# Моделирование временных рядов

## Импорт данных

import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns  
import matplotlib.dates as mdates  
# Use seaborn style defaults and set the default figure size  
sns.set(rc={'figure.figsize':(16, 4)})

## Детерминированные модели

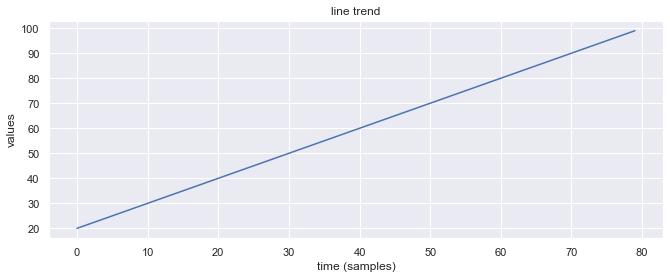
Простейшим случаем детерминированного временного ряда является одномерная (одномерная) зависимость значения от времени, представленная в следующей форме

где: - это временной ряд - набор выборок, проиндексированных некоторой переменной , обычно – это временные отметки, если временной шаг дискретный, он также может быть обозначен как (номер выборки), в этом случае в реальном времени - значение шага будет соответствовать , где - период шага (период дискретизации, с которым берутся отсчеты). - некоторый начальный постоянный уровень, - это наличие некоторого тренда, который является частью зависимости с медленным изменением. - это сезонность или некоторые «относительно быстро изменяющиеся» периодические составляющие - это относительно быстро меняющаяся часть взаимосвязи. - это некоторые периодические компоненты с "относительно медленным изменением" с нерегулярным периодом и относительно высокой интенсивностью. Часто в тренд включаются циклическая и 𝑎0 части, в этом случае модель может быть задана как

### Исследование тренда

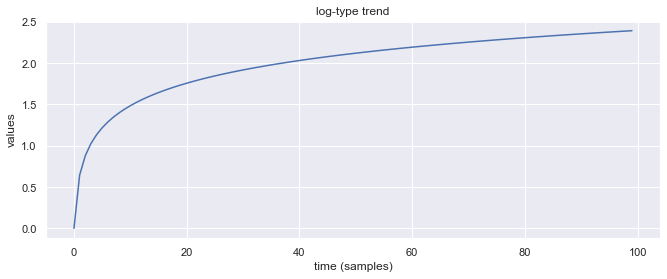
Сначала промоделируем временной ряд как имеющий только линейный тренд, взятый с единым периодом выборки.

ts = np.arange(20,100)  
fig, ax = plt.subplots()  
ax.plot(ts)  
ax.set(xlabel='time (samples)', ylabel='values',  
 title='line trend')  
plt.show()



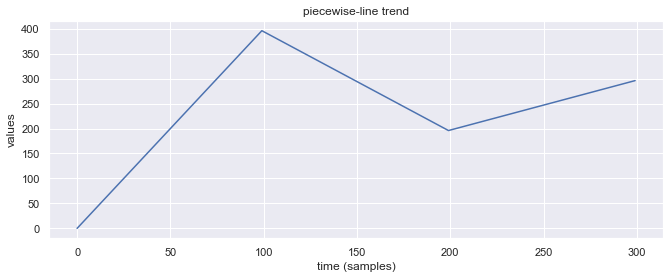
Есть несколько простейших типов трендов, которые могут быть представлены во временных рядах: Линейный тренд параболический тренд полиномиальный тренд гиперболический тренд экспоненциальный тренд насыщение (логистический) тренд   
 логарифмический тренд циклический тренд (огибающая)   
 многие другие функции, которые, как правило, сглажены, очень медленно меняются или даже монотонны.   
Теперь мы можем попробовать логарифмический тренд с основанием ( Число Эйлера, натуральный логарифм) и .

N\_OF\_SAMPLES=100 # Number of samples  
a = 4#const  
c = 0.4   
n = np.arange(N\_OF\_SAMPLES)  
ts = c\*np.log(1+a\*(n))  
  
fig, ax = plt.subplots()  
ax.plot(ts)  
ax.set(xlabel='time (samples)', ylabel='values',  
 title='log-type trend')  
plt.show()



Для многих реальных временных рядов кусочно-монотонное поведение является естественным, поэтому часто необходимо моделировать кусочно-монотонный тренд с одной или несколькими точками перегиба.

N\_OF\_SAMPLES=100 # Number of samples  
  
a = 4#const  
n = np.arange(N\_OF\_SAMPLES)  
ts1 =a\*n  
  
a = 2#const  
n = np.arange(1,N\_OF\_SAMPLES+1)  
ts2 = ts1[-1]-a\*n  
  
a = 1#const  
n = np.arange(1,N\_OF\_SAMPLES+1)  
ts3 = ts2[-1]+a\*n  
  
ts = np.concatenate((ts1,ts2,ts3))  
  
fig, ax = plt.subplots()  
ax.plot(ts)  
ax.set(xlabel='time (samples)',   
 ylabel='values',  
 title='piecewise-line trend')  
plt.show()



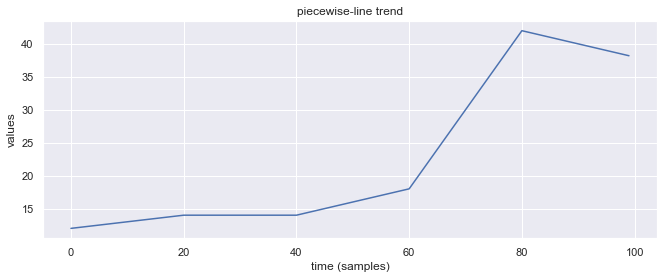
Давайте теперь смоделируем поведение кусочно-линейного тренда, предложенное моделью Prophet,

где матрица изменения роста, описывающая точки перегиба (матрица с единицами),  
 постоянная скорости роста, смещение, вектор изменения скорости роста, коэффициенты изменения роста , точки перегиба.

*Заметим*:

В простейшем случае модель сводится к $y(t) = (k)t + m, $ для временного ряда без точек перегиба . В модели Facebook Prophet предложили рассматривать логистическую модель как альтернативу линейной, в этом случае тренд можно представить в виде

N\_OF\_SAMPLES=100 # Number of samples  
  
k = 0.1  
m = 12  
  
n = np.arange(N\_OF\_SAMPLES)  
  
inflection\_points = np.array([20, 40, 60, 80])#change points  
  
a = np.zeros(shape=(inflection\_points.size, N\_OF\_SAMPLES)) # the matrix of growth changing   
  
# fill matrix  
# n[:,None] -mean add new dimention,   
#(n[:,None] > inflection\_points) is the logic operation to fill matrix with false, true  
#(n[:,None] > inflection\_points)\*1 prodece 1 for true and 0 for false  
a = ((n[:,None] > inflection\_points) \* 1).T  
  
  
delta = np.array([-0.1, 0.2, 1, -1.4])#vector with growth rate adjustments  
  
growth = (k + np.dot(a.T,delta))   
  
gamma = -inflection\_points \* delta  
offset = m + np.dot(a.T,gamma)  
  
ts = growth\* n + offset  
  
  
fig, ax = plt.subplots()  
ax.plot(ts)  
ax.set(xlabel='time (samples)',   
 ylabel='values',  
 title='piecewise-line trend')  
plt.show()



### Сезонность

Простейшую сезонную часть временного ряда можно представить в виде

где: — интенсивность сезонной составляющей; — сезонный период (месяц, день, неделя и т. д.); — начальный сдвиг (начальная фаза) сезонности; — период выборки; и — сезонная частота () и частота дискретизации . Давайте смоделируем эту серию.

Примечание В соответствии с теоремой Шеннона-Найквиста-Котельникова минимальное значение должно быть

Для оценки полученного количества периодов используйте

N\_OF\_SAMPLES=365 # Number of samples  
  
n = np.arange(N\_OF\_SAMPLES)  
  
a = 1  
  
  
Ts = 1/365  
  
T =1/3  
  
theta = np.pi/2  
  
print('number of periods = ',N\_OF\_SAMPLES\*Ts/T)  
  
ts = a\*np.sin(2\*np.pi\*n\*Ts/T+theta)  
  
fig, ax = plt.subplots()  
ax.plot(ts)  
ax.set(xlabel='time (samples)',   
 ylabel='values',  
 title='seasonal part')  
plt.show()

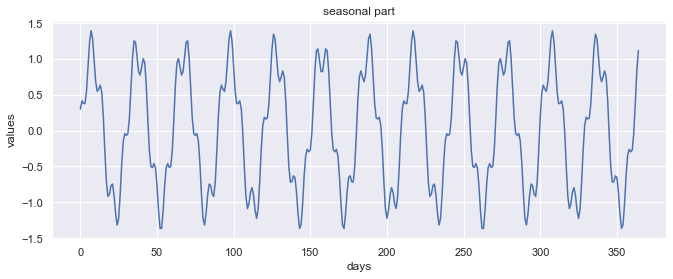
number of periods = 3.0



Давайте теперь смоделируем более сложную сезонность в году, например месяц и неделю. Мы начнем с аддитивной модели:

Отметим, что тут в отличии от циклического тренда период сезонности достаточно быстрый и в общем случае ожидается постоянным.

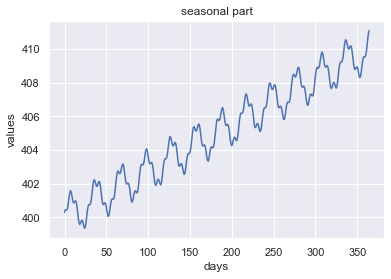
N\_OF\_DAYS=365# Number of samples  
  
days = np.arange(N\_OF\_DAYS)  
  
a\_w = 0.3 #weak influence  
  
a\_m = 1.1 #month influence  
  
T\_w = 7/365  
  
T\_m = 30/365  
  
Ts = 1/365  
  
theta\_w = np.pi/2  
  
theta\_m = 0  
  
ts = a\_w\*np.sin(2\*np.pi\*days\*Ts/T\_w + theta\_w)+a\_m\*np.sin(2\*np.pi\*days\*Ts/T\_m + theta\_m)   
  
fig, ax = plt.subplots()  
ax.plot(ts)  
ax.set(xlabel='days',   
 ylabel='values',  
 title='seasonal part')  
plt.show()



Теперь мы можем моделировать аддитивные и мультипликативные временные ряды:

В нашем случае это будет реализовано как:

YEAR = 365  
  
WEEK = 7  
  
MONTH = 30  
  
N\_OF\_DAYS=YEAR# Number of samples  
  
days = np.arange(N\_OF\_DAYS)  
  
a\_w = 0.3 #weak influence  
  
a\_m = 1.1 #month influence  
  
T\_w = WEEK/YEAR  
  
T\_m = MONTH/YEAR  
  
Ts = 1/YEAR  
  
theta\_w = np.pi/2  
  
theta\_m = 0  
  
a\_trend = 10 #slope  
  
bias\_trend = 400  
  
trend = a\_trend\*days\*Ts+bias\_trend   
  
seasonality = a\_w\*np.sin(2\*np.pi\*days\*Ts/T\_w + theta\_w)+a\_m\*np.sin(2\*np.pi\*days\*Ts/T\_m + theta\_m)   
  
ts =trend + seasonality  
  
fig, ax = plt.subplots()  
ax.plot(ts)  
ax.set(xlabel='days',   
 ylabel='values',  
 title='seasonal part')  
plt.show()

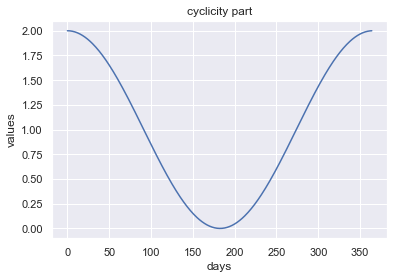


### Цикличность тренда

Помимо тренда и сезонности мы можем добавить некоторую цикличность (как альтернативу можно рассматривать как дополнение трендового поведения).

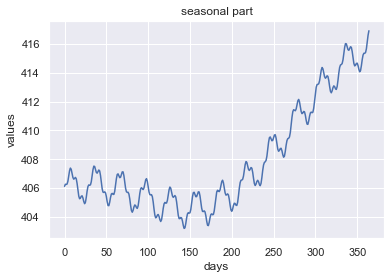
Давайте смоделируем это как некоторую зависимость год-сезон. Например, в приведенном ниже примере мы моделируем падение продаж в середине года (летом).

a\_cycl = 1  
T\_cycl = 1  
cyclicity = a\_cycl +a\_cycl \*np.sin(2\*np.pi\*days\*Ts/T\_cycl + np.pi/2)  
  
ts =cyclicity  
  
fig, ax = plt.subplots()  
ax.plot(ts)  
ax.set(xlabel='days',   
 ylabel='values',  
 title='cyclicity part')  
plt.show()



Теперь мы можем добавить цикличность к линейному тренду

YEAR = 365  
  
WEEK = 7  
  
MONTH = 30  
  
N\_OF\_DAYS=YEAR# Number of samples  
  
days = np.arange(N\_OF\_DAYS)  
  
a\_w = 0.3 #weak influence  
  
a\_m = 1.1 #month influence  
  
T\_w = WEEK/YEAR  
  
T\_m = MONTH/YEAR  
  
Ts = 1/YEAR  
  
theta\_w = np.pi/2  
  
theta\_m = 0  
  
a\_trend = 10 #slope  
  
bias\_trend = 400  
  
trend = a\_trend\*days\*Ts+bias\_trend   
  
seasonality = a\_w\*np.sin(2\*np.pi\*days\*Ts/T\_w + theta\_w)+a\_m\*np.sin(2\*np.pi\*days\*Ts/T\_m + theta\_m)   
  
a\_cycl = 2.91  
T\_cycl = 1  
cyclicity = a\_cycl+a\_cycl \*np.sin(2\*np.pi\*days\*Ts/T\_cycl + np.pi/2)  
  
ts =trend + seasonality + cyclicity  
  
fig, ax = plt.subplots()  
ax.plot(ts)  
ax.set(xlabel='days',   
 ylabel='values',  
 title='seasonal part')  
plt.show()



### Особые события

Помимо тренда и регулярной сезонности, в модель временных рядов могут быть введены "особые события".

Например, если мы моделируем временные ряды продаж, будет интересно добавить некоторые изменения спроса в рабочие дни.

В простейшем случае это можно сделать следующим образом:

где - означает остаток деления; — дельта-функция Кронекера,

i — номер дня ().

*Примечание* Если вы хотите считать дни не с понедельника, используйте

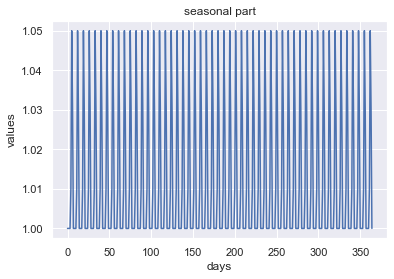
где  – количество дней, на которые нужно сдвинуться. Давайте проверим этот результат

N\_OF\_DAYS =14  
shift = 1  
days = np.arange(1,N\_OF\_DAYS+1)  
print((days-1+(shift-1))%7+1)

[1 2 3 4 5 6 7 1 2 3 4 5 6 7]

Для моделирования поведения дней недели мы будем использовать набор из 7 коэффициентов

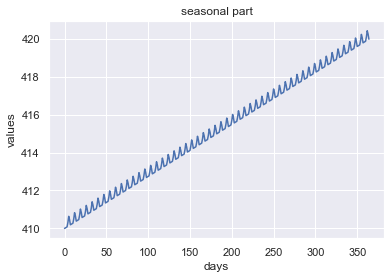
N\_OF\_DAYS=365  
  
days = np.arange(N\_OF\_SAMPLES)  
  
# week days coefficients  
a\_week = np.array([1, 1, 1, 1, 1.01, 1.05, 1.03])  
  
#for the number of days multiples of the week  
week\_days = list(a\_week)\*int(N\_OF\_DAYS/7)   
  
# add rest of the days  
week\_days = np.array([\*week\_days,\*a\_week[:N\_OF\_DAYS%7]])  
  
#check that week\_days size equal to N\_OF\_DAYS  
assert week\_days.size==N\_OF\_DAYS  
  
ts = week\_days  
  
fig, ax = plt.subplots()  
ax.plot(ts)  
ax.set(xlabel='days',   
 ylabel='values',  
 title='seasonal part')  
plt.show()



Для имитации влияния дней недели на линейный тренд можно ввести следующую модель

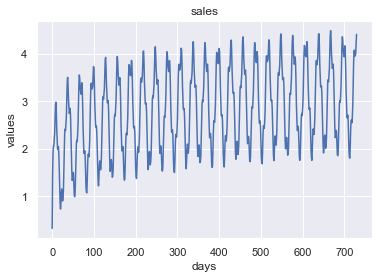
где модель дней недели с трендом

N\_OF\_DAYS = 365  
days = np.arange(N\_OF\_SAMPLES)  
  
a\_trend = 10 #slope  
bias\_trend = 400  
week\_coefficients = np.array([1, 1, 1, 1, 1.02, 1.05, 1.03])  
  
a\_week = week\_coefficients\*a\_trend  
  
week\_days = np.array([\*list(a\_week)\*int(N\_OF\_DAYS/7), \*a\_week[:N\_OF\_DAYS%7]])  
  
trend = a\_trend\*n\*Ts+bias\_trend   
  
ts =week\_days + trend   
  
fig, ax = plt.subplots()  
ax.plot(ts)  
ax.set(xlabel='days',   
 ylabel='values',  
 title='seasonal part')  
plt.show()



Давайте теперь добавим часть сезонности и создадим модель на два года с логистическим трендом.

YEAR = 365  
  
WEEK = 7  
  
MONTH = 30  
  
N\_OF\_DAYS=YEAR\*2# Number of samples  
  
days = np.arange(N\_OF\_DAYS)  
  
a\_w = 0.3 #weak influence  
  
a\_m = 1.1 #month influence  
  
T\_w = WEEK/YEAR  
  
T\_m = MONTH/YEAR  
  
Ts = 1/YEAR  
  
a\_trend = 5   
c\_trend = 0.34   
  
week\_coefficients = np.array([0.95, 1, 1, 1, 1, 1.25, 1.03])  
  
a\_week = week\_coefficients\*c\_trend  
  
trend = c\_trend\*np.log(1+a\_trend\*days)  
  
seasonality = a\_w\*np.sin(2\*np.pi\*days\*Ts/T\_w )+a\_m\*np.sin(2\*np.pi\*days\*Ts/T\_m )   
  
  
week\_days = np.array([\*list(a\_week)\*int(N\_OF\_DAYS/7), \*a\_week[:N\_OF\_DAYS%7]])  
  
  
ts = week\_days + trend + seasonality   
  
fig, ax = plt.subplots()  
ax.plot(ts)  
ax.set(xlabel='days',   
 ylabel='values',  
 title='sales')  
plt.show()



## Моделирование шумов

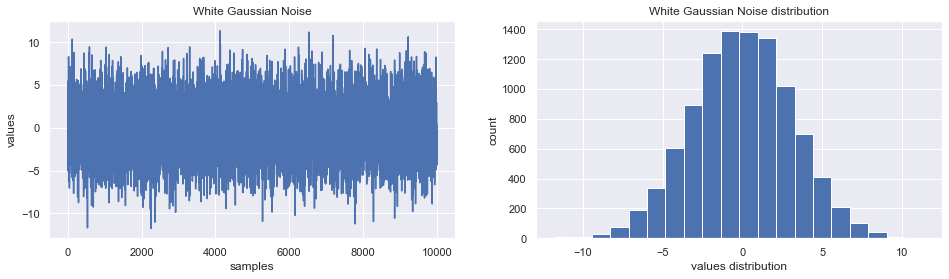
### Белый Гауссов Шум

Помимо детерминированной части временного ряда, важно смоделировать его стохастическое поведение. Стохастическое поведение временного ряда в первую очередь связано с влиянием шума. Наиболее простой и наиболее распространенной моделью шума является Белый Гауссов шум (White Gaussian Noise, WGN) (идентичен понятию независимой и одинаково распределенной величины (independent and identically distributed, i.i.d) модели шума). WGN имеет нормальное распределение с нулевым средним значением и дисперсией . Шум обычно обозначается как

а его распределение задается функцией плотности вероятности:

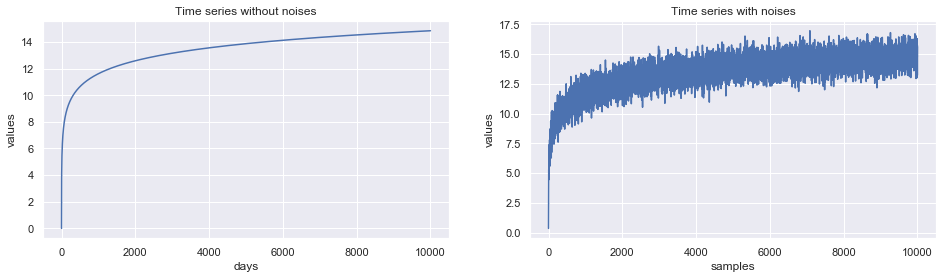
где мощность шума - стандартное отклонение шумов (квадратный корень из дисперсии). Давайте смоделируем белый гауссовский шум.

N\_OF\_SAMPLES = 10000  
  
noise\_power = 10   
  
wgn = np.sqrt(noise\_power)\*np.random.normal(size = N\_OF\_SAMPLES)  
  
ts = wgn   
  
fig, ax = plt.subplots(1,2)  
  
ax[0].plot(ts)  
ax[0].set(xlabel='samples',   
 ylabel='values',  
 title='White Gaussian Noise')  
  
ax[1].hist(ts, bins = 20)  
ax[1].set(xlabel='values distribution',   
 ylabel='count',  
 title='White Gaussian Noise distribution')  
plt.show()



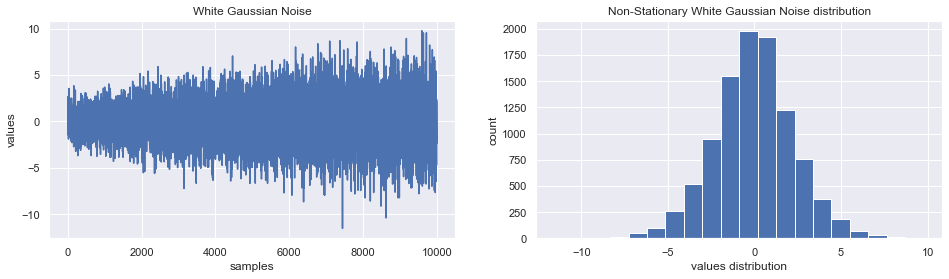
Таким образом, мы можем видеть, как шум влияет на временной ряд.

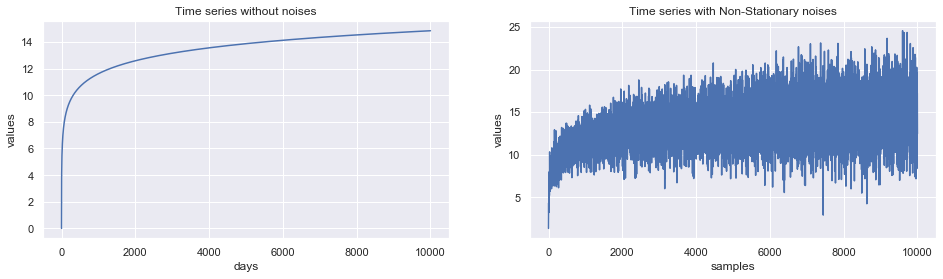
N\_OF\_SAMPLES = 10000  
  
noise\_power = 0.5   
  
wgn = (np.sqrt(noise\_power))\*(np.random.normal(size = N\_OF\_SAMPLES))  
  
a = 4#const  
c = 1.4   
n = np.arange(N\_OF\_SAMPLES)  
ts = c\*np.log(1+a\*(n))  
  
ts\_wn = ts + wgn  
  
fig, ax = plt.subplots(1,2)  
  
ax[0].plot(ts)  
ax[0].set(xlabel='days',   
 ylabel='values',  
 title='Time series without noises')  
  
ax[1].plot(ts\_wn)  
ax[1].set(xlabel='samples',   
 ylabel='values',  
 title='Time series with noises')  
plt.show()



Помимо одинаково распределенного шума, соответствующего стационарной модели шума, важно моделировать нестационарные случаи. Самый простой случай — линейно возрастающая вариация,

N\_OF\_SAMPLES = 10000  
a = 4#const  
c = 1.4   
  
noise\_power = np.linspace(1,10,N\_OF\_SAMPLES) #linearly growing noise power   
  
wgn = np.sqrt(noise\_power)\*np.random.normal(size = N\_OF\_SAMPLES)  
  
ts = c\*np.log(1+a\*np.arange(N\_OF\_SAMPLES))  
  
ts\_wn = ts + wgn  
  
fig, ax = plt.subplots(1,2)  
  
ax[0].plot(wgn)  
ax[0].set(xlabel='samples',   
 ylabel='values',  
 title='White Gaussian Noise')  
  
ax[1].hist(wgn, bins = 20)  
ax[1].set(xlabel='values distribution',   
 ylabel='count',  
 title='Non-Stationary White Gaussian Noise distribution')  
fig, ax = plt.subplots(1,2)  
ax[0].plot(ts)  
ax[0].set(xlabel='days',   
 ylabel='values',  
 title='Time series without noises')  
  
ax[1].plot(ts\_wn)  
ax[1].set(xlabel='samples',   
 ylabel='values',  
 title='Time series with Non-Stationary noises')  
plt.show()



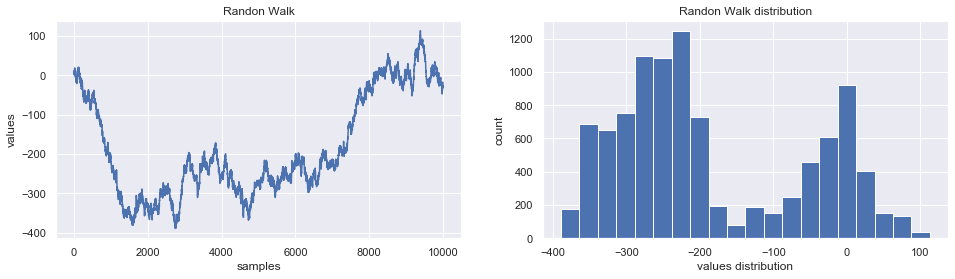


### Случайное блуждание

Помимо аддитивного шума, важной моделью шума является случайное блуждание, которое в простейшем случае можно смоделировать как

где Модель широко распространена при исследовании ряда бизнес-процессов.

N\_OF\_SAMPLES = 10000  
  
noise\_power = 10  
  
wgn = np.sqrt(noise\_power)\*np.random.normal(size = N\_OF\_SAMPLES)  
  
ts = np.cumsum(wgn )  
  
fig, ax = plt.subplots(1,2)  
  
ax[0].plot(ts)  
ax[0].set(xlabel='samples',   
 ylabel='values',  
 title='Randon Walk')  
  
ax[1].hist(ts, bins = 20)  
ax[1].set(xlabel='values distribution',   
 ylabel='count',  
 title='Randon Walk distribution')  
plt.show()



### Упражнения для самоконтроля (необязательные) 1

1. Реализуйте модель логистического тренда Facebook Prophet
2. Для модели тренда добавьте аддитивную квартальную сезонность.
3. Для модели тренда добавьте мультипликативную ежемесячную сезонность.
4. Реализуйте мультипликативную детерминированную модель временных рядов с сезонной, циклической и трендовой частями.
5. К модели добавьте падение спроса в праздничные дни в начале года.

### Упражнения для самоконтроля (необязательные) 2

1. Исследовать влияние аддитивного стационарного и нестационарного белого шума на временные ряды с сезонными частями и частями тренда в следующей форме
2. Смоделировать модель временного ряда в следующем виде
3. Исследуйте 3 модели случайного блуждания:
   * Модель с дрейфом
   * Модель с трендом
   * Модель с изменением интенсивности

**Вопросы**

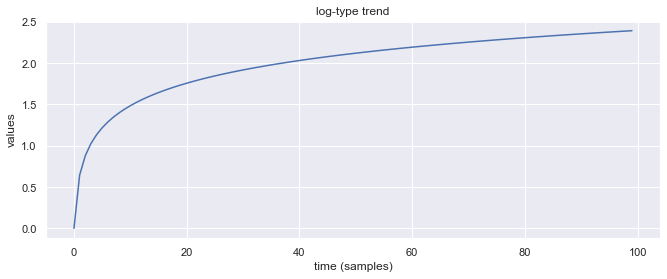
1 Если период сезонности 10 отчетов, то какой максимальный шаг в отчетах можно выбрать. Ответ цифрой

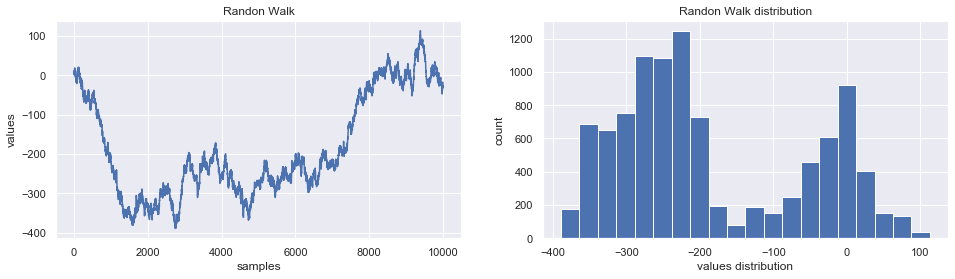
Ответ 5

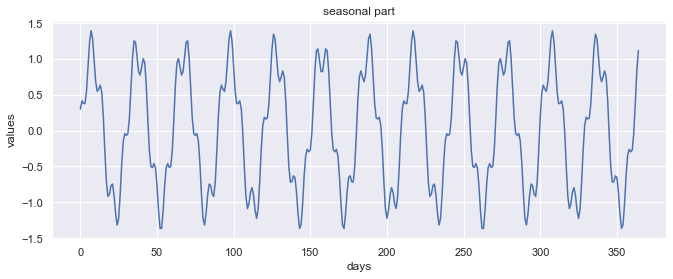
2 Если моделируется ВР типа y(n) = 7n + 3sin( 2π10n/100) (вопрос можно сделать параметрическим), то какими будут период, интенсивность сезонной составляющей и шаг

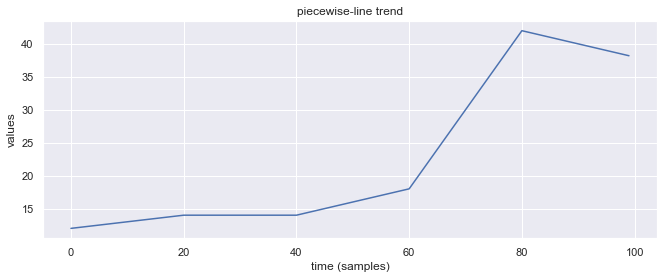
Ответ 10 3 100

*3 соотнесите*

 *тренд с насыщением*

 стохастический тренд

 многокомпонентная сезонность

 Тренд с точками перегиба