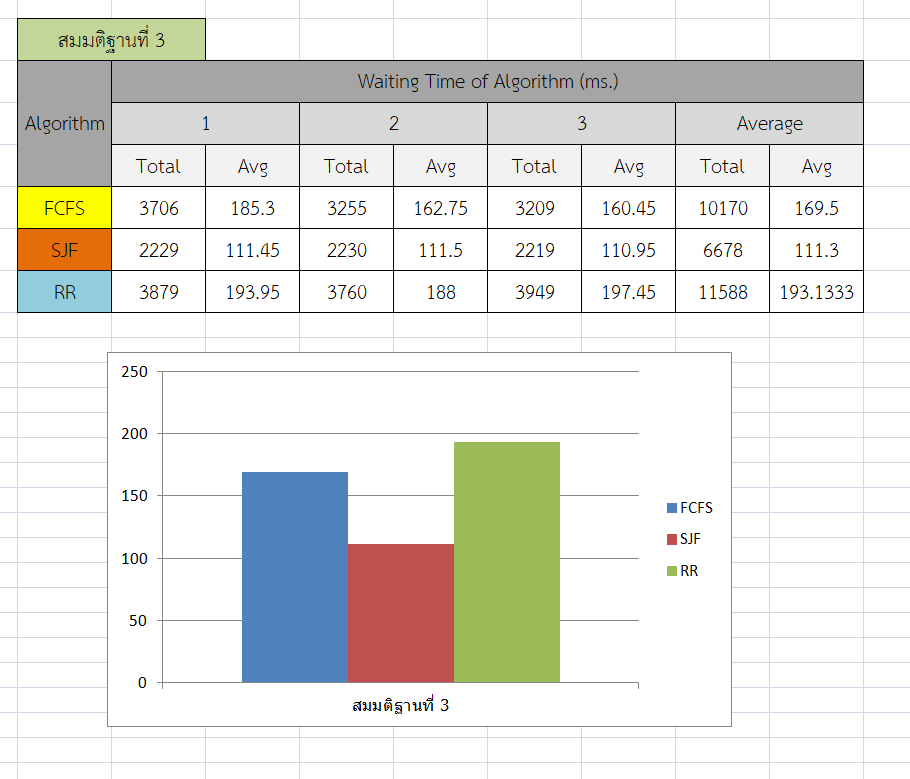
จากสมมติฐานที่ 1ถ้ามีชุดโปรเซส ที่ต้องการใช้ CPU ความยาวไม่เท่ากันในการทำงาน เข้ามาเพื่อให้จัดคิวจำนวน 60 โปรเซส โดยให้นักศึกษาสร้างชุดโปรเซสเองโดยทำการสุ่มโปรเซสที่ใช้เวลา (2 ถึง 8 milisec) จำนวน 70 %, โปรเซสที่ใช้เวลา (20 ถึง 30 milisec) จำนวน 20 %  , โปรเซสที่ใช้เวลา (35 ถึง 40 milisec) จำนวน 10 % และเมื่อได้โปรเซสครบแล้วให้สุ่มลำดับที่เข้ามาในคิว พบว่า **SJF ใช้เวลาน้อยที่สุด ตามมาด้วย RR และ FCFS ใช้เวลามากที่สุด**

**สมมติฐานที่ 2**



จากสมมติฐานที่ 2ถ้ามีชุดโปรเซส ที่ต้องการใช้ CPU ความยาวไม่เท่ากันในการทำงาน เข้ามาเพื่อให้จัดคิวจำนวน 40 โปรเซส โดยให้นักศึกษาสร้างชุดโปรเซสเองโดยทำการสุ่มโปรเซสที่ใช้เวลา (2 ถึง 8 milisec) จำนวน 50 %, โปรเซสที่ใช้เวลา (20 ถึง 30 milisec) จำนวน 30 %  , โปรเซสที่ใช้เวลา (35 ถึง 40 milisec) จำนวน 20 % และเมื่อได้โปรเซสครบแล้วให้สุ่มลำดับที่เข้ามาในคิวพบว่า **SJF ใช้เวลาน้อยที่สุด FCFS ใช้เวลารองลงมา และ RR ใช้เวลามากที่สุด**

**สมมติฐานที่ 3**



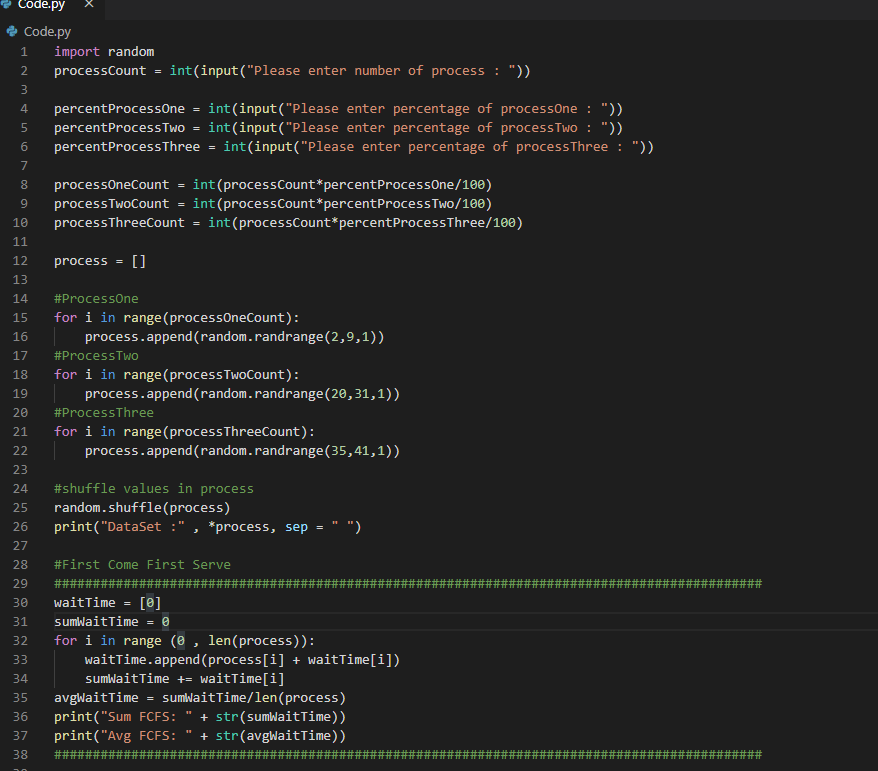
จากสมมติฐานที่ 3ถ้ามีชุดโปรเซส ที่ต้องการใช้ CPU ความยาวไม่เท่ากันในการทำงาน เข้ามาเพื่อให้จัดคิวจำนวน 20 โปรเซส โดยให้นักศึกษาสร้างชุดโปรเซสเองโดยทำการสุ่มโปรเซสที่ใช้เวลา (2 ถึง 8 milisec) จำนวน 40 %, โปรเซสที่ใช้เวลา (20 ถึง 30 milisec) จำนวน 40 %  , โปรเซสที่ใช้เวลา (35 ถึง 40 milisec) จำนวน 20 % และเมื่อได้โปรเซสครบแล้วให้สุ่มลำดับที่เข้ามาในคิว พบว่า **SJF ใช้เวลาน้อยที่สุด FCFS ใช้เวลารองลงมา และ RR ใช้เวลามากที่สุด**

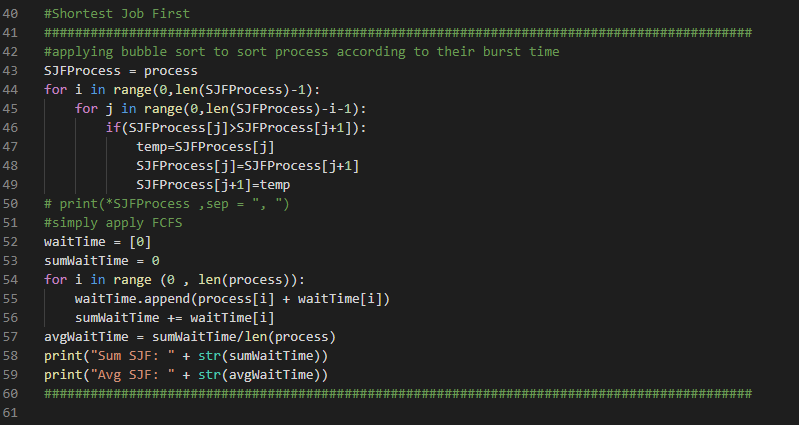
**สรุปผลการทดลอง**

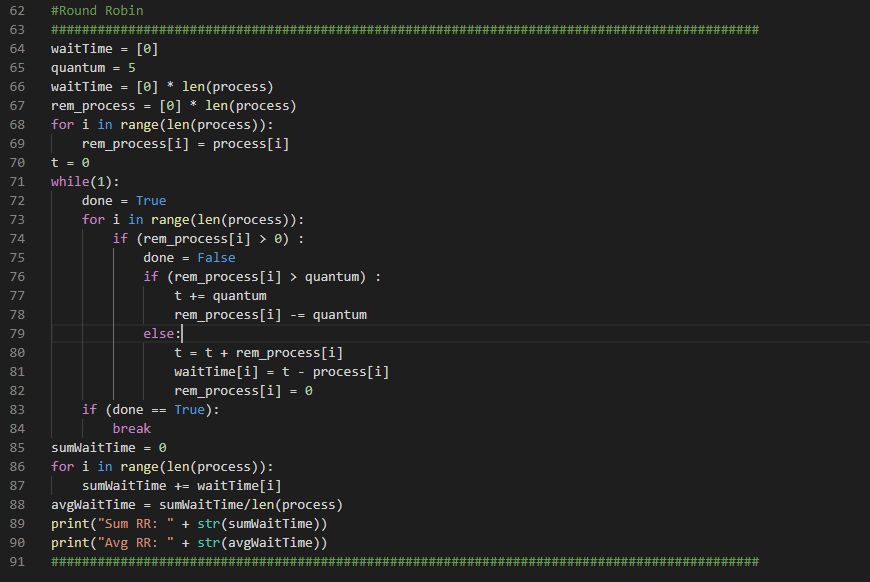
จากการทดลองทั้งหมดสรุปได้ว่า algorithm ของ Shortest Job First นั้นมีเวลาเฉลี่ยน้อยที่สุด ในส่วนของ First Come First Serve เป็นวิธีที่ธรรมดามากที่สุดแต่จะทำให้บางครั้ง Process สั้นๆต้องรอเป็นเวลานาน และในส่วนของ Round Robin นั้นการกำหนด Quantime จะเป็นตัวกำหนดเวลาที่ต้องรอ ซึ่งควรปรับตามความเหมาะสม

จากการสังเกตของผู้ทดลองพบว่า Shortest Job First เป็นการพัฒนามากจาก First Come First Serve ทำให้เกิดเวลารอที่น้อยที่สุด และยังพบว่า Round Robin ถูกพัฒนามาเพื่อลดการเกิดปัญหา starvation ซึ่งจะพบบ่อยใน FCFS และ SJF

**ภาคผนวก**

****

****

****