1) From problem statement

$$P(y_2, y_1, y_0 \mid \alpha) = P(y_2 \mid y_1) \cdot P(y_1, y_0) \cdot P(y_0 \mid \alpha)$$

$$f(x) = \left(\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{y_2 - \alpha y_1}{\sigma}\right)^2} \right) \left(\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{y_1 - \alpha y_0}{\sigma}\right)^2} \right) \left(\frac{1}{\sqrt{2\pi\lambda}} e^{-\frac{1}{2}\left(\frac{y_0 - 0}{\sqrt{\lambda}}\right)^2} \right)$$

$$= \frac{1}{\sigma^2(2\pi)} \cdot \frac{1}{\sqrt{2\pi\lambda}} e^{-\frac{1}{2}\left[\left(\frac{y_2 - \alpha y_1}{\sigma}\right)^2 + \left(\frac{y_1 - \alpha y_0}{\sigma}\right)^2 + \left(\frac{y_0}{\sqrt{\lambda}}\right)^2\right]}$$

$$\ln(f(x)) = \ln\left(\frac{1}{\sigma^2(2\pi)} \cdot \frac{1}{\sqrt{2\pi\lambda}} e^{-\frac{1}{2}\left[\left(\frac{y_2 - \alpha y_1}{\sigma}\right)^2 + \left(\frac{y_1 - \alpha y_0}{\sigma}\right)^2 + \left(\frac{y_0}{\sqrt{\lambda}}\right)^2\right]}\right)$$

$$= \ln\left(\frac{1}{\sigma^2(2\pi)}\right) + \ln\left(\frac{1}{\sqrt{2\pi\lambda}}\right) + \ln\left(e^{-\frac{1}{2}\left[\left(\frac{y_2 - \alpha y_1}{\sigma}\right)^2 + \left(\frac{y_1 - \alpha y_0}{\sigma}\right)^2 + \left(\frac{y_0}{\sqrt{\lambda}}\right)^2\right]}\right)$$

$$= \ln\left(\frac{1}{\sigma^2(2\pi)}\right) + \ln\left(\frac{1}{\sqrt{2\pi\lambda}}\right) - \frac{1}{2}\left[\left(\frac{y_2 - \alpha y_1}{\sigma}\right)^2 + \left(\frac{y_1 - \alpha y_0}{\sigma}\right)^2 + \left(\frac{y_0}{\sqrt{\lambda}}\right)^2\right]$$

To find maximum value, we can find derivative of α

$$\frac{d}{d\alpha}\left(\ln\left(\frac{1}{\sigma^2(2\pi)}\right) + \ln\left(\frac{1}{\sqrt{2\pi\lambda}}\right) - \frac{1}{2}\left[\left(\frac{y_2 - \alpha y_1}{\sigma}\right)^2 + \left(\frac{y_1 - \alpha y_0}{\sigma}\right)^2 + \left(\frac{y_0}{\sqrt{\lambda}}\right)^2\right] \right) = 0$$

$$-\frac{1}{2}(2)\left(\frac{y_2 - \alpha y_1}{\sigma}\right)(-y_1) - \frac{1}{2}(2)\left(\frac{y_1 - \alpha y_0}{\sigma}\right)(-y_0) = 0$$

$$(y_2 - \alpha y_1)(y_1) + (y_1 - \alpha y_0)(y_0) = 0$$

$$y_2 y_1 - \alpha y_1^2 + y_1 y_0 - \alpha y_0^2 = 0$$

$$\alpha = \frac{y_2 y_1 + y_1 y_0}{y_1^2 + y_0^2}$$

4.1) No. Because the difference between 2 scenarios is blocking and new channel. The blocking might be the cause.

4.2) No. Since 0.1 is greater than significant level at 0.05. That means we cannot reject H_0 and we cannot conclude anything.

4.3) No. We cannot conclude that adding new channel has significance to number of visitors.

Yes. We can use hypothesis testing by adjusting the significant level of 4.2 to be 0.1.

5.1) H_0: The die is fair, and the probability of rolling the selected number is $\frac{1}{6}$

H_a: The die is biased, and the probability of rolling the selected number is less than $\frac{1}{6}$

5.2) One-sided because the player is interested in whether the probability of rolling the selected number is less than $\frac{1}{6}$ or not.

5.3) From the code below, the P value is not less than 0.1. So, we can't reject H_0

```python
import numpy as np
from scipy.stats import binom

n = 30
p = 1/6

prob = binom.cdf(3, n, p)
print("P Value:", prob)
```

✓ 0.0s

```
P Value: 0.23961952685801333
```

5.4) The reject region is ≤ 26 times

```python
import numpy as np
from scipy.stats import binom

n = 200
p = 1/6
region_bound = binom.ppf(0.1, n, p) - 1
print("Region Bound:", region_bound)
```

✓ 0.0s

```
Region Bound: 26.0
```

5.5) the reject region is ≤ 26.579

```python
import numpy as np
from scipy.stats import norm

e_x = 200*1/6
var_x = 200*1/6*(1-1/6)
x = norm.ppf(0.1, loc=e_x, scale=np.sqrt(var_x))
print("x:", x)
```

✓ 0.0s

```
x: 26.5789635232075
```

5.6) The lowest probability is 0.148

```python
import numpy as np
from scipy.stats import binom

p = 1/6

low = 0

for n in range(1, 201):
    region_bound = binom.ppf(0.01, n, p) - 1
    # print("Region Bound:", region_bound)

    precision = 0.0000000000000001
    l = 0
    r = p
    while((r-l) > precision):
        mid = (l+r)/2
        prob = binom.cdf(region_bound, n, mid)
        if prob > 0.05:
            l = mid+precision
        else:
            r = mid
        low = max(low, l)

print("lowest prob:", round(low, 3))
```

✓ 0.7s

```
lowest prob: 0.148
```

5.7) The lowest probability is 0.16664

```python
import numpy as np
from scipy.stats import binom

p = 1/6

low = 0

for n in range(1, 201):
    region_bound = binom.ppf(0.01, n, p) - 1
    # print("Region Bound:", region_bound)

    precision = 0.000000000000001
    l = 0
    r = p
    while((r-l) > precision):
        mid = (l+r)/2
        prob = binom.cdf(region_bound, n, mid)
        if prob > 0.01:
            l = mid+precision
        else:
            r = mid
        low = max(low, l)

print("lowest prob:", round(low, 5))
```
✓ 0.7s

```
lowest prob: 0.16664
```

7.1) $H_0 : \mu_{new} \leq \mu_{old}$

$H_a : \mu_{new} > \mu_{old}$

```python
from scipy.stats import norm

def getMean(data):
    return np.mean(data)

def getPValue(data):
    return norm.sf(getMean(data), loc=5000, scale=20/np.sqrt(len(data)))

allProduct = np.concatenate((fac[0], fac[1], fac[2], fac[3]))

print("P Value of whole factory:", getPValue(allProduct), "\tReject H0: ", getPValue(allProduct) < 0.05)
for i in range(4):
    print("P Value of factory", i, ":", getPValue(fac[i]), "\t\tReject H0: ", getPValue(fac[i]) < 0.05)
```
✓ 0.0s

```
P Value of whole factory: 1.135401468573459e-05         Reject H0:  True
P Value of factory 0 : 0.015206852813733384             Reject H0:  True
P Value of factory 1 : 0.0011282972610209107            Reject H0:  True
P Value of factory 2 : 0.04033684048706412              Reject H0:  True
P Value of factory 3 : 0.06587347432044806              Reject H0:  False
```

7.2) Yes, we can conclude that factory productivity increased as a whole. Since the P Value is less than significant level. So, we reject H_0

7.3) For factory 0, 1, 2, the P value is less than significant level. So, we reject H_0 but for factory 3, the P value is greater than significant level. So, we can't reject H_0

7.4) The result of reject H_0 or not is the same as result from z-testing but the p value is a bit different since the student's t distribution is a bit different from normal distribution.

```python
from scipy.stats import t

def getPValueFromT(data):
    return t.sf(getMean(data), df=len(data)-1, loc=5000, scale=20/np.sqrt(len(data)))

print("P Value of whole factory:", getPValueFromT(allProduct), "\tReject H0: ", getPValueFromT(allProduct) < 0.05,
      "\tDifferent from z-test:", getPValueFromT(allProduct) - getPValue(allProduct))
for i in range(4):
    print("P Value of factory", i, ":", getPValueFromT(fac[i]), "\t\tReject H0: ", getPValueFromT(fac[i]) < 0.05,
      "\tDifferent from z-test:", getPValueFromT(fac[i]) - getPValue(fac[i]))
```

✓ 0.0s

```
P Value of whole factory: 2.2508681122579442e-05        Reject H0:  True        Different from z-test: 1.1154666436844853e-05
P Value of factory 0 : 0.019392070080230472             Reject H0:  True        Different from z-test: 0.004185217266497088
P Value of factory 1 : 0.002400663131852932             Reject H0:  True        Different from z-test: 0.0012723658708320214
P Value of factory 2 : 0.045629338198744374             Reject H0:  True        Different from z-test: 0.005292497711680257
P Value of factory 3 : 0.07128208428671556              Reject H0:  False       Different from z-test: 0.005408609966267505
```