"""SST simulator case I.

```
Basic simulator of an organization of Stand By, stabilization
and transport of a cryonic case.
Where the system (SST) is composed of:
+ a subject (patient)
+ three agents (doctor, thanatopractician, ambulance)
+ a variable set of time ranges.
The objective is to simulate a set of cases to extract the
average and statistics to decide how to improve a
cryonic suspension procedure.

NOTE: this is ideal case (Die by old age, euthanasia on public hospital,
perfect collaboration between center and SST, perfect timing between steps,
all costs paid, all documents and arrangements made It.
```

"""

```
1 !pip install simpy statistics colorama matplotlib
```

```
Collecting simpy
  Using cached https://files.pythonhosted.org/packages/20/f9/874b0bab834068
Collecting statistics
  Downloading https://files.pythonhosted.org/packages/bb/3a/ae99a15e6563655
Collecting colorama
  Downloading https://files.pythonhosted.org/packages/44/98/5b86278fbbf250d
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-
Requirement already satisfied: docutils>=0.3 in /usr/local/lib/python3.7/di
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/pytho
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dis
Requirement already satisfied: numpy>=1.11 in /usr/local/lib/python3.7/dist
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-pa
Building wheels for collected packages: statistics
  Building wheel for statistics (setup.py) ... done
  Created wheel for statistics: filename=statistics-1.0.3.5-cp37-none-any.w
  Stored in directory: /root/.cache/pip/wheels/75/55/90/73aa7662bfb4565b567
Successfully built statistics
Installing collected packages: simpy, statistics, colorama
Successfully installed colorama-0.4.4 simpy-4.0.1 statistics-1.0.3.5
```

```
1 # Imports
2 import simpy
3 import random
4 import statistics
5 from colorama import init
6 from termcolor import colored
7 import matplotlib.pyplot as plt
```

✓ 0 s completado a las 1:08

```
SST = Stand By, Stabilization, Transport
Embalmer = thanatopractitioner
Medico = dispensing death certificate and embalming
Stand By = metabolic support and hypothermia of the patient,
manual (voluntary) or mechanical (ECMO, LUCA)
Stabilisation = controlled perfusion of cryoprotective fluids,
following IC protocol and formulas for reasons of economy.

This class instantiates the environment with its constituent
agents to iterate their actions with varying times, simulate good and bad times,
collect them and make a statistic of the average time.

Allowing to make simulations ahead of any practice and
thus reduce incidences or hazards for the patient.

"""
```

```
class Sst(object):
    # Environment
    def __init__(self, env, agent_embalmer, agent_doctor, agent_ambulance, ag
        self.env = env
        self.embalmer = simpy.Resource(env, agent_embalmer)
        self.doctor = simpy.Resource(env, agent_doctor)
        self.ambulance = simpy.Resource(env, agent_ambulance)
        self.place = simpy.Resource(env, agent_place)

    def death_authorization(self, patient):
        # Remember: timeout is limited to 12 hours
        yield self.env.timeout(random.randint(1, 6))

    def embalming_authorization(self, patient):
        yield self.env.timeout(random.randint(1, 6))

    def stand_by(self, patient):
        # No more than 1h support
        yield self.env.timeout(random.randint(0, 1))

    def perfusion(self, patient):
        # Time is optimizing for only head perfusion.
        yield self.env.timeout(random.randint(0, 1))
```

```
    all possible states of every case.

"""
```

```
def deanimation(env, patient, sst):
    # patient arrives at the Sst
    arrival_time = env.now

    if random.choice([True, False]): # Hypotetical case of extreme delay and
        with sst.doctor.request() as request: # apply stand by or straigh fre
            yield request
            yield env.process(sst.stand_by(patient))

    with sst.embalmer.request() as request:
        yield request
        yield env.process(sst.death_authorization(patient))

    with sst.ambulance.request() as request:
        yield request
        yield env.process(sst.embalming_authorization(patient))

    with sst.place.request() as request:
        yield request
        yield env.process(sst.perfusion(patient))

    # patient heads into the Sst
    wait_times.append(env.now - arrival_time)
```

"""Run SST.

```
Function who runs the procedure of execution environment.
Take the constants and variables Environment, Embalmer,
Doctor, ambulance.

Launch one deanimation per patient.
And generate another case after ending the last.
For take enought data for statistics.
```

```
 Function with procedure to generate the average time and
 feed the result to show general average time of
 suspension.

"""
```

```
def get_average_wait_time(wait_times):
    average_wait = statistics.mean(wait_times)
    # Pretty print the results
    minutes, frac_minutes = divmod(average_wait, 1)
    seconds = frac_minutes * 60
    return round(minutes), round(seconds)
```

```
def graph_and_statistics(wait_times):
    time_limit=[50]
    average_wait = statistics.mean(wait_times)
    mode_wait = statistics.mode(wait_times)
    median_wait = statistics.median(wait_times)
    print("Print mode times: ",mode_wait)
    print("Print median times: ",median_wait)
    print("Print wait times list: ",wait_times)
    print("Print average wait: ", round(average_wait),'red line')
    plt.axhline(round(average_wait), color="red")
    plt.bar(wait_times,wait_times)
    plt.title("global wait times per case")
    plt.ylabel('Max time: 100 minutes')
    plt.xlabel('Sorted generated cases')
    plt.show()
```

"""Get user input.

"""Main.

Main function launch and run all simulation, process and show the results.

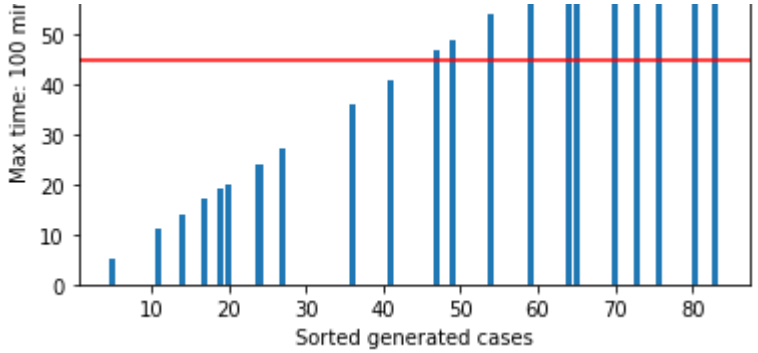
Next objectives:

+ Add param times ambulance.
+ Add individual times of doctor, embalmer, ambulance, wait time at funeral home, fly
+ Add readable and markdown color table he general and specific data.

"""

```
def main():
    # Setup
    random.seed(42)
    agent_embalmer, agent_doctor, agent_ambulance, agent_place = get_user_inp

    # Run the simulation
    env = simpy.Environment()
    env.process(run_Sst(env, agent_embalmer, agent_doctor, agent_ambulance, a
    env.run(until=90)
    # View the results
    mins, secs = get_average_wait_time(wait_times)
    print(colored(f"Based in stimated population in country: {population}","g
    print(
```



Conclusion:

The times to be managed within a hospital should be based on **prior preparation and planning between the support team, the embalmer and the physician on duty.** The patient should have** all his documents prepared **and once he has passed the **two medical opinion allowing him to euthanize**, (1 minute) the optimal average time should not exceed one minute until he is on the standby equipment, premedicated and receiving hypothermia.

Lower his body to the ambulance (maximum time 1 hour) to the funeral home where he will