Setwd ("c:/Users/abc/Desktop/R Programming")
getwd ()

* setting wd on starting R session makes importing, exporting data files, code files etc easier

* R commands are mostly in form of func^n.
  plot(x,y), mean(x)
  - We can assign values to x and y ourselves.
  - or import a dataset that contains x and y.

* Create your own objects (Variables, Vectors, matrices, lists etc.)
  - access names of these objects.

* Class - to check data type of
* c(1,2,3) → combining func^n → Vector

* m ← matrix (1:4,2,2)
  c ← c(5,6)
  m_new ← cbind (m,c)
  m_new

1	3	5
2	4	6

* To [ ] → to select one or multiple elements from a matrix.
  my_matrix [1,2] → selects element at first row and second column.

* To select all elements of a row or a column, no number is needed before or after comma.
  my_matrix [,1] → selects elements of first column
  my_matrix [1,] → selects all elements of first row

Scanned by CamScanner

* Data frame (2d object)
- can hold numeric, character or logical values
- within column - same data type
  different column - different " "

* List

my_list <- list (my_vector, my_matrix)
my_vector <- 1:10
my_matrix <- matrix (1:9, ncol=3)
my_list[[2]] gives

1	4	7
2	5	8
3	6	9

* Packages - Collection of R functions and data sets.
* install.packages ("package name") and select the CRAN mirror closest to your location.
* Once installed we need to call the package in when needed using library (package name)

Scanned by CamScanner

Data by choice (databychoice.com) Sudipt Roy
Prof.Roy123@gmail.com
Philip Cotler (B2C)

Retail and Marketing Analysis

mainly about understanding ~~your~~ customer behaviour and preferences.

Bank tie-ups with financial institutions to cross sell products -

Prediction - way to think/decide for future, Based on understanding historical data.

→ In B2B, biggest expenditure is on Sales and marketing (40%)

→ Analytics not necessarily for retail. Can be for any business or operations. For ex computers/operations efficiency

→ Finance area, lot of compliance requirements, data is of better quality

→ In Retail, very less compliance requirements, data quality can be very bad. They do not capture all fields.

→ Amazon recommendations started ~~neo~~ in late 90's.

→ Money Ball moments - Bill James: Used statistics to build a world winning cricket team.

→ Competing on Analytics (Harvard Paper) by Davenport (2006)

→ Gary Loneman - Pioneer of analytics in marketing. - analyzed casino data, Started/segregating customers based on loyalty tiers (platinum, Diamond, Gold), Several other policy experiments using data.

→ Rise of analytics - Facilitating factors

- Rise of data
- Increasing use of computers and computing Power

→ If I spend 1$ on Adv. — Impact on demand is 0-0.1%

Adv. Elasticity of demand → 0-0.1%

If " " 1$ on Marketing analytics, Impact on demand 39%

Scanned by CamScanner

∴ Incremental benefit of spending money on analytics is much higher.

Use case of a company that sold products in rural india.

→ 6.5 lakh villages
→ Understand the demand of the product in each village, Tell market size accordingly.
→ Plan marketing resources according to that
  → where to deploy sales people
→ Bottom up approach from village to sub-district to district to state

→ For visualization "Vizard" tool was used.
→ Not just data crunching, lot of data presentations

Decisions to Data and not data to decisions.
→ We cant stop making decisions if we dont have data.
→ Data is often not prepared for the kind of questions we are asking.
→ Take a decision (problem statement) first and then build your data.

→ Village level data collected - area, crop, soil conditions, irrigation, Prices of crops (or any other product/fertilizer/seeds)
→ Segmentation of villages like top 10, top 25, rest, in a particular sub-district. Segmentation based on market size at village level. and then tell where dealers are available.

Scanned by CamScanner

Understanding Customer Segmentation

Segmentation is to understand customers and to focus which one is our market and which is not
So that unnecessary money does not go into reaching out to non-customers.

→ Segmentation - Ist step in marketing.
→ Big mistake by most companies - They do not do segmentation - serves everyone equally (commoditization)
It should be done to move away from clutter.
→ As a marketer, struggle is to differentiate ourselves with rest, for which we need to do segmentation.
(generally commodity does not matter, image matters)
Clear segmentation gives target market
→ ~~Three~~ IInd mistakes, even if people segmentation their market, they find it difficult to not to reach "not a very ~~of~~ high-selling" segment.
Mental block of companies here (how can we leave a single ~~cluster~~ customer)

K-means vs KNN

Unsupervised	supervised
doesnot Know classes	You already Know classes and predict based on that
clustering	classification

expedia inc collects data about the device also which a customer uses for booking.

An apple user can be offered a high price hotel as that user can either afford or has ~~tad~~ taste for finer things in life (premium room

However other device user wont pay this premium.

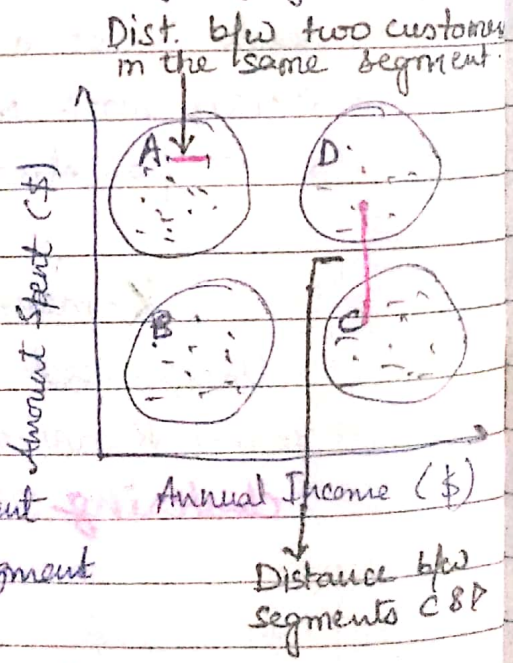
So expedia would show different products based on their customer segment.

Hence segmentation is to prioritize by customers, who to reach and what to offer them based on their needs

So STP gives a strategy which is vs Everything to Everybody.

Drawback of segmentation - You may get 'stuck' if focussing and targeting just one segment.

Effective segmentation



(similarity) homogenity within
(dissimilarity) hetrogenity across

Segmentation - Group customers based on similar needs, Profile each segment

Targeting - Assess attractiveness of each segment
Select segments to target

Positioning - Define value proposition for target segments. Develop an action plan

Scanned by CamScanner

Cambridge analytica — used facebook 'likes' to understand the person. Based on likes, they could predict age, gender, income etc (market segmentation using facebook data)

Bases for segmentation

Bases

- (mostly used) Observable Factors
  - B2C
    - Demographics
      - Age
      - Income
      - Gender
      - Marital status
      - ...
  - B2B
    - Firmographics
      - Age
      - Turnover
      - Industry
      - Ownership
      - ...
- Unobservable factors (not used frequently but most fruitful)
  - Psychographic
    - personality traits
    - Interests
    - Values
    - Lifestyle
    - Attitude
    - ...

Others : Behavioural, Benefits sought

facebook, Google working on psychographic (unobservable) factors to know everything about a person.

Hence all the marketing budget goes to these 2 companies today.

Scanned by CamScanner

Methodologies for segmentation

Few leading methodologies
- Correlation, ANOVA
- Cluster Analysis
- Tree (CHAID)
- RFM
- Conjoint Analysis
- Latent Class Analysis

Data Type and Clustering Methodologies

Data Type	k-Means	Hierarchical	K-Mode	Mixed
Quantitative	✓	✓		✓
Ordinal			✓	✓
Nominal			✓	✓

Case Study - Couch Potato

To segment the market (customers) based on demographics (age, gender, income) and then do territory mapping (where are my people, my business and how can we reach to them)

K-means + works on distance. For categorical data K-means can't be used. Can be used only for (1) continuous data. It is a drawback of K-means. Secondly, we have to Scale the variables. (2)

Scanned by CamScanner

less clusters - less variability is captured (too generic)
more clusters - very specific to those data points

So that all the variables get same weightage

- Reading the data
- Scaling the data (Imp. step for K means)
- Correlation to understand (weak correlation in this)
- Convert to data frames.
- We ran k means algorithm with diff k values (2 to 20 to find ~~best~~ optimum number of clusters). Also, we run 100 iterations to get 'global' minimum of "Sum of square error within"
- Setting seed value to tell you where to start, in order to get global optima.

(3) Biggest challenge with K-means is to get global optima

Build in your code to find Global minimum

Strategy by looking at the results

- Have product line, to address the need to different customs like
  - Higher price package for smaller market (less viewership)
  - Lower price package for large market (who watches more)

Part worth - Product attributes not customer attributes
menu vs kids v/s family vs price

Hierarical clustering - check code file 'Sudipt's code TV Hierarical clustering'. It also gives around 4 to 6 clusters

① Need not decide K while running the model

② Agglomerative process - each point is placed in different clusters and then ~~etc~~ those clusters are clubbed together based on distance b/w 2 points, till we get single cluster. Then the tree is cut

③ Cant work with very large dataset as it takes a ~~big~~ lot of time to converge for large datasets. K-means is more practical approach in such cases.

TiVo Case

TiVo case is about a new product of 1990's which allowed users to record the episodes, to view later with family, skip adds etc

Shows Clustering with mixed data (Categorical as well as continuous variables)

Gender, Work status, Education, Purchase Location, Technology adoption, TV viewing, Decision making (Family/Single), Purchase Freq, Favourite features

As there are too many categories, so we need to make it managble (1280 possible levels by multiplying each category level - 2X2X4X2X4X2X5 = 1280) for analysis.

→ PCA is a challenge as it gives factors which

are difficult to interpret

So, I have dropped Gender, age and other variables as first cut analysis (Need to do multiple analysis in real life situation)

Moved favourite features from 5 to 2, i.e instead of using all features, I am classifying them as Technology feature or family related feature.

If purchase is done by family, then Family favourite feature matters So removed marital status and Gender from analysis.

However, if we want to target individual we will redo analysis with age, gender etc as well.

2 levels purchase decision maker, 2 levels of Technology adopters and 2 levels of favourite feature (8 levels)

Basically dropped highly correlated values as first cut.

- Use Gower distance to handle mixed data.
- Scale the data first
- Dice coefficient - Suppose we have 2 factors, A ~~one~~ with values 1 to 3 and B have values 1 to 2. Dice coefficient does exactly the same as what euclidean distance does for continuous (Similarity index measurement)

It tries to find out how close A and B are. Relative agreement matrix is created for each variable which tells 0 and 1, ~~whether each~~

~~were~~

Silhouette tells distance b/w segments

Dice coefficient - Distance for categorical data

ID	A	B	A	1	2	3	4	5
1	2	1	1	1				
2	1	2	2	0	1			
3	3	2	3	0	0	1		
4	1	1	4	0	1	0	1	
5	2	1	5	1	0	0	0	1

B	1	2	3	4	5
1	1				
2	0	1			
3	0	1	1		
4	1	0	0	1	
5	1	0	0	0	1

Data

	1	2	3	4	5
1	(1+1)/2				
2	(0+0)/2				
3	(0+0)/2				
4	(0+1)/2				
5	(1+1)/2				

Relative agreement

	1	2	3	4	5
1	1				
2	0				
3	0				
4	0.5				
5	1				

Dice coefficient

Gower is overall term which calculates both euclidean and dice coefficients

for continuous for categorical

1 and 5 rows are full similar

2 and 4 are half similar

For all categorical variables, we have made 1 dice coefficients

for code, refer to

Silhouette width is opposite of TS within, a way to understand how well seperated these segments are so it has to be maximum

* More the silhouette width better results
* Farther the Clusters they are well defined

Cluster Solution

	Cluster	D.Maker	TechAdopt	FF mod	Income	Elec-Spend	HH Spend	Purch Freq	TV Hrs
Family oriented ←	1	F : 180	E : 0	(Family oriented) F : 200	Median .290	Median -0.069	Median 0.480	Median 0.505	Median 0.904
		S : 20	L : 200	P : 0	Mean .293	Mean -0.055	Mean 0.490	Mean 0.501	Mean 1.359
	2	F : 380	E : 800	F : 0	Median -0.258	Median -0.069	Median -0.458	Median -0.287	Median -0.471
		S : 420	L : 0	P : 800	Mean -0.073	Mean 0.014	Median -0.123	Median -0.125	Median -0.340

Product oriented features

Recommendation strategy for TiVo

↳ Family oriented features

1st cluster is family oriented, uses family oriented features, late technology adopters, buy less frequently. Income is more, TV viewership is more

2nd is technology focussed, early Tech adaptors more individualists, buy more frequently.

Question – Is TV as a Technology product would fly or as a family oriented product would fly

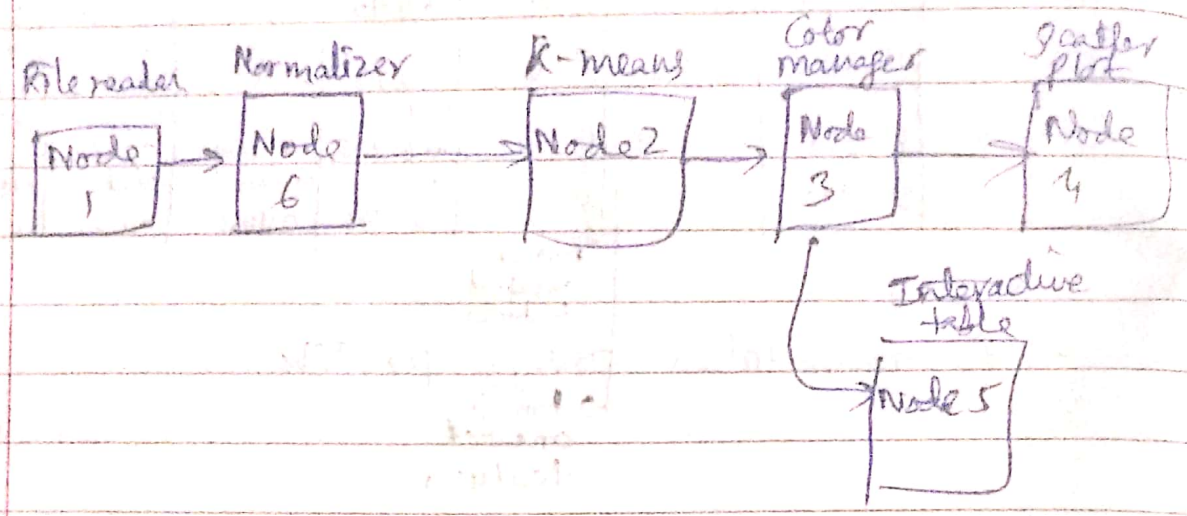
At this stage, positioning TiVo as Technology product, it is not going to fly, by looking at current data.

So you would advertise it as family product.

Scanned by CamScanner

KNIME

You create a workflow, pick components and place in the workflow



After dragging and placing nodes at right place Give output of previous to next and then Configure the nodes. Start running each node.

Scanned by CamScanner

RFM Analysis

Objective – RFM tells who are my core customers (buy often, buy lot and who has bought recently) so we need to protect them

New customers – who has brought recently → need to grow them

High value core defactors – bought from me in the past but not buying from me anymore. → need to bring them back

Frequency	Monetary	Recency: recent	Recency: far
One	low	First time customers	Low targeting value
	medium		
	High		
A few	Low	Early repeat customers	Non-core defectors
	Medium		
	High		
More than a few	Low	Core (high value) customers	Core defectors
	medium		
	High		

Now to pick example with real data, we need to bin our values.

Core defectors

New customers

# of customers in each segment	Frequency	Monetary	<=5	Recency >5 and <12	>12	Total
Total 27 segments 3X3X3	1	<=40	25	45	150	220
		41-80	30	80	244	354
		>80	18	44	92	154
	2-3	<=40	16	39	56	111
		41-80	31	53	112	196
		>80	18	41	43	102
Core customers	>3	<=40	19	18	12	49
		41-80	35	46	10	91
		>80	28	32	5	65

Scanned by CamScanner

Total revenue of each segment - To do Break-Even analysis

Catalog Cost = $8

Catalog cost per year = $16

Profit margin = 20%

Break even revenue = $80

If profit margin is 20% then we need to earn revenue of $80 in order to extract $16 catalog cost

Frequency	Monetary	Recency <=5	>5 and <=12	>12
1	<=40	$26.28	$31.76	29.6
	40-80	$54.10	60.44	58.9
	>80	$112.89	107.02	110.9
2-3	<=40	$76.06	71.87	68.2
	40-80	$133.16	126.60	126.81
	>80	$255.89	262.5	233.4
>3	<=40	$153.68	148.33	133.5
	41-80	$377.11	289.57	268.0
	>80	$720.54	528.25	446.8

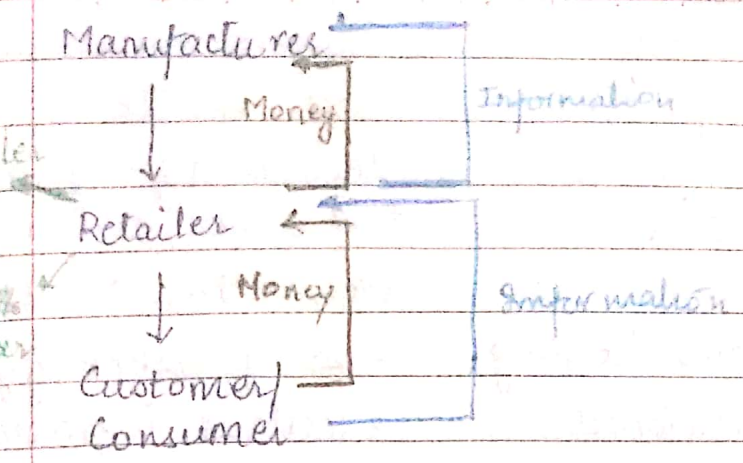
Target segments which are giving >80 revenue in order to ~~not~~ extract catalog cost.

Total # of customers above break even point

= 657 = 49%

Scanned by CamScanner

Retailer.



Retail involves the process of selling consumer goods or services to end-customers through multiple channels of distribution to earn profit

In few cases, retailers may also have businesses as customers (E.g. Amazonbussiness.in)

Power of retailers | resellers - If retail do not make enough from product they are going to kill it

Incentive schemes of retailers and at same time protecting company interest.

Retail in India - 10% of GDP.
- Provides 8% of employment.

Basics of Retail

>90% of retail business

Tradition Trade (TT): often situated nearby the residential area and catering to the needs of the populace residing in the locality. These are owner-managed and employing personal capital. E.g. Kirana Shops.

Modern Trade (MT): These are characterised by high capital requirements, trained staff, large premises, wide variety of products and multi-locational store chain.

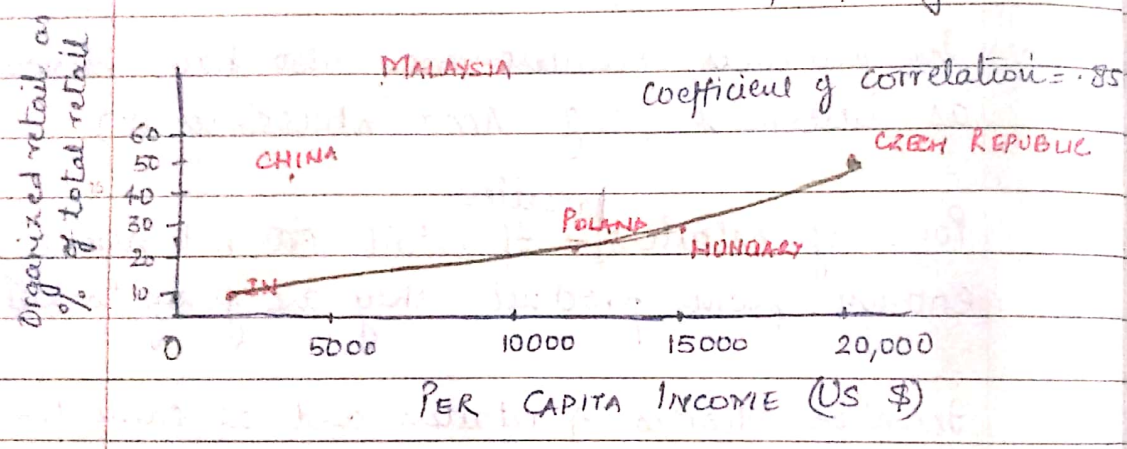
7-8% of total retail

AIDA model – Attention, Interest, Desire, Action model
Customers are dealt in diff ways based on their stage

Traditional Vs Modern Trade

①	Owner driven	Professional driven
②	Active salespersons	self-service
③	Traditional format	Several diff. formats
④	Small in size	Different sizes but larger than TT stores.
⑤	Manual book keeping, frugal infrastructure	Use of technology, excellent infrastructure
⑥	More as gate keeper	Active participation of 4Ps of marketing.

What drives growth of MT – Money, Economic prosperity



WELL KNOWN RETAIL FORMATS

PHYSICAL

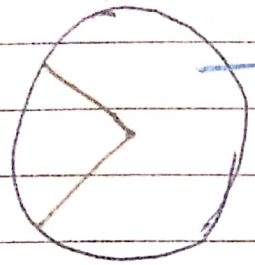
① Flagship store – Apple.
② High street store – stores selling branded good e.g. Tiffany.
③ Mall store – store in Mall.
④ Store-within-a store – ~~Store~~ well known Sweet shop in Grocery store.
⑤ Kiosk/Stand-alone – Watch/perfume store kiosk in lifestyle.
⑥ Pop-up store – comes up in response to sudden opportunity e.g. Holi/Diwali

⑦ Travel Retail (airport etc.)

VIRTUAL

Oldest virtual store – ~~Tele shopping~~ does not depend on intern

① Catalogues
② Online store – one to many.
③ Market place – Many to many.
④ Click-and Collect (Brick and Click)



Non Grocery (28%)

→ Grocery (72%) → 3/4 of entire retail business

→ sells fridge, furniture, Consumer durables everything.

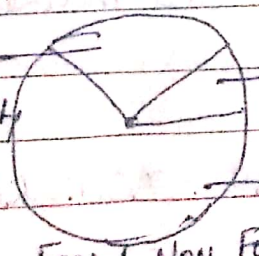
TYPES OF RETAIL FORMATS: FMCG & CD

① Department stores (E.g. Shopper's Stop)
② Hypermarkets (E.g. Hyper City) Low Margin High Volume
③ Supermarkets (E.g. Food World)
④ Convience stores (E.g. 24x7)
⑤ Discounters (E.g. Brand Factory for Apparels)
⑥ Cash and Carry (E.g. Metro)
↳ for B2B

→ Big format selling consumer durables smaller than department stores.

★ INSIGHT INTO BUYER BEHAVIOUR IN MODERN TRADE.
understanding buyer needs, behaviour, recommend etc

INDIAN CONSUMER: PLANNED Vs UNPLANNED



REMINDED 29% (Price sensitivity is less)

Impulse 12% (Price sensitivity is min^m)

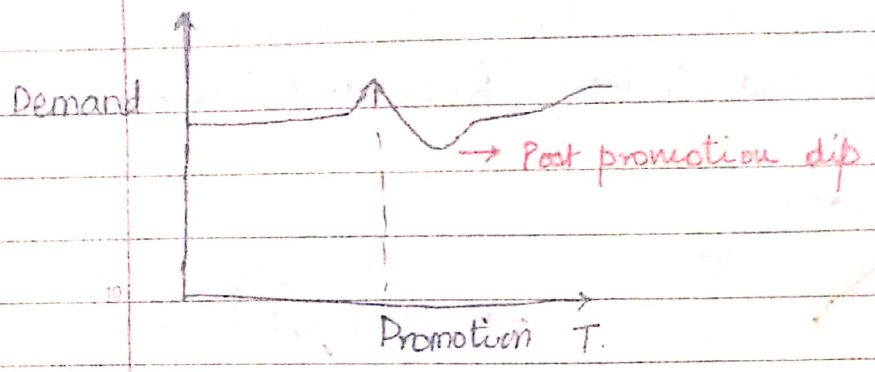
PLAN 59% (~~Price~~ sensitive to price)

FOOD / NON FOOD PURCHASE

Scanned by CamScanner

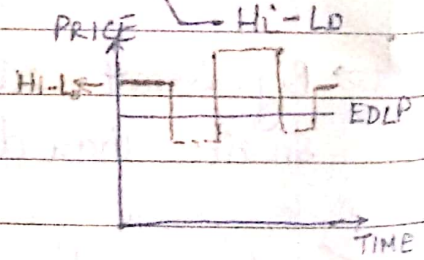
* Even though majority of purchases are planned, retailers could, potentially, influence $7bn of purchase in MT.

Retailer - Customer Interactions: Leveraging Data



Influencing purchase decision in Retail - Issues

1. Location of the store
2. Which Categories (Some product categories are more profitable than other)
   - Brand
     - National
     - Store
3. Which Store
4. Pack sizes
5. At what price - today, tomorrow
   - Pricing Format
     - Every Day Low Price (EDLP)
     - Hi-Lo



6. Relative price of brands and SKUs
7. Co-locating categories and market basket
8. Shelf space, display
9. Newspaper inserts
10. Shopping environment - AC, Parking, Check-out

Scanned by CamScanner

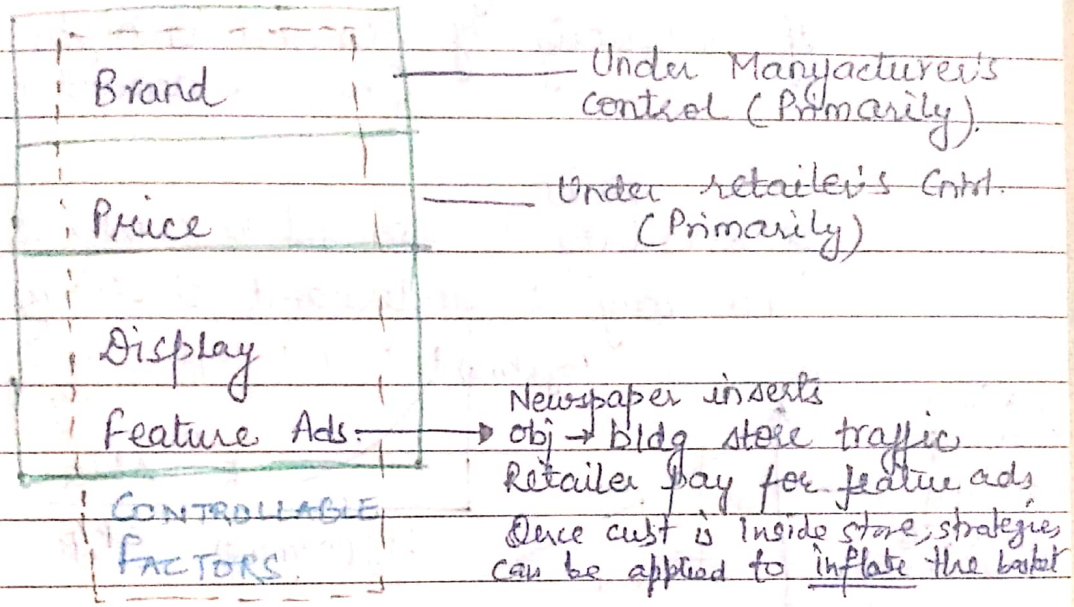
Influencing Purchase decision in Retail – Common levers

★ 3 main marketing ~~leve~~ mix | lever that retailer has in short run are

Price — PDF

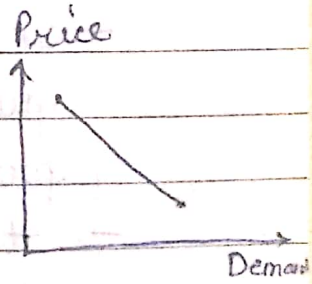
Display – where to display (end/front aisle, back)

Feature – featured ads (eg newspaper) to bring customer to store.



* Display is also spend cost for manufacturer

Retailer – Customer Interactions: Own, Cross-Price Elasticity and Demand Prediction



$$E_{\text{of demand}} = \frac{x\% \text{ change in demand}}{1\% \text{ Price change}}$$

If $E = 1$ (perfectly elastic)

< 1 (Inelastic)

When price change by 1%, demand change by < 1%.

i.e. Impact of price on demand is less

If $E > 1$, small change in price would lead to larger change in demand

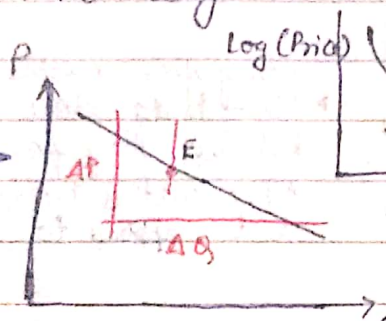
If Price Elasticity $E < 1$, price strategy should be to increase

e.g. Appl

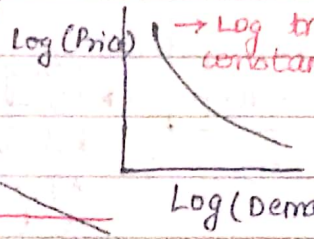
If Elasticity of demand is low, ~~[illegible]~~ raise price

If Elasticity of demand is high, reduce price (as more demand will compensat margin)

If ~~Elasticity of demand~~ Price is constant, there is no way to understand elasticity of demand.



This is a changing elasticity curve. As elasticity is changing with changing price. To have a "Constant" elasticity curve, need to take log.



→ Log transformation give constant elasticity of demand curve

$$E = \frac{\Delta Q/Q}{\Delta P/P} = \frac{dQ/Q}{dP/P} = \frac{\log Q}{\log P}$$

$$\frac{d(\log x)}{dx} = \frac{1}{x}$$

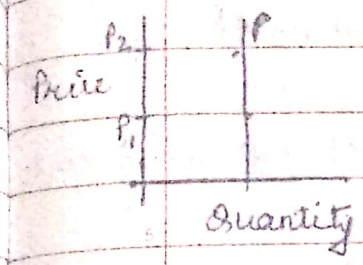
Price elasticity of demand -

- It is the measurement of change in the quantity of demand due to change in price of the commodity.
- It refers to the degree of responsiveness of quantity of demand due to change in price.

$$e = \frac{\text{Proportionate change in quantity demand}}{\text{Proportionate change in price}}$$

Scanned by CamScanner

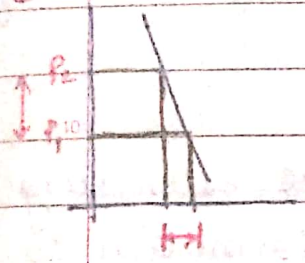
$e = 0$
Perfectly Inelastic



With any change in price, demand does not change.

~~With Pen~~

$e < 1$ (less elastic) → for myself

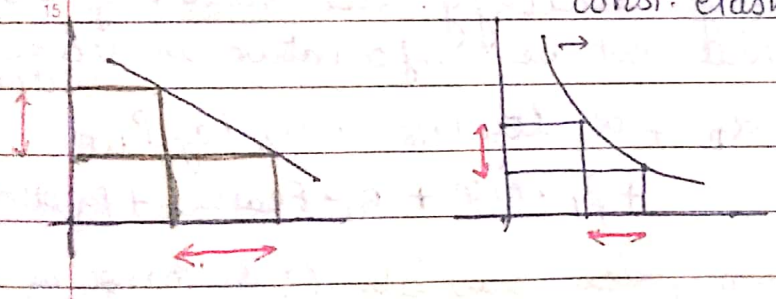


Price change in more but ~~demand~~ change in demand is less..

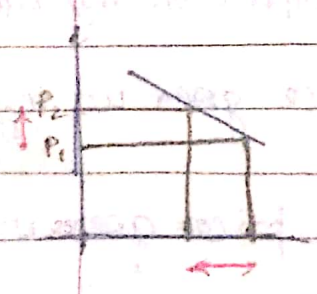
E.g. Apple.

$e = 1$ (Unitary elastic)

const. elasticity curve.

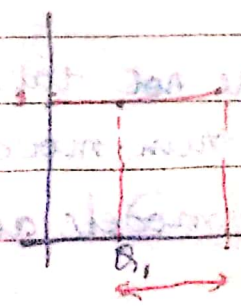


$e > 1$ (more elastic) for my competeter.



small change in price, high impact on demand

$e = \infty$



very minimal change in price while impact on quantity is extremely high.

Scanned by CamScanner

→ this will not tell us elasticity → Independent

Quantity → SALES = $\alpha_0 + \alpha_1 * Price_{(ME)} + \alpha_2 * Price_{(COM)} + \alpha_3 * Price_{(WM)}$ ...
dependent

$+ \beta_1 * DISP + \beta_2 * F$

To run above model to get const. elasticity we do log transformation

$\log(sales) = \alpha_0 + \alpha_1 \log P_1 + \alpha_2 \log P_2 + \alpha_3 * \log(WM\ price)$

* No x-formation of display and feature, They will not change.

~~So, model that we would~~ be estimating log
We are doing this transformation to recover elasticity of our demand as that will tell us ~~how~~ what should be our pricing strategy. Raw values of α & β that we will get will not be informative to decide price strategy

$\log(Q) = \alpha_0 + \alpha_1 \log P_{ME} + \alpha_2 \log P_{UT} \ldots$
$+ \beta_1 * DISP + \beta_2 * Feature + \beta_3 DISP * Feature$

α_1 = own price elasticity (tells change in my price how its impacts my demand)

α_2 = CROSS price elasticity
↳ competetor price and impact on my demand

α_1 → (−)ve because when my price goes up my demand goes down

α_2 → (+)ve when my competetor price goes up my demand goes up.

Under specified model — If we are not taking our competetor price into account and run model without it, it is under specified model and will get wrong information

Scanned by CamScanner

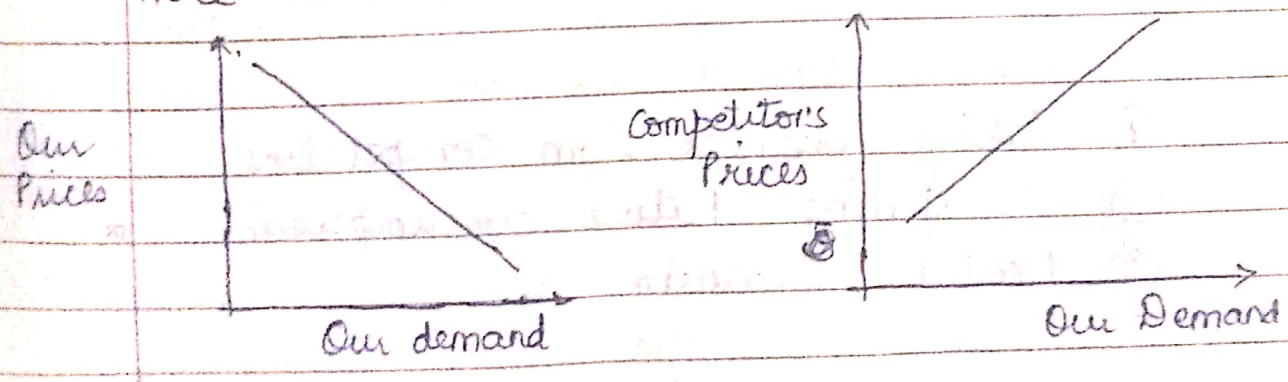
Why do we did Log transformation.

To understand if my demand is elastic or non elastic as that would decide my pricing strategy. If we run above eqⁿ without transformation, elasticity would change from price to price and therefore we will never be able to understand if our demand is elastic or inelastic

It is important to understand if our demand is elastic or not. That will determine pricing strategy. If my demand is elastic, I will try to bring my prices down. That will increase overall profit.

~~For~~ Refer Tea Party Example session 1 - 21 July, 2018 (1:48 Hrs)

Price and Demand



Prediction : Aggregate Effect

Price_TeaMe	Price_TeaU	LastPrice_Wk1	LP_Wk2	LP_Wk3
-1.27	0.36	0.27	0.14	0.11

Display	Feature	Interaction
0.00	0.00	0.14

Our Brand Promotion Effect 1.27

Cross Brand Effect .36

Stockpilling Effect 0.27 + .14 + .11 = .52

1. Impact of past prices on today's demand
If past prices were lower they have bought this product, and today they are buying less, therefore it is stock pilling effect

Consumption Expansion Effect 1.27 − .52 = .75

Increase demand come from

1. Taking business from competetors
2. Stockpiling future consumption
3. Market Expansion

* ~~It is looking~~ at your history but also it is trying to profile yourself with ~~customers that~~ I already have

* Some kind of ~~profile it creates~~ and tries to match what those people are buying, what you most likely to buy

Market Basket Analysis & Recommender Systems

Recommender Systems – tells co-occurrence – products that go together (Could be on individual/group purchase history)

Obj of Market Basket Analysis

→ put same product that are sold together in same place

→ facilating shopping process for customers (i.e. help their decision making)

→ Helps company decide ~~to keep~~ which product to keep where.

Recommender System – predicts a buyer's preference for products and recommends a list of products that the user is most likely to buy.

3 Types of recommender Systems (Engine) for building Market Basket which uses association rule

① **Knowledge - based recommendations:-**

Knowledge of buyers, products and their relationships. E.g. Physical Stores (E.g Super markets)

"Those who buy X also bought Y"

② **Content - based recommendations**

Knowledge of the buyer's historical purchase data. E.g. Movie, News items, Grocery stores

→ ③ **Collaborative Filtering models**

Knowledge of the preference of similar buyers

E.g. Online Stores (Amazon)

(Refer slide 65 | 44:37HR | ~~Day~~ 4th lecture)

Scanned by CamScanner

KNOWLEDGE BASED RECOMMENDER SYSTEMS FOR MARKET BASKET ANALYSIS

1. Most appropriate when several different products are being recommended to the same buyer.
2. Bundle of products that go together
3. Starting point is after user has made the first purchase. Meaning there has to be purchase history. There has to be data about user purchase in the past → Requirement
4. Grocery, Home Improvement, Consumer electronics
5. Value of product is not in isolation but in association with other supplimentary and complimentary product.
6. Also useful in placing related products close-by in physical stores.

What product goes with what product?
→ "Affinity Analysis"

E.g. In current basket if you have already added Salad + Olive Oil, than it is most likely we will buy "Salad Dressing" probably we are trying to make Salad. That would be knowledge based recomm.

→ Knowledge of product that go-together (fine tune it with buyer's preferences)

→ Category assoc[illegible] based [illegible] enough repeat purchases (a pre-condition)

Scanned by CamScanner

DIFFERENCE B/W CONTENT AND KNOWLEDGE BASED RECCOM

CONTENT – Are you looking for healthy (preference) or tasty salad dressing. Depending on that content based would give ~~pref~~ recommendation specific to you.

KNOWLEDGE – Primarily about what goes together. It is not about individual preference but more about baskets in the market.

Types of Association:-

1. Useful e.g. On saturdays, Hypercity customers purchase diapers and beer together
2. Trivial – E.g. Croma customers who purchase maintenance contracts also purchase large appliances.
3. Inexplicable – E.g. When a new Home Depot store opens in an area, the most commonly sold items are toilet rings.

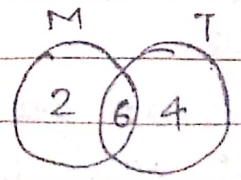
Important Measures:-

100 customers visit a store,

→ 10 bought tea

→ 8 Milk

→ 6 Tea + Milk



$$P(\text{Buy Both Tea + Milk}) = \frac{6}{100} = .06$$

$$\text{CONFIDENCE} = \frac{P(\text{Buy Both Tea + Milk})}{P(\text{Buy Milk})}$$

Conditional Probability, Given that they bought milk, what is the probability that they bought Both Milk + Tea

$$= \frac{.06}{.08}$$

$$= 0.75$$

How much we have gained by creating this basket

$$\text{LIFT} = \frac{\text{CONFIDENCE}}{P(\text{Buy Tea})} = \frac{0.75}{0.1} = 7.5$$

→ Have improved my inf. from .06 to .75 because I could make this basket. This is basic concept of basket. If we have inf. of current partially filled basket, with lot of confidence we can tell what is the next product going to be compared to an unconditional, uninformed probability.

Based on How much aggressive, I am with my confidence levels and support levels, recommendation would change.

BIG BASKET CASE STUDY

BUSINESS CHALLENGE -

→ To get products that sell together.

BENEFITS

* To ensure you buy all of them together (bundled them together)
* Most often bought product | low value item, Keep it in front of store (To build buying momentum)
* Keep products that they most likely to buy at the end of store so that they go there.

Scanned by CamScanner

LOSS LEADER

Use a few product to draw people into store, build store traffic, than create these recommender system to make their basket bigger and make profit from that customer

Steps | Process

1) Identify products that are popular among people
2) Use them in your insert as loss leader
3) Bring them into store
4) Create basket

CHURN

(Deterministic model) - Seeing off customers

Survival analysis - Looking at future in probablistic manner.

Losing Customers - Reasons

68% - Customers believes you don't care about them
14% - Customer is dissatified with your service.
9% - Customer persuaded to go to competitor
6% - Customer gets friend to provide service
3% - Customer moves away.
1% - Customer dies

Disadv. of Customer Churn

1. Acquiring new customer is costly.
2. Loss of business - dissatified customers spread bad words
3. Damage to brand Equity.
4. Lower market share
5. Opportunity for competitors.

Scanned by CamScanner

6. Negative word of mouth
7. Loss of reference customers in the segment
8. Demoralising for staff

① Analytics helps us understand why customers leave so that mitigating strategy could be found.
② Predicting which customers are likely to leave so that company could have a proactive strategy to retain.

As a business decision, you also need to think whether customer has any other option to go or not (eg railways)

Customer who is complaining is ~~giving~~ expecting company will do something and ? do not have to look for alternative. Therefore, complaining customer is giving opportunity compared to customer who is non complaining. Those customer if they do not like something, go and buy something alternative.

* It cost 5 times more to buy a new customer than retaining old cust.

PREDICTING CHURN

Basic model for churn - logistic regression and not linear regression because :-
① Outcome is 0 or 1 (Yes/No)
② If outcome is (-)ve you do not know how to interpret.

MARUTI MARKETING CHALLENGE

- Maruti wants to predict churn using its customer database for Swift Dzire
- Churn = f (age, income, purpose)

Hypotheses :-
- If customer turns 45, he may churn
- If customer makes > Rs 20L, he may churn
- If the car is for customer's own use (vs. as a cab) then he may churn!

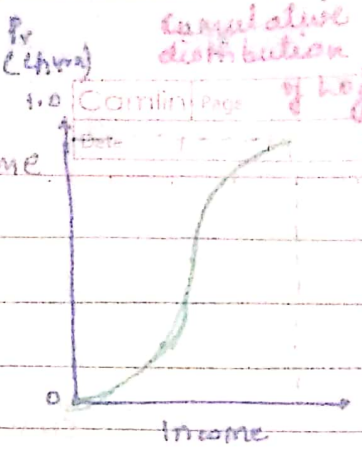
Scanned by CamScanner

satisfaction with product

$V_1 = \alpha_0 + \alpha_1 * Age + \alpha_2 * Income$

Probability of buying product

$$P(1) = \frac{e^{V}}{\sum e^{V'}}$$



Cumulative distribution fund of Logit

Telecom Example

Business Obje – Why some people stay, why some move away

LOGISTIC Vs LINEAR REGRESSION

1. LINEAR REGRESSION
   - Churn = $b_0 + b_1 * Income$
   - This will predict churn to take values <0 and >1
2. Logit
   - $V = b_0 + b_1 * age$
   - $Pr(Churn = "Yes") = e^{V} / 1+e^{V}$
   - $Pr(Churn = "No") = 1 / 1+e^{V}$
   - Churn = 1 if $Pr(Churn = "Yes") \geq 0.5$, else = 0
   - This allows churn to take either of two values 0 or 1

Telecom ex. continued

In a more realistic situation where we do not know whether customer has stopped buying or does not need this product, that we will do in survival ~~model~~ analysis which is probablistic model

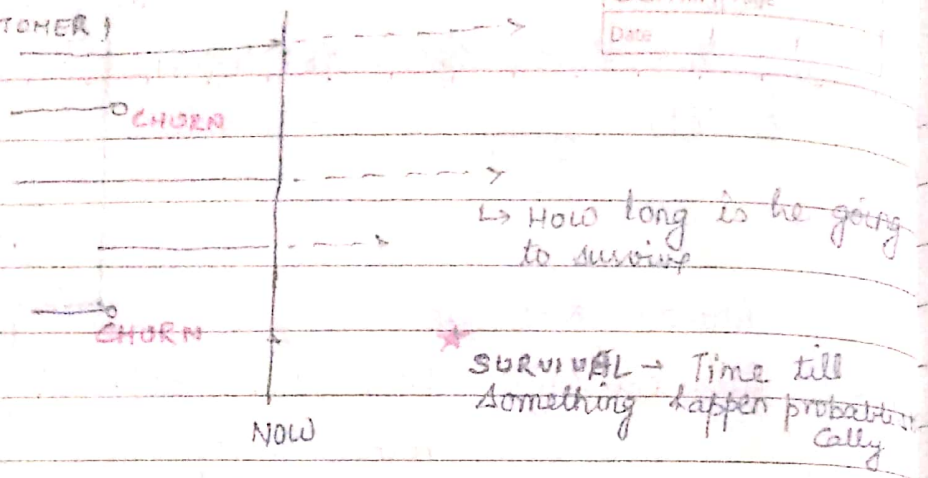
★ ★ Refer R code for telecom example

Churn – is within a pre-defined time period – (Likelihood of leaving)

Survival – is churn over a period of time – if given what has happened, what is likely to happen. (Likelihood of staying)

Scanned by CamScanner

CHURN Vs SURVIVAL



On the basis of Churn data on LHS, I could say how long you will survive.

In a non contractual setup, where we can only guess if you are my customer or not, trying to understand after 6 months how many people will be there or 1 year in a probabilistic manner. This is SURVIVAL MODEL i.e. If you have survive till now, what is the probability you will survive after 6 months/ 1 year. That is the more realistic model.

For some customers we have data that they've left. Using this inf. we're trying to ① figure out what happens in future for others customers.

Surviving — ① as a customer (How long you'll stay as customer)
— ② as a non-customer and than become customer. (How long you'll stay as Non customer)

① + ② → Both are survival model

BUSINESS CHALLENGE

Refer recording. Example: To figure out probability of conversion for term deposit, for someone who is already customer and bought other products like loan, FD etc. How to increase probability of selling term deposit

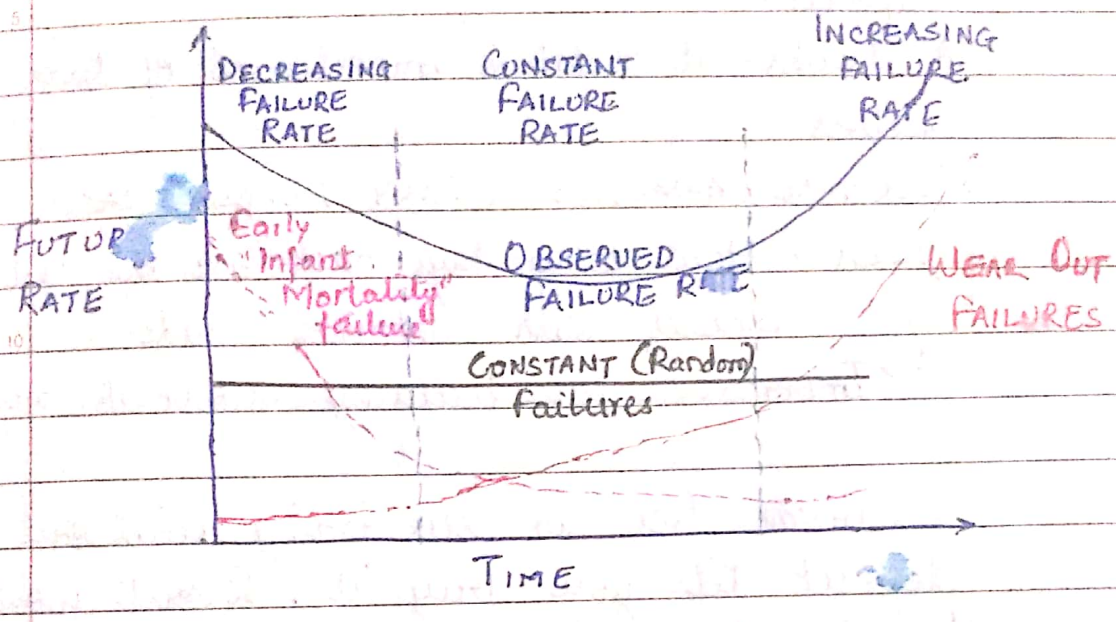
	CHURN	SURVIVAL (more into realistic)
①	Deterministic model	Probabilistic model.
②	Within pre-defined period	Churn over a period of time
③	About understanding why people leave	About extending relationship [After doing survival analysis done, we could have insight how to improve customer rela]
④	Why do they go	How much longer will they st
⑤	Who leaves (definitive)	Why some stay longer than other

as non cu (uncertain

Hazard Model.

Hard Function $h(t)$ – Risk of losing a customer in future (after t) given that customer has survived till 't'

→ Also refer proportional hazard model.



Above Churn rate / failure rate which is reducing with time, increasing with time There is various different ways it can be modelled.

Look at Proportional Hazard model

as non customer / customer
(Uncertainity)

Customer Lifetime Value (CLV)

KAPLAN MEIER CURVE

Advantages

* Useful for comparison specifically for categorical data.

~~Objective~~

* looks at relative importance of two factors

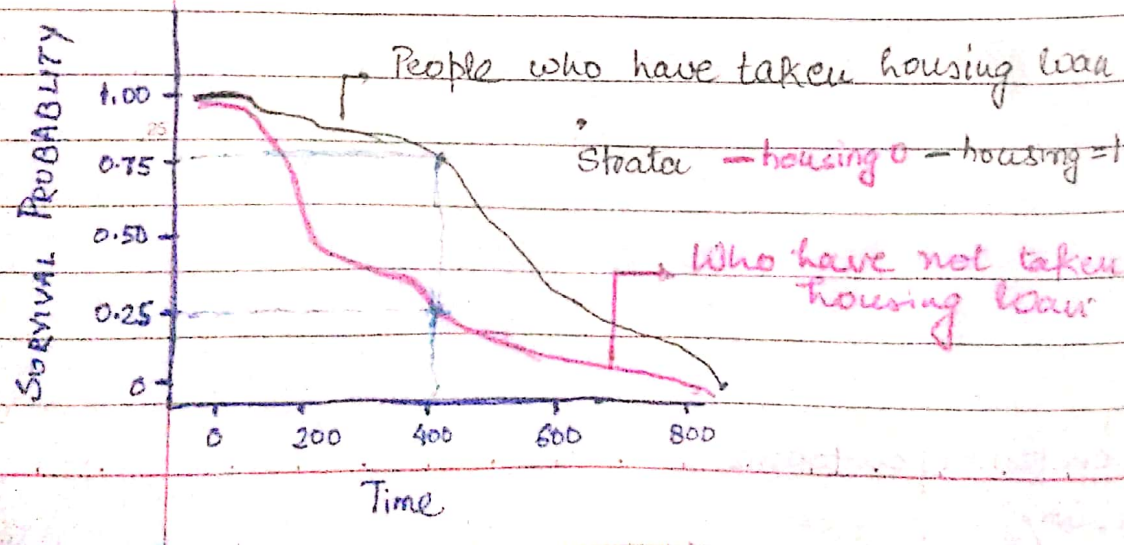
→ SURVIVAL ANALYSIS (TERM DEPOSIT CASE

pdays – How many days back was the last contact with customer made

↳ Important for building survival model.

Survival here is surviving without term deposit till you buy it. Event would be buying term deposit

If we model this as logistic regression, that will be churn problem. But we are not doing this here. Given this inf, what do I do to increase effectiveness of my marketing strategy. This is one step ahead of churn analysis



Scanned by CamScanner

P Packages - Survival, Surv Mdc, Survmcurv

obj - cross sell of term deposit.

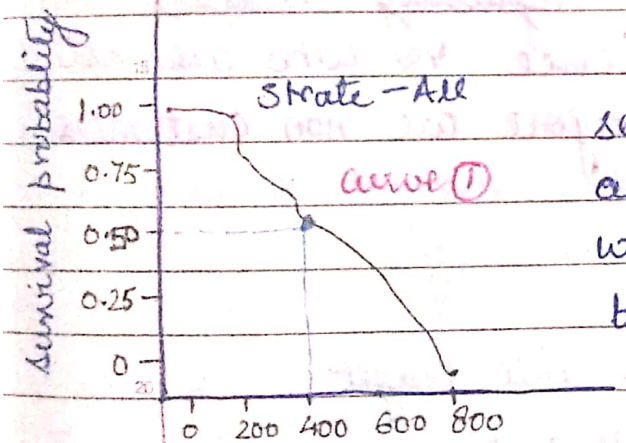
~~After 400~~

** In context of term deposit, ~~surviv~~ those who have not bought is the survival probablity.

~~After~~ Red Curve - People who has not bought HL 75% of people has bought and 25% has not bought term deposit after 400 weeks

Therefore, red curve is the more favourable condition in this case.

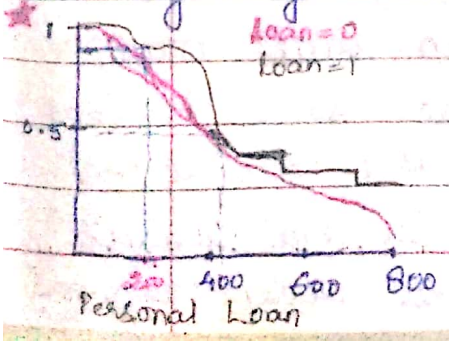
As business challenge is people who has not bought and favourable condition is they are buying term deposit.



If we do not do any kind of segarration, after 800 days about 90% of people would buy and we are trying to improve conversion.

Kaplan Meier Curve - Building a Strategy

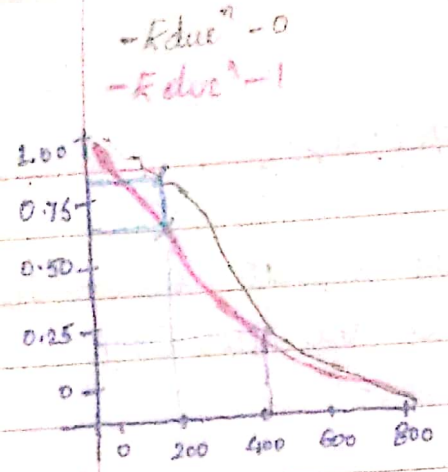
★ Those who have not taken housing loan are more likely to buy term deposit and therefore they should be my target compare to those who has taken HL



Personal loan Vs Not taken personal loan

data is same here / story is same.

Scanned by CamScanner



People with tertiary education are slightly better. There is not much separation b/w 2 curves but at early stage there is some separation. Red curve is always lower. Reasons could be higher education means more income

* Therefore, if we try to sell to a group those who has not taken housing loan, we will increase probablity in 400 days to about 75%.

IMP - By using curve ①, we can increase from 50 to 75% by segrating market based on target audience vs who are not likely to buy and therefore are non customers.
That's how or this analyses helps and stratize marketing strategy.

Marketing Strategy. we will target
① People who has not taken HL.
② Educated people
③ Who don't have personal loan.
④ Calling them will increase conversion hugely.

Therefore, if data is richer, if we have many such columns we can draw many such curves or we can use some Hazard model to build this model and look at best way to increase conversion. Survival analysis looks at future scenario gives our current exp. and

Tell us how we can increase our marketing strategy.

INFLOW - OUTFLOW

CUSTOMER LIFETIME VALUE (CLV)

Discounted cash flow, Activity based costing - These are extremely relevant to get good sense of ~~customer~~ value of customer.

	Year 0	Year 1	Year 2	Year 3	Year 4	Year 5
No of Cust	100	100	100	100	100	100
Revenue / Cust		100	110	121	133	146
Var Cost / Cust		70	74	77	81	85
Margin / Cust		30	37	44	52	61
Acquisition / Cust	-40					
Profit	-4000	3000	3,650	4383	5207	6132
Present Value (in Yr. 0)	-4000	2727	3,017	3293	3556	3808
NPV (in Rs)	12,400					
Value / cust	124.0					

Margin = Revenue - Variable Cost

$$NPV = \sum_{t=0}^{5} \frac{\text{Present Value at time } t}{(1 + \text{Discounting Rate})^t}$$

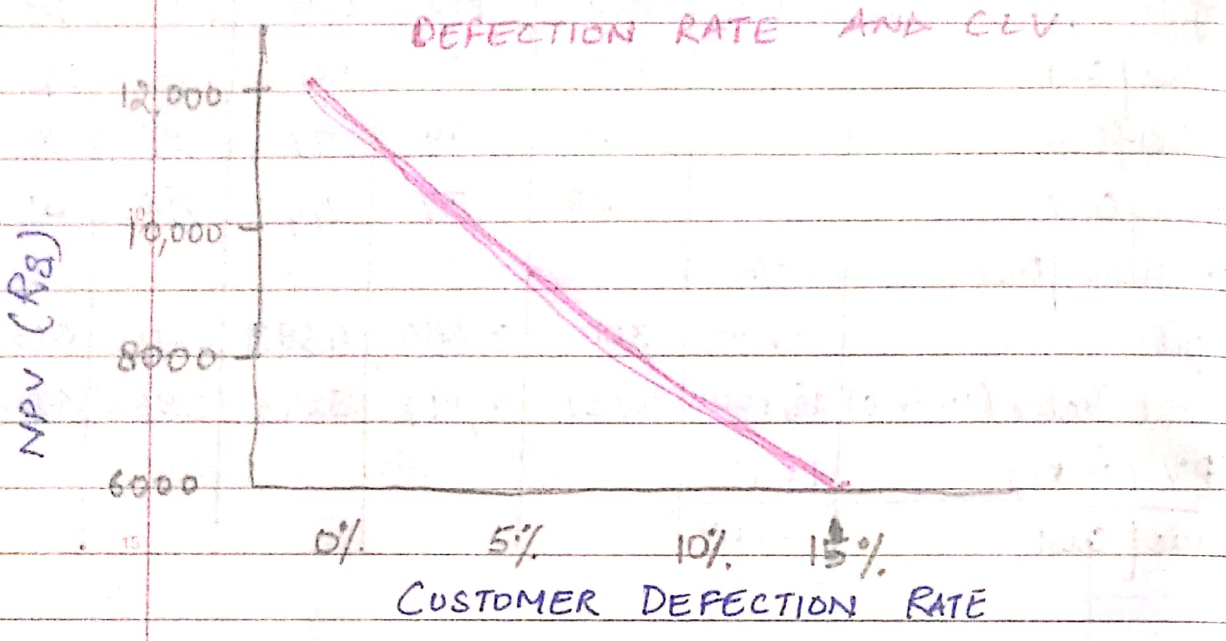
d = Discounting Rate = 10% CHURN - IMPACT ON CLV

No. of cust	100	90	81	73	66	59
Rev / Cust		100	110	121	133	146
Var Cost / Cust		70	74	77	81	85
Margin / Cust		30	37	44	52	61
Acquisition C / Cust.	-40					
Profit	-4,000	2700	2957	3195	3416	3621
Present Val (in Yr. 0)	-4,000	2455	2443	2400	2333	2240
NPV	7 880					
Value Per Custom	78.80					

Scanned by CamScanner

DEFECTION RATE – Percent of customers who leave the business

* Value per customer goes down from 124 to 78 when customer are leaving @ 10% per year and revenue increasing @ 10%.



Above CLV comes down sharply from 12000 to 6000
Impact on business is huge if 15% of customer leaves

CLV – a metric that represents the total net profit a company makes from any given customer. It is a projection to estimate customer worth to a business after factoring in the value of the relationship with a customer over time. CLV is an important metric for determining how much money a company wants to spend on acquiring new customers and how much repeat business a company can expect from certain customers.

Scanned by CamScanner

ACTIVITY BASED COSTING

→ Looks primarily at Indirect cost.
→ Consultant for short time, sales persons, service engg, resources that are shared across customers are not assigned proportionately to particular customer based on how much they consume.

E.g. if HR is providing services to 100 accounts does his cost is divided based on the efforts he is giving to individual account.

ABC tries to fix problem of fixed costing As there is no rigsous accounting done for fixed cost (VP cost, electricity bill, AC cost).

Unless we do not know cost, we will not know value. Else we have notional cost & notional value.

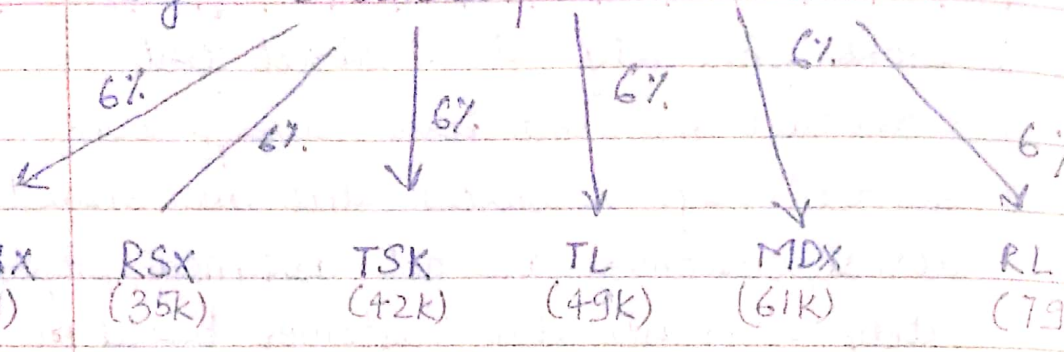
Value of customer (or segment) is driven by
1. Revenue earned from them.
2. Cost to serve them.

* It is important to not only earn revenue but also attribute revelant cost correctly.

Activity-based costing (ABC) is a costing methodology that identifies activities in an organization and assigns the "cost" of each activity with resources to all products & services according to actual consumption by each.

Scanned by CamScanner

Target: Overhead / Indirect Cost



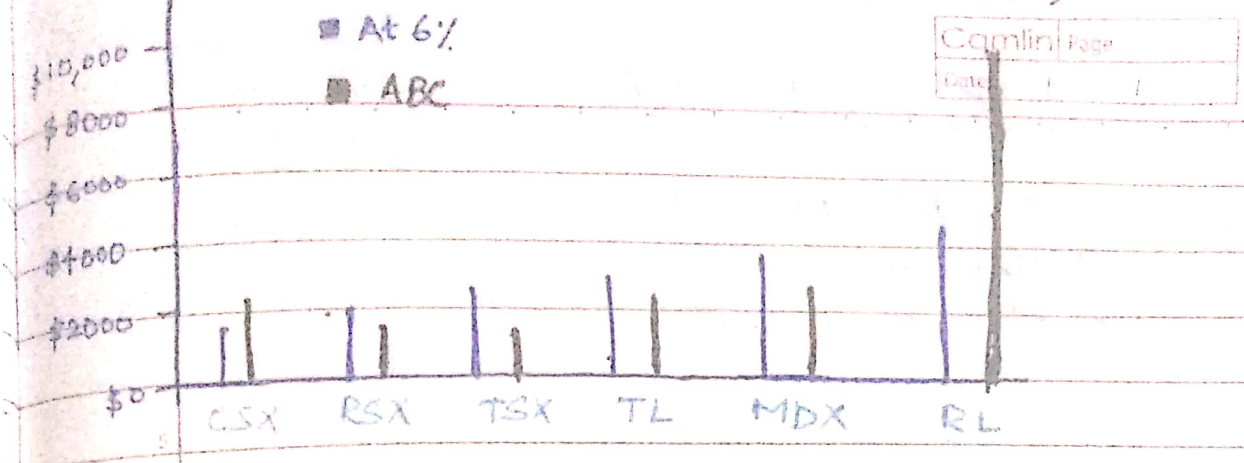
Indirect cost 6% is loaded equally across 6 variant. It is very difficult to understand how much each model is consuming in terms of sales persons effort and number of sales person trying to sell each car. Therefore, it is challenge to understand value of each model. Therefore, here we are trying to do valuation of each model.

* RL is costliest, therefore requires most resources to sell.

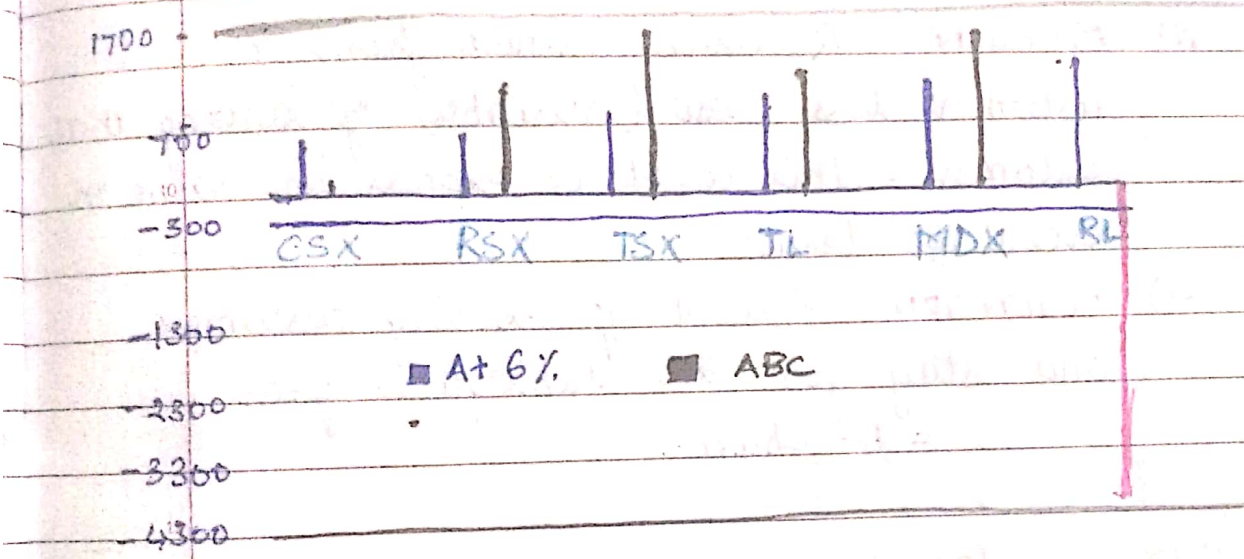
Car	CSX	RSX	TSX	TL	MDX	RL
Total Sales	79	59	114	212	148	35
Cost / Car	29500	32500	39800	46000	57200	73900
Indirect Cost / Car (6%)	1770	1950	2388	2760	3432	4434
Margin / Car (2%)	590	650	796	920	1144	1478
Price / Car	31860	35100	42984	49680	61776	79812
Total Indirect Cost/model	139830	$1,15,050	2,72,232	585120	507936	155190
Total Indirect Cost	$17,75,358					
Share of Indirect Cost/model	8%	6%	15%	33%	29%	9%
ABC share of Indirect Cost/model	10%	5%	10%	30%	25%	20%
ABC Indir Cost/Car	2247	1505	1557	2512	2999	10145
ABC Margin/Car	113	1095	1627	1168	1577	-4233

Scanned by CamScanner

INDIRECT COST PER CAR (-$)



MARGIN PER CAR ($)



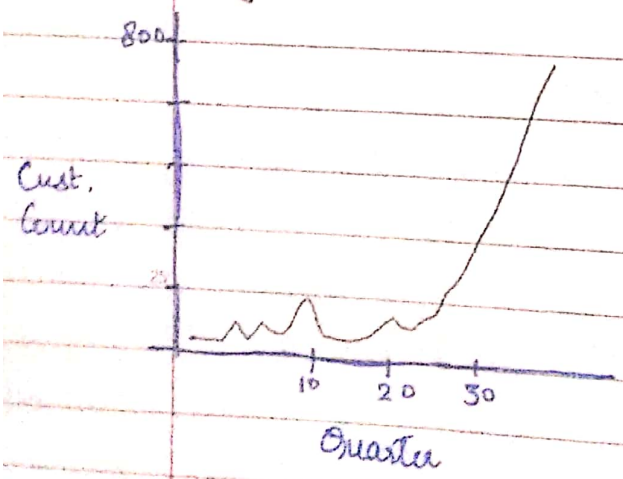
Scanned by CamScanner

DRIVERS OF CLV

1) ACQUISITION COST
2) MARGIN
3) RETENTION RATE

1) ACQUISITION COST – Money spent in acquiring a new customer
2) MARGIN – Revenue earned from a customer less cost (variable) of serving that customer. This could be express in value or percentage terms.
3) RETENTION – Percent of existing customers who stay for the subsequent year.
= 1 – Churn.

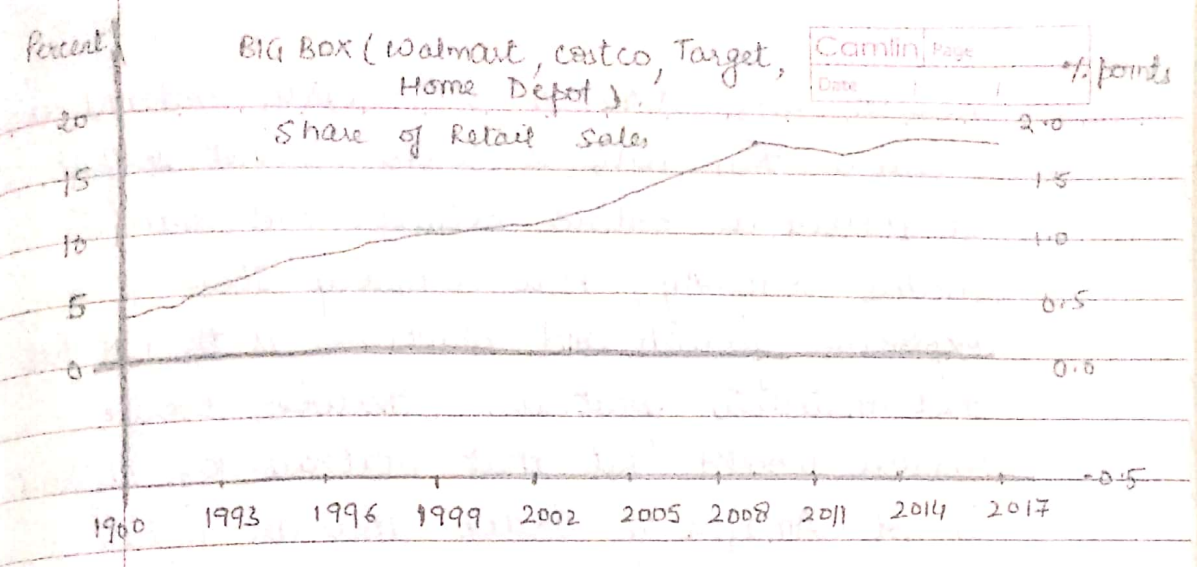
Example – Biography of start-up.
Cust aquisition cost – Rs 10/customer.
Margin – Rs 30/unit.
Customer Retention – 80%.
Objective – Evaluation of Comp.



Business can't grow beyond what environment will be able to support. There is upper limit for every business. After sometime there will be saturation, alternative in the market.

Scanned by CamScanner

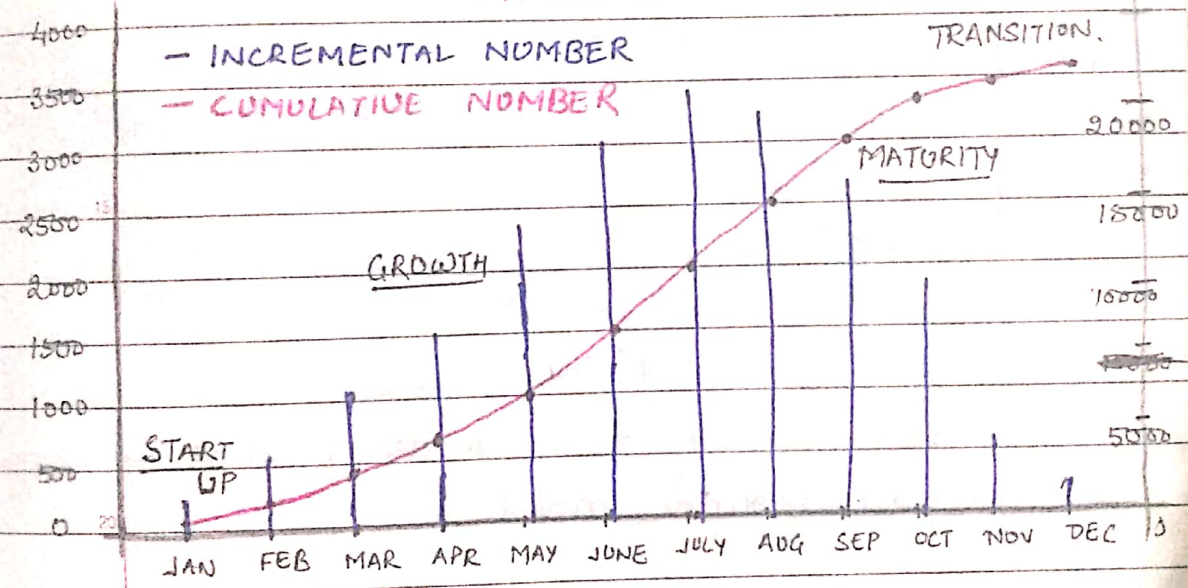
US BIG Box Retail Sales



Business goes through 4 phases

LOGISTIC GROWTH MODEL – is the most realistic representation of business in long term

TYPICAL S CURVE



Logistic Growth model takes into account that companies will grow slow at early stage, growth rate will explode after while and than it will slow down.

$$N_t = \frac{K * N_0 \times e^{r*t}}{K + N_0 * (e^{r*t} - 1)}$$

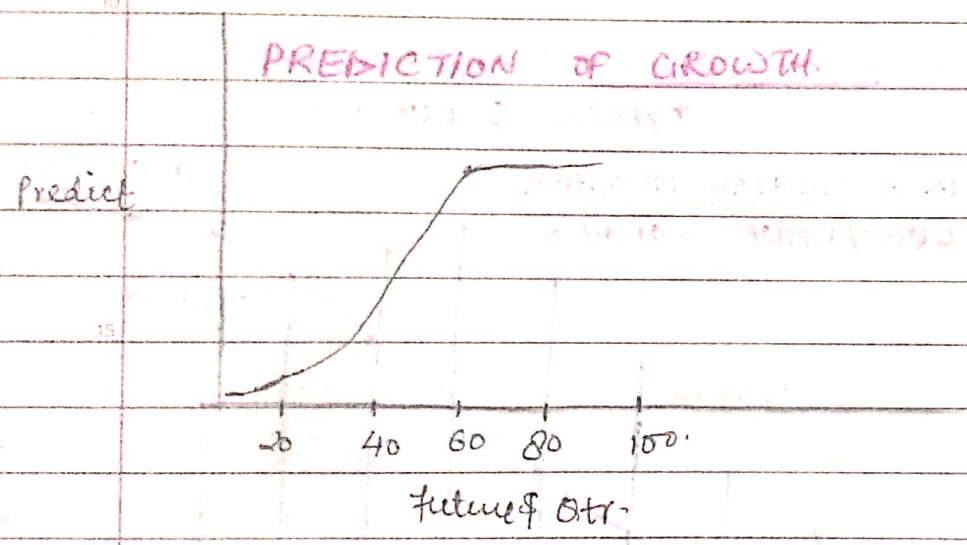
N_t - Cumulative no. of customers at time t

K – Capacity → need to be estimated

r – Growth rate

K, No, r → Need to be estimated

Reason why Amazon so highly valued is because they redefine their market so fast. It started as online retailer but now its media company, H/w company. There is explosive growth and objective is to not hit that maturity plateau. Otherwise, every business would hit that plateau. E.g Flipkart is not Amazon of India they are online retailer. They are not into anything else like Amazon from last 15-20 yrs.



Above is long term model of start-up using logistic growth model.

Refer ~~video~~ 5th video (47:31 Hrs) slide 19.

Scanned by CamScanner

DCF → What are they worth NOW. Because we are evaluating comp. today.

WHAT-IF EXPERIMENT – FACTORS

- Acquisition Cost.
- Margin
- Retention Rate
- NPV
- Geometric Progression.

Assumptions. → everything is same across customers e.g. margin/cost etc.

Below example we are talking about cust. which came in year 1.

Quarter (0)	1	2	3
Margin from Customer	$N_1 * (m-c)$	$N_1 * (m-c) r$	$N_1 * (m-c) r^2$

N_1 → No. of Cust. in Year 1

m → Margin from Custome

c → cost

$(m-c)$ → Getting from Cust. in year 1

r → No of cust. retained in year 2.

$(1-r)$ → left

DCF →

$\frac{N_1 * (m-c)}{(1+d)}$	$\frac{N_1 * (m-c) r}{(1+d)^2}$	$\frac{N_1 * (m-c) r^2}{(1+d)^3}$

Cost of Aquisition Customer. ← $N_1 * a$

Discount Cost of Acquisition $\frac{N_1 * a}{1+d}$

Customers that get added in yr. 2

$\frac{N_2 * (m-c)}{(1+d)^2}$	$\frac{N_2 * (m-c) r}{(1+d)^3}$

builds Geometric progression

$\frac{N_2 * a}{(1+d)^2}$ → Discount rate of ~~cost of acquisition~~ Customer who got added in year 2.

Profit = Margin – Cost.

$$NPV(Q1) = \underbrace{\frac{N_1 * (m-c)}{1+d}}_{PV(Earnings)} - \underbrace{\frac{N_1 * a}{1+d}}_{PV(cost)}$$

NPV(Q2) =

Scanned by CamScanner

After around 78 quarters no new customers are getting added and business is losing out because of Churn.

Valuation of Enterprise = $NPV_{a_1} + NPV_{a_2} + \dots NPV_{a_n}$

This would give us approximate life span of comp

Cust Acq Cost – Rs 10 / customer
Margin – Rs 30 / Customer
Retention – 80% / 100
Discount 12% per annum.

ELASTICITY ESTIMATES

What-If (Numbers)			CHANGE IN NPV DUE TO	What-If (%	
Decr 1%	Steady	Incr (1%)		DECR (1%)	INC
48,405.67	48330.72	48,256.36	CUST Acq. Cost	0.15%	-0.1
47773.06	48330.72	48,888.38	Margin	-1.15%	1.15
46426.51	48330.72	50,332.58	Retention	-3.94%	4.14

"Retention has highest impact on any enterprise value compare to to any another like margin / acq. cost. 1% reduc^n in retention will have ~~4~~ reduce approx 4% reduction valuati of firm.

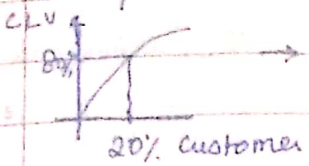
Cohort analysis is a subset of behavioral analytics that take the data from a given set (eg. e-commerce) and then looking at all users as one unit, it breaks them into related grp ~~of~~ for analysis. These related grps (or cohorts) share common characteristics within a defined time span. Eg – students graduated in 2010 forms a cohort and any study on them (say avg income after 5 of graduation is called cohort analysis)

Scanned by CamScanner

A forward-looking probabilistic model by Schmittlein (1987)

Future is uncertain and every customer is unique : PARETO / NBD model for CLV

For particular distribution not for all distribution

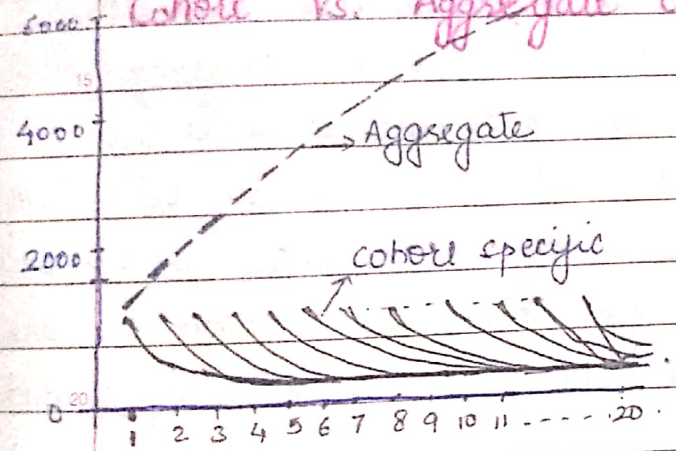


→ At 20% of customers, you will get 80% of life time value. 80-20 works

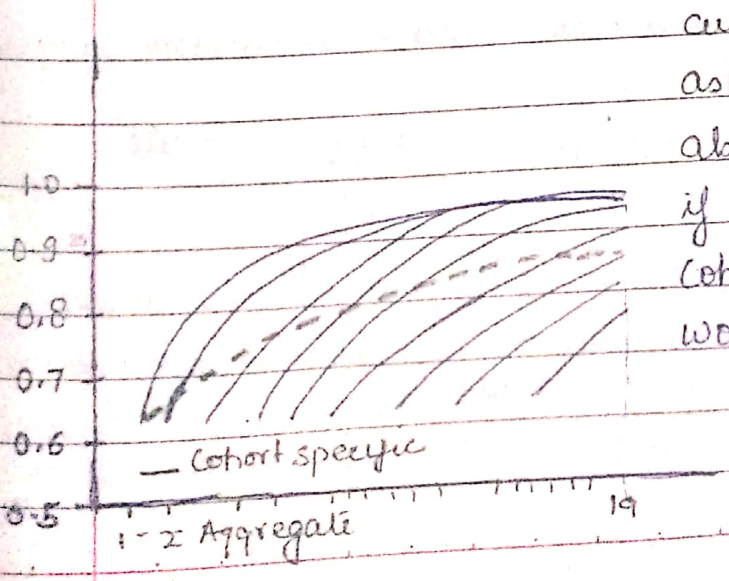
Pareto / NBD proposed ~~this~~ was proposed because of the assumptions made till now which are unrealistic.

1. Every customer has equal value.
2. Past is a good indicator of future
3. Behaviour remains unchanged over time.

Cohort vs. Aggregate Customers:



Customer aggregate no are increasing but looking at each cohort, number of customers will keep falling



Customer stickiness ↑ as time goes by. This is also rising curve but if we look at each cohort characteristic would be different.

Scanned by CamScanner

Example Data - CLV-PNBD.xlsx

We could do similar calⁿ to calculate value of cust. as in What-If experiment but it would not be correct because every customer is giving diff. business and it is changing over period.

This is refinement over RFM analysis
Recency & Frequency tells us stickiness of customer, how long they are going to survive
M - tells us value of each customer.

Pareto / NBD Model in non-contractual setting

Telecom business is contractual business where we have at least month long contract. Here this example of online store is non-contractual and is most realistic situation as we do not know if customer is surviving or churned away. Therefore, we have to make realistic ~~model~~ estimation of it using a model

1. Individual purchases follow a Poisson process with rate λ

$$p(x \mid \lambda, \Delta t) = \frac{(\lambda \Delta t)^x}{x!} e^{-\lambda \Delta t}$$

Probablity of x no. of occurances (transactions/purchases) for poisson process which has rate λ within time period Δt is $\frac{(\lambda \Delta t)^x}{x!} e^{-\lambda \Delta t}$ ↳ represents characteristic of poisson process.

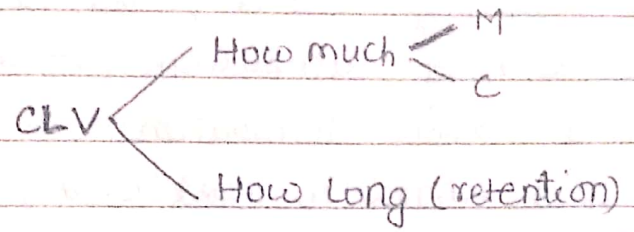
Scanned by CamScanner

Therefore, all these occurences that we see in purchases are following unknown poisson process with unknown λ. λ is recovered from data.

2. Heterogeneity : λ follows a Gamma distribution

$$\lambda \sim \Gamma(r, \alpha)$$

3. If we are trying to create CLV, there are two major factors that contribute to CLV.



We are trying to model these two. First one is "How much" which is Poisson process as it is number | count. Another that we have to model is "How Long" which is probabity of any customer ~~which~~ is a 'Pareto function with mean death rate μ. If mean death rate is μ, what is the probablity that particular customer will survive for time period τ

3. Probability of survival of any customer is a Pareto function death rate μ

$$p(\tau | \mu) = \mu e^{-\mu \tau}$$

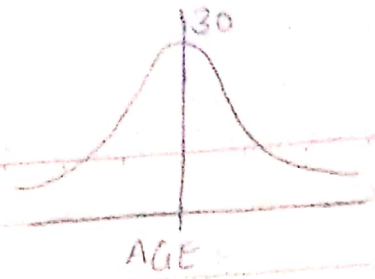
4. Heterogeneity : μ follows a Gamma distribution,

$$\mu \sim \Gamma(s, \beta)$$

5. λ and μ are independent from each other.

Scanned by CamScanner

Avg. age of class is 30



~~Practo~~ Here not everyone is of same age and this is known as Hetrogeneity.
~~Thus~~ distribution here is done by Gamma

Gamma Distribution: (Reasons of using it)
- It is extremely flexible function/distribution. Depending on value of K & θ, we can get any any kind of distribution (normal, exponential, etc) using Gamma distribution.
- It is easier to work with, calculate in comparison to other distributions.

Closed form solution - when distribution has a closed form solution, we can estimate it easier otherwise we have to simulate to estimate parameter which is complex.

$\Gamma(K, \theta)$ K: Shape parameter
θ: Scale ,,

When we estimate this model, we will get r, α, s, β.

Scanned by CamScanner

* train (calibration) and test (holdout) *
calibration and hold out model

Pareto-NBD Likelihood func^n

$$L(\lambda,\mu \mid x, t_x, T) = \frac{\lambda^x \mu}{\lambda+\mu} e^{-(\lambda+\mu)t_x} + \frac{\lambda^{x+1}}{\lambda+\mu} e^{-(\lambda+\mu)T}$$

T : Training period
x : No. of purchases during the training period
t_x : Recency (time)

R Package : BYTD

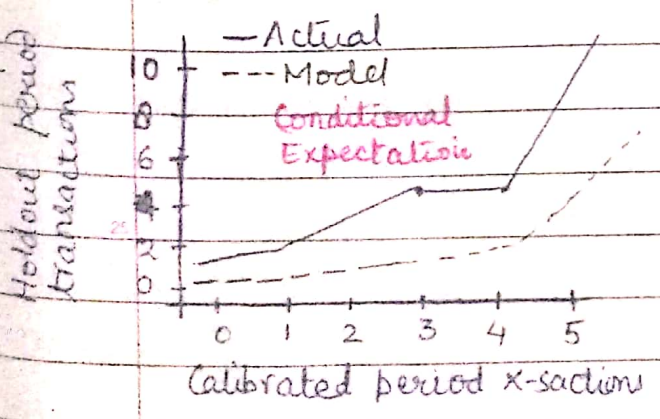
We will estimate this model, given the hetrogenity that was defined.

```
params <- pnbd.EstimateParameters(Cal.cbs)
```
↳ Estimates parameters for Pareto NBD

Using these parameters, used it to estimate the average number of transactions.
Given discounting rate of 10%, got ~~con~~ expected lifetime value (CLV) of each customer [Ex. done in class]

Model Evaluation & Expected CLV

Online Store. (Refer 5 video slides 32/33)
Ex: Data_CLV_PNBD.



* Model prediction compared to holdout sample, conditional on observations during the calibration period
* Model slightly underestimating numbers

exp_transactions → 5.46 (a new customer is expected to make 5.47 purchases in one year)
Cust_13047 → 57 (Cust. no 13047 is expected to make 57.6 purchase in a year)

Scanned by CamScanner

Data to Insight to Marketing Action.

1. Identify existing high-value customers
2. Identify high-value customers who have churned
3. Identify high-value customers who are likely to churn
4. Conduct policy experiments.
   - Two similar groups of gamblers from the same town

★ Peter Fader (Wharton) and Bruce Hardie (LBS)
http://www.brucehardie.com/

Implementing BG/NBD in Excel
http://www.brucehardie.com/notes/004/

Scanned by CamScanner

Indirect Benefits: CLV in the networked World

① Word of mouth (WoM)

Pareto/NBD is also underestimating value of customer because customer bring in customers. Therefore, what is the impact of WOM where we all are connected to each other.

WOM in the context of CLV: CRV (Customer referral Value)

CLV – is the business that I get from cust. directly

CRV – is the value I get because they refer other customer to me.

If we add CRV to CLV, it becomes much bigger. Therefore, when we are doing conventional CLV calculation, still not looking into CRV. It is difficult to get that data but it is important to incorporate this data into CLV calculation.

Referral Process.	Financial Services	Telecom Services
① Willing to refer Company to other people	68%	81%
② Actually referred Co. to other people	33%	30%
③ Actually generated business.	14%	12%
④ Profitable new Customers	11%	8%

CRV is effective and less expensive.

Calculating CRV.

CRV — CLV_1, CLV_2, CLV_3

CLV of a typical Custom $1144

CRV of a typical Cust. $257

$2+5

$1049

CRV is much higher than CLV

$CRV = CLV_1 + CLV_2 + \ldots + CLV_n$

CLV and CRV: Different values from diff. groups

Average CLV after 1 year	Average CRV after 1 year: Low	Average CRV after 1 year: High
High	**AFFluents** 29% of customers CLV = $1219 CRV = $49	**CHAMPIONS** 21% of customers CLV = $370 CRV = $590
Low	**MISERS** 21% of customers CLV = $130 CRV = $64	**ADVOCATES** 29% of customers CLV = $180 CRV = $670

Scanned by CamScanner

Ensemble, Integration & Conclusion.

Agenda -

1. Using tools in combination - Ensemble.
2. Data and Intuition.
3. Ensemble using simulation and advanced computing.

Ensemble

~~of Ensemble approach~~

Improve prediction by:

1. Combining different methods.
2. Combining diff. data.
3. Combining diff. approaches.

~~Example~~

Netflex gave challenge of creating recommendation engine. ~~and~~ There were 3 groups working independently and later found each one working independently and bldg. separate model and giving out to Netflix. Rather if they use their model together, recommendation performance was much better. Therefore, instead of 3 entries gave single recommendation and won the event.

How to combine

By taking simple average

By taking weighted average

(with higher weight to better performer)

Scanned by CamScanner

Applying Ensemble approach to "Left Hanging case"

* Objective is to get a better predictive model than previous by bldg. multiple model and than combining to get better results".

Ensemble : Approach 1

Full Dataset → Split-1 Estimate the Parameters

Full Dataset → Split-2 Estimate the Parameters

Full Dataset → Split-3 Combine and test the predictive power

① Logit 1 = $\alpha_1, \beta_1, \gamma_1$

Data ② Log 2 = $\alpha_2, \beta_2, \gamma^2$

③ Test ← Weighted Avg.

Scanned by CamScanner

python.exe location programmatically
import sys
print (sys.executable) → 3.x

Python 1.0 - Jan 1994
2.0 - Oct 2000
3.0 - December 2010

Python / General purpose language
It can be implemented in:
1. Machine learning
2. GUI
3. s/w development
4. Web development

Python is backward incompatible. Python 2 is legacy but because of difficulty in migration, companies still using it

Python is
1. Interpreted → open source - has many additional open source libraries
2. Object oriented
3. High-level.

Windows command line

cmd → cd (current directory)

dir (reports all contents of directory)
cd Desktop (to change directory)
cd.. to move up a directory
cls close screen

Anaconda - It is a distribution which includes python as well as many other useful libraries, including ~~libraries~~ jupyter (which is a notebook system)
Miniconda - smaller version of anaconda, if space constraints (fewer libraries)

→ Download and install anaconda
→ Do not add anaconda to the path variables (environment variable). But trainer suggested to add, as it allows to access jupyter notebook and other objects directly from command line.

Scanned by CamScanner

→ Anaconda Navigator (Desktop app)

→ Python code can now be run as a script on command line or through jupyter notebook

→ 3 development env for python

① Text editors (eg. Sublime text & Atom)

② Full IDEs (eg. Pycharm & Spider)

③ Notebook environment (Jupyter)

① General text editor for any text file. Not designed specifically for python. Can be customized with plugins or addons.

② Specifically used in org working on python use cases and with large files. Designed specifically for python, lots of extra functionality.

③ See input and output next to each other. Supports in-line markdown notes (normal text), visualization, videos and more. Special file formats that are not .py

You split the code into different cells and run individually

In text editors you create a big program and run that script on command line

- .py scripts can open in any text editor.
- .ipy extension is notebook extension and opens the file as notebook file.

Notebook env is very popular in the fields of data science and machine learning as it allows to see visualization and code input right next to output.

9990011375

Sublime Text Editor

Sublime Editor - unlimited free trial

Create script and save as .py file.

Jupyter Notebook

Cell wise code

Git and Github

Git is free 'open source' version control system

Github is a webpage to publish your git repository and collaborate with other people

Git ~~hub~~ can be used locally without github

Python objects and data structures

Key data types

- Integer - int - 3, 300, 200 (whole num)
- Floating point - float - 2.3, 4.6, 100.0 (decimal)
- strings - str - "hello", "* $" (sequence of chars)
- lists - list - [10, "hello", 200.3] (ordered sequence of objects)
- Dictionaries - dict - Unordered key : value pairs ("mykey" : "value", "name" : "Frankie")
- Tuples - tup → Ordered immutable sequence of objects (10, "hello", 200.3)
- Sets - set → Unordered collections of unique objects: { "a", "b" }
- Booleans - bool → Logical value indicating True or False

data structures collection of other data types objects

In tuples you can't change ~~the order of~~ objects or any object.

Scanned by CamScanner

Numbers < Integers / Floats

arithmetic operators - +, -, *, /

mod operator - % (7%4 = 3 ← reminder)

% can be used to check even/odd as 23%2 would give 1 if odd.

Power of - 2**3 = 8

abs(100-n) gives absolute value of 100 - n

Variable names - cant start with numbers

no spaces allowed, can use '_' instead

Cant use symbols :", <>/?|()!@#$%^&*~-+

- Use lowercase naumes as best practice
- avoid using special meaning words/built in keywords like lists, etc.

- Python uses Dynamic Typing. This means you can reassign variables to different data types.
- This makes python very flexible in assigning data types, this is different than other languages that are "Statically-Typed"

E.g. my_dogs = 2 ✓

my_dogs = ["Sammy", "Frankie"] ✓

In other languages,

int my_dog = 1;

my_dog = "Sammy"; → Error as this (Sammy) is not integer

To check a data type use type()

as in R class(x)

Pros of Dynamic Typing:
- Very easy to work with
- Faster development time.

Cons of Dynamic Typing:-
- May results in bugs for unexpected data types
- You need to be aware of type ().

if a=10
type (a) gives int

Built in data type are highligeted. If you see that your variable name highlighted than normal value avoid using this.
To restart session → select restart Kernel.
It would delete all temp/env variables. You would need to restart session if in case you have assinged some variable to built in data type (which is incorrect) and ran that piece of code or made some other re-assignment mistake.

Strings — single/ double quotes
"I don't do that" — as a single ' is there in the text, so include whole string in " " to avoid issues
Ordered sequences — strings are ordered sequences, as we can use indexing and slicing to grab sub sections of the string. Indexing allows to grab even single characters from ~~word~~ string using index position

String	H	e	l	l	o
Index position	0	1	2	3	4
Reverse indexing	0	-4	-3	-2	-1

Scanned by CamScanner

Slicing - to get sub section of string
→ [Start : stop : step]

Escape sequence print ("Hello \n World")
Output Hello World → Python automatically considers \n as escape sequence as soon as it encounters \

\n, \t required

length function - len()

Indexing & slicing

mystring = " Hello World"
mystring [1] = 'e'
mystring [-3] = 'r'
mystring [2:] = 'llo world'
mystring [:3] = 'Hel' * last index is not included.
mystring [::2] = Hlo Wrd
'Hello World'[8] returns r

name = 'Siddhartha'
name [::-1]
htrahddiS

s = "I am genius"
' '.join(text.split()[::-1])
genius am I

Immutability

Strings are immutable as you cant reassign /change characters in string (Cant use indexing to change individual element of string)

name = "Sam"
name[0] = 'p' will give error

So use concatenation logic to add p+"last letters"
'p' + last_letters where last_letters = name[1:]
Pam

letter = 'z'
letter * 10 = zzzzzzzzzz multiplication of strings.
2+3 = 5
'2' + '3' = 23 (concatenation)
X = "Hello world"
X. . . would give a list of possible operations

Scanned by CamScanner

X.upper() = 'HELLO WORLD'

However original x is still Hello world as we have not assigned it.

X.split()

o/p ['Hello', 'World'] → split ~~the~~ based on white space and create list

by default it is split on white space.

X = 'Hi this is a string'

X.split()

['Hi', 'this', 'is', 'a', 'string'].

X.split('i')

['H', 'th', 's', 's a str', 'ng']

String Interpolation

↳ Add a variable to a string for printing or other purposes

print ("Hello" + name) where name = "Siddharth"

→ 2 methods to do this

- format() method → intro.

f-strings (formatted string literals) - ~~newer~~ newer method for latest versions (3.6)

print ('This is a string {}'.format('INSERTED'))

This is a string INSERTED.

print ('String here {} then also {}'.format('something', 'something2'))

print ('the {2} {0} {1}'.format('fox', 'brown', 'quick'))

width allows you to kind of add in some whitespace if you have a really large with value.

print('The {q} {b} {f}'.format('f='fox', 'b'='brown', q='quick')) → float

Float formatting "{value:width.precision f}"
No of digits before decimal No of digits after decimal

result = 100/77 = .12870012870012870

print("The result was {r:1.3f}".format(r=result))
The result 0.129

f-strings (formatted String)

name = "Ishi"
print('My name is {}'.format(name))
↳ Instead of above, there is alternate way of doing as below.

print(f'My name is {name}')
My name is Ishi

Lists

- Ordered sequences thats why we can use indexing.
- Can hold variety of objects
- Use [ ] brackets and commas to separate objects in list.

Ex:- [1,2,3,4,5]

- Support indexing, concatenation and slicing. Lists can be nested and also have a variety of useful methods than can be called off of them → work same as string.

len(my_list) → gives no. of elements in list

* Most of operations of list are similar to string except mutation is possible in list.

Ex. new_list = ['one', 'two', 'three']

new_list[0] = 'ONE ALL CAPS'

new_list = ['ONE ALL CAPS', 'two', 'three']

new_list.append('four') → allows to append list with new element at end

new_list = ['ONE ALL CAPS', 'two', 'three', 'four']

Removing last element from list

// new_list.pop → If we run this without (), it gives <function list.pop>. meaning there is pop up func^n inside the list object. If you want to call it you have to use () parenthesis.

new_list.pop() gives 'four'

new_list gives ['ONE ALL CAPS', 'two', 'three']

↳ 'four' is removed

If we want to remove element at specific loc,

new_list.pop(0) gives 'ONE ALL CAPS'

now, new_list gives ['two', 'three']

* Reverse indexing also work with list just like string

Scanned by CamScanner

Sorting of string elements

new_list = ['a', 'e', 'x', 'b', 'c']
num_list = [4, 1, 0, 3]
new_list.sort() → does not return anything
but new_list gives ['a', 'b', 'c', 'e', 'x']

★ [... my_sorted_list = new_list.sort()
If you type, my_sorted_list returns nothing]

NoneType → Type for None object.

None → Special object in python
→ used to indicate no value and a lot of times people hold it as a placeholder. But it is the return value of a func^n or a method that doesn't actually return anything.

Instead,

new_list.sort()
my_sorted_list = new_list
my_sorted_list gives ['a', 'b', 'c', 'e', 'x']

Reverse method

num_list.reverse()

Indexing of nested list [1, 1, [1, 2]]

my_list [2, 1] gives 2 from [1]

	Ordered	Mutable
String	✓	✗
List	✓	✓
Tuples	✓	✗
Dict	✗	✓
Sets	✗	

Scanned by CamScanner

meaning while printing key or values they will not follow any order or ~~get~~ sort.

Dictionaries

- Unordered mapping ~~of~~ for string objects
- Key - Value pairing
- Allows users to quickly grab objects without needing to know the index location
- You call the key and the value is returned

  {'key1' : 'value1', 'key2' : 'value2'}

List or Dictionary ? which one to choose.

Dictionaries -
- objects retrived by key name.
- Unordered.
- Can't be sorted.

List - Objects retrived by loca^n.
- Ordered sequence
- ~~Sequence~~ can be indexed or sliced.

Ex: my_dict = {'key1': 'value1', 'key2': 'value2'}

my_dict['key1']

gives o/p 'value1'

Ex2: d = {'k1': 123, 'k2': [0,1,2], 'k3': {'insidekey': 100}}

d['k3'] gives {'insidekey': 100}

Stacking key ← d['k3']['insidekey'] gives 100

d['k2'][2] gives 2

Ex 3: d = {'key1': ['a', 'b', 'c']}

~~d['~~ mylist = d['key1']

my list gives ['a', 'b', 'c']

letter = my list[2]

letter give 'c'

letter.upper() gives 'C'

Scanned by CamScanner

★ d['key1'][2].upper() gives 'C'

Add new key value pair to dictionary.

d = {'k1':100, 'k2':200, 'k3':300}

d['k4'] = 300

d → gives {'k1':100, 'k2':200, 'k3':300, 'k4':400}

d['k1'] = 'NEW VALUE'

d → gives {'k1': 'NEW VALUE', 'k2':200, 'k3':300}

★ d.key() gives dict_keys(['k1', 'k2', 'k3'])

d.values()

d.items() gives {'k1':100, 'k2':200, 'k3':300}

Check orderedddict object lecture for ordered dictionary.

★ Sorting is not possible in dictionary because normal dictionaries are mappings not a sequence.

later on in larger piece of code.

Scanned by CamScanner

Camlin Page
Date / /

Tuples.

- Very similar to lists.
- Immutable
- Once element is inside tuple, it cannot be reassigned.
- Tuples use parenthesis : (1,2,3)

t=(1,2,3) mylist = [1,2,3]

t = ('one', 2).

t[0] = 'one'

t[-1] = 2

Ex:- t = ('a', 'a', 'b')

✓ t.count('a') gives 2

✓ t.index('a') gives 0.

↳ gives first time occurence of element.

Tuples	List
t= (1,2,3)	t=[1,2,3]
t[0] = 'NEW'	t[0]= 'NEW'
↳ gives err 'tuple' object does not support item assignment.	t gives ['NEW', 2, 3]

Benefits of tuples - Used when you are passing around objects in your program and you need to make sure that they don't accidently get changed.

- provides data integrity.
- The fact that we can't do reassignments like this by accident will get an error instead is going to be really useful when you want to make sure that elements don't get flipped or reassigned

Scanned by CamScanner

Sets → don't allow duplicate items

- ~~unique collection~~
- unordered collection of unique elements
- There can only be one representative of the same object.

```
myset = set()
① myset.add(1)
② myset
③ {1}
④ myset.add(2)
⑤ myset
⑥ {1,2}
⑦ myset.add(2)
⑧ myset
⑨ {1,2}
```

→ It still shows 1 and 2. It doesn't show 1,2,2.

- It has to be unique values in there.

Advantage

Casting a list to a set that we only get the unique values.

Ex:- mylist = [1,1,1,2,2,2,2,3,3,3,3].
set(mylist)
{1,2,3}

Ex:- set('Parallel') would return
set {'P', 'a', 'r', 'l', 'e'}

Scanned by CamScanner

Booleans

- operators

We can use 'None' keyword as a placeholder for an object that we don't want to assign yet. Later on we may want to assign 'b' to something in my code, But right now just use as a placholder.

① b
↳ gives error : nam 'b' is not defined.

② b = None

Files

Write File

%%writefile myfile.txt → creates file where jupyter notebook is installed.
Hello this is a text file
This is second line

It would save the file to the location where you have kept your .py files (where jupyter notebook is stored)

myfile = open('myfile.txt')

pwd — To check where Jupyter Notebook is saved

If file loca" is in other directory than Jupyter loca" — 'pwd' command is used.

myfile = open("C:\\Users\\Username\\Folder\\xyz.txt")

C:\\Users\\Sujatha Neelakannan' → This is the directory where notebook is currently ~~saved~~ located and that's the same location that any .txt files you want to work with should be saved.

myfile.read() → returns just a giant string of everything that's in text file.

Ex: 'Hello this is a text file \n This is the second line'

9663170710

1st/4 - 31st Jan

Camlin Page
Date / /

~~Takes~~ read as a string.

my file.read() → if we run it again gives o/p ''

Reason- There is cursor at the beginning of the file and when you read it the cursor goes all the way to the EOF and you need to reset the cursor or seek it back to zero in order to read it again

myfile.seek(0) → reset the cursor to zero.
Now, if we call myfile.read(), it will display complete text.

① Contents = myfile.read()
② Contents → reads contents of file

~~Sometimes grabbing everything~~ read as list where each element represents line.

myfile.readlines() → Gives each line as a separate object or elements in this list.

Advantage → We can loop through this list or can index off of this list for the lines themselves.

['Hello this is a text file \n,
'This is the second line']

Best Practices-

myfile.close() → all open file should be closed, ~~before closing~~ in order to not get any errors because let's say you open

this file somewhere else on your computer and you were trying to delete it you'd get an error saying Python is still using this file ~~somewhere else on your computer and you were trying to delete it~~ and you'd have to manually close it once you are done working with it to avoid any such errors.

Another way of working with file where we don't have to worry about closing it.

```
with open('myfile.txt') as my_new_file:
    contents = my_new_file.read()
```

variable for myfile.txt

→ any code within this block this indented block of code is going to use my_new_file as the variable of this text file.

· Contents → variable can be used outside block

```
with open('myfile.txt', mode='r') as myfile:
    contents = myfile.read()
```

mode = 'w' → to overwrite file completely.

mode = 'a' → adding more lines to the end of text file

Scanned by CamScanner

d['k']['nest key'] list3[2,2]

Reading, Writing, Appending Modes.

1. mode = 'r' is read only
2. mode = 'w' is write only (will overwrite files or create new)
3. mode = 'a' is append only (will add on to files)
4. mode = 'r+' is reading and writing.
5. mode = 'w+' is writing and reading (overwrites existing files or creates a new file!)

```
X = open('myfile.txt','w')
X.write('This is my file')
X.close()
```

100 ** 0.5 → square rot gives 10

* Build the list [0,0,0] in two separate ways.

```
# [0]*3
# list2 = [0,0,0]
  list2
```

* list4 = [5,3,4,6,1]

```
# sorted(list4)
# list4.sort()
  list4
```

Operator.

== (a==b) If the values of two operands are equal, then the condition becomes true

3.0 == 3 gives TRUE

'Bye' = 'bye' gives FALSE

'2' = 2 " "

Scanned by CamScanner

Chaining Comparison Operators

- and
- or
- not not 1==1 returns FALSE

Control flow - IF, elif, else statements

- Control flow uses colons and indentation (whitespace)
- Indentation system sets python apart from other languages
  ↳ easily readable, quick to prototype

```
if some-condition :
    # execute some code
elif some-condition :
    # do something
else :
    # do something
```

Ex: def myfunc(a):
    if a==True:
        return 'Hello'
    elif a==False:
        return 'GoodBye'

For loops

- To iterate over every element in the object (for ex every element in a list or every character in a string or iterate over every key in dictionary)

```
my-iterable = [1, 2, 3]
for item-name in my-iterable :
    print(item-name)
```

variable name

variable name → placeholder for every single item in your iterable object.
item-name represents the no's in the list

Scanned by CamScanner

```
for _ in 'Hello World':
    print ('Cool!')
```
→ prints Cool! 11 times.

Tuple Packing.

- Tuples have a little bit of a special quality when it comes to for loops.

```
mylist = [(1,2), (3,4), (5,6), (7,8)]
for item in mylist:
    print (item)
```

o/p.
(1,2)
(3,4)
(5,6)
(7,8)

(a,b) or a,b

Variable name that looks like tuple here

Tuple Unpacking

```
for (a,b) in mylist:
    print (a)
    print (b)
```

o/p.
1
2
3
4
5
6
7
8
9
10

- For loop for Dictionary.

```
d = {'K1':1, 'K2':2, 'K3':3}
for item in d:
    print (item)
```

o/p K1
K2
K3

By default if you iterate thru dictionary you only iterate thru keys

If you want to iterate thru items.

```
# for item in d.items():
        print (item)
```

o/p. ('k1', 2)
('k2', 2)
('k3', 3)

```
# for key, value in d.items():
        print (value)
```

o/p 1
2
3.

While loops

- Continue to execute a block of code 'while' some condition remains TRUE.

Ex: while my pool is not full, keep filling my pool with water.

```
while some_boolean_condition:
        # do something
else
        # do something different.
```

```
x=0
while x < 5:
    print (f'The current value of x is {x}')
        x=x+1  or  x+=1
else
    print ('X IS NOT LESS THAN 5')
```

Scanned by CamScanner

break, continue, pass

- These keywords add additional functionality

break - Breaks out of the current closest enclosing loop.

continue - Goes to the top of the closest enclosing loop.

pass - Does nothing at all.

```
x= [1,2,3]
for item in x:
    # comment
```

When we try to run this it gives syntax error "Unexpected EOF while parsing" because it uses indentation and whitespace as a crucial part of its code. It expects to basically have something here and even comment isn't enough to fill it.

Therefore, we can use 'pass keyword and pass just says do nothing at all and a lot of time programmers keep it as a placeholder to avoid syntax error as you can imagine implementing a larger piece of code and I know eventually I want to come up and fill out this for loop right now I do not wnant to deal with it

```
x= [1,2,3]
for item in x:
    # comment
    pass
print ('end of my script')
```

Scanned by CamScanner

Continue.

```
mystring = 'Sammy'
for letter in mystring:
    if letter == 'a':
        continue
    print(letter)
```

→ op S m m y

Range Function

```
mylist = [1,2,3]
for num in range(10):
    print(num)
```

It would print numbers from 0 to 9 i.e. not including 10

→ Now to specify starting index position

```
range(3,10)
```

It would print from 3 to 9 (10 not included)

→ To mention step size

```
range(0,10,2)
```

output → 0, 2, 4, 6, 8,

→ Cast range operator to list

```
list(range(0,11,2))
```

as range(0,11,2) wont print anything as it is a generator

— Generator is a special type of function that will generate information instead of saving it all to memory.

Eg. for ex range(0,100,1) would store a giant list in memory so

Enumerate

```
index_count = 0
for letter in 'abcde':
    print ('At index {} the letter is {}'.format
                (index_count, letter)
    index_count += 1
```

workaround -

```
index_count = 0
word = 'abcde'
for letter in word:
    print (word[index_count])
    index_count += 1
```

Enumerate is the workaround of not using index_count every time

```
word = 'abcde'
for item in enumerate(word):
    print (item)
```

Basically doing index count in the form of tuples

Output is in form of tuples

```
(0, 'a')
(1, 'b')
(2, 'c')
(3, 'd')
(4, 'e')
```

Now we can use tuple unpacking

```
for index, letter in enumerate (word):
    print (index)
    print (letter)
    print ('\n')
```

Scanned by CamScanner

ints

returns list of tuples

zip - opposite of enumerate

```
mylist1 = [1, 2, 3]
mylist2 = ['a', 'b', 'c']
zip (mylist1, mylist2)
```

Here we are zipping these two lists together and output gives a location ~~zip~~ where zip generator is waiting.

```
<zip at 0x2xx3456>
```

```
for item in zip(mylist1, mylist2)
    print (item)
```

```
o/p  (1, 'a')
     (2, 'b')
     (3, 'c')
```

tuples

* If lists are uneven, zip is only going to be able to go and zip together as far as the list which is the shortest.

```
list (zip(mylist1, mylist2))
o/p  [(1, 'a'), (2, 'b'), (3, 'c')]
```

* Very useful for zipping together to lists if you want to later use them together using tuple and packing.

Scanned by CamScanner

In keyword to check if an object is in a list

Ex: ① 'x' in [1,2,3] → iterable object

False

② 'x' in ['x', 'y', 'z']

True

③ 'a' in 'a world' → string

True

④ 'mykey' in {'mykey':345} →

True

⑤ d = {'mykey':345}

345 in d.values()

True

Min | Max

```
mylist = [10,20,30,40,50]
min (mylist)
10
max (mylist)
50
```

→ inbuilt library in python

from random import shuffle

↳ function in random library

Ex

```
mylist = [1,2,3,4,5,6,7,8,9,10]
shuffle (mylist)
mylist
[3,9,7,8,10,5,1,2,6,4]
```

→ in-place func^n meaning it operates in place on that list

If you run one more time, it will give different list

from random import randint

```
randint (0,100)
79
```

→ returns random integer

Input func" for user input

```
# input ('Enter a number here: ')
# result = input ('what is your name ? ').
What is your name ? Sid.
result
Sid.
# result = input ('Favorite number :')
Favorite Number :30
result
'30'
```

this is always string.

'30' → always gives string.

```
float (result)        or    int (result)
30.0 → float              30
```

```
# result = int (input ('Favorite number :')
Favorite number :20
type (result)
int
```

List Comprehensions.

- Unique way of quickly creating a list with Phytho
- Using a for loop along with .append() to create a list, list comprehension can also be used.

```
~~mylist = []~~ mystring = 'hello'
mylist = []
for letter in mystring:
    mylist.append (letter)
mylist
o/p. ['h', 'e', 'l', 'l', 'o'].
```

Using list Comprehension

```
mylist = [letter for letter in mystring]
mylist
['h', 'e', 'l', 'l', 'o']
```

Scanned by CamScanner

Print Prime
Fibonacci

Any operation can be perform on first variable name.

mylist = [num * 2 for num in range (0,11)]

mylist → gives square of all no's from 0 to 11

For printing even numbers,

mylist = [x for x in range(0,11) if x % 2 == 0]

mylist gives [0, 2, 4, 6, 8, 10]

results = [x if x % 2 == 0 else 'ODD' for x in range(0,11)]

Nested loop

```
mylist = [ ]
for x in [2,4,6]:
    for y in [100,200,300]:
        mylist.append (x* y)
mylist
[200, 400, 600, 400, 800, 1200, 600, 1200, 1800]
```

Using list comprehension above can be written

mylist = [x*y for x in [2,4,6] for y in [1,10,1000]]

```
# def dog_check (mystring):
    return 'dog' in mystring.lower()
```

Scanned by CamScanner

mylist.insert ← shift + Tab (in jyptex notebook)
or help(mylist.insert)
or https://docs.python.org/3/

Camlin Page
Date / /

~~Methods~~ Functions

- can be easily executed many times out rewriting

```
def name_of_function():
    '''
    Docstring explains function.
    '''
    print ("Hello")
```

```
>> name_of_function()
>> Hello
```

```
def add_function(num1, num2):
    return num1+num2
```

→ colon indicates there will be indented block of code

```
result = add_function(1,2)
print (result)
3
```

```
# def say_hello(name = 'NAME'):
    print ('hello' + name)
```

→ default argument to avoid getting error if I call func without passing argument

```
>> say_hello()
   hello NAME
```

```
# def dog_check(mystring):
    if 'dog' in mystring.lower():
    return TRUE
    else:
        return False
```

→ to find out if the word 'dog' is in a string.

```
dog_check ('My dog ran away')
```

Scanned by CamScanner

*args and **kwargs

```
def myfunc(a,b):
    return sum((a,b)) * 0.05
```

↳ can be tuple or list

```
myfunc(40,60)
```

* a and b are called positional arguments. 40 is assigned to 'a' because it was in the first position or first argument and that 60 is assigned to B because it was in the second position or second argument

If the no of func^n ~~arguments~~ parameters are more, therefore *args can be used instead which allows us to set arbitary number of argument

builds dictionary of key value pair ← treated as tuple in the func^n

*args

```
def myfunc(*args):
    print(args)
    return sum(args) * 0.05
```

→ returns tuple

'args' → this is an arbitrary choice, can be anything else also, However as a good coding practice use args or arguments

**kwargs

```
def myfunc(**kwargs):
    print(kwargs)
    if 'fruit' in kwargs:
        print('My fruit of choice is {}'.format(kwargs['fruit']))
    else:
        print('I did not find any fruit here')
```

→ returns dictionary

```
myfunc(fruit = 'apple')
```

o/p My fruit of choice is apple

Scanned by CamScanner

Camlin Page
Date / /

myfunc (fruit = 'apple', veggie = 'lettuce')
{'fruit': 'apple', 'veggie': 'lettuce'}

def myfun (*args, **kwargs):
    print (args)
    print (kwargs)
    print ('I would like {} {}'.format(args[0], kwargs['food']))

Args & kwargs has to be in this order only

myfunc (10,20,30, fruit='orange', food='eggs', animal='dog')
o/p (10,20,30)
{'fruit': 'orange', 'food': 'eggs', 'animal': 'dog'}
I would like 10 eggs.

* Useful while using outside libraries

Scanned by CamScanner

Lambda Expressions – Map, filter

Map — Way to create anonymous functions

```
def square(num):
    return num**2

my_nums = [1,2,3,4,5]
for item in map(square, my_nums):
    print(item)
```

o/p
1
4
9
16
25

or

```
list(map(square, my_num))
```

o/p [1,4,9,16,25]

* Map applies ~~exe~~ func (square) to every element of the list

* Map itself executes square func^n. That is why we do not put square with (). We just pass func^{n}tion

Filter function

→ returns iterator yielding those items of the iterable for which when you pass in the item into the func^n it's true.

Ex.
```
def check_even(num):
    return num%2==0
mynums = [1,2,3,4,5,6]
list(filter(check_even, mynums))
[2,4,6]
```

or
```
for n in filter(check_even, mynums):
    print(n)
```

Scanned by CamScanner

Lambda Expressions / Anonymous func^n

* when you do not have to use func^n repeatedly, therefore there is not any requirement to use def keyword or actually give a name to func^n.

Ex:-
```
def square(num):
    result = num**2
    return result
```
⟶ `lambda num: num**2`

↓

These are used in conjuctions with other func^n such as map and filter

```
square(3)
```
o/p 9

```
# lists(map(lambda num: num**2, mynums))
  [1, 4, 9, 16, 25, 36]
# list(filter(lambda num: num%2==0, mynums))
  [2, 4, 6]
```

IMP:- we should use lambda expressions when you can easily read it if you come back to code later

Scanned by CamScanner

Nested Statements and Scope

- Variable name is stored in what's called the namespace

LEGB Rule:

L: Local - Names assigned in any way within a function (def or lambda) and not declared global in that function.

E: Enclosing function locals - Names in the local scope of any and all enclosing functions (def or lambda), from inner to outer.

G: Global (module) - Names assigned at the top-level of a module file, or declared global in a def within the file.

B: Built-in (Python) - Names preassigned in the built-in names module: open, range, SyntaxError

```
name = 'This is a Global String'  — global
def greet():
    name = 'Samy'  → Enclosing func^n
    def hello():
        name = 'This is local'  → local
        print ('Hello' + name)
    hello()

greet()
```

★ However, if you want to access global variable inside func^n

Scanned by CamScanner

(1) x=50

```
def func()
    global x
    x = 100
```

→ This should be avoided and it is not a good programming practice

```
func()
print(x)
100
```

Rather use this to change value of global variable.

(3) x=50

```
def func(x)
    x = 100
    return x

print (x)
50
x = func (x)
x = 100
```

Scanned by CamScanner

class - Blueprint that defines the nature of a future object. From class, we can create instance of object and instance is a specific object created from particular class.

OOP (object oriented programming)

- Allows to create own objects that have methods and attributes.

for user defined object ← class NameOfClass1):

basic way to define object is using class keyword. ∴ objects are also called classes in python.

① represents instance of object itself
② Most object oriented lang. passed this as a hidden parameter to the methods defined on an object. But python does not do this you have to declare explicitly by 'self' keyword.

```
def __init__(self, param1, param2):
```

→ method

this allows to create instance of actual object

→ init method is called automatically when you create the instance object

self keyword connects this method to the instance of the class and it allows us to refer to itself and then we pass in any attribute that we want the user to define

```
self.param1 = param1
self.param2 = param2
```

attribute name

```
def some_method(self):
    # perform some action
    print(self.param1)
```

Conventions:-

Class - Camel Casing SampleWord

variable name & func^n → Snake Casing

Methods & Functions

```
class Dog():
    def __init__(self, breed):
        self.breed = breed
```

Instance of Sample class ← my_sample = sample()

```
my_dog = Dog(breed='Lab')
type(my_dog)
__main__.Dog
mydog.breed
Lab
```

→ my_dog is of type Dog class

this means that this particular instance of Dog is connected to main script there*

# class object attribute
# Same for any Instance of class,
Species = 'mammal' → common for all the instance of class

Camlin Page
Date

class Dog():
    def __init__(self, mybreed):
        self.breed = mybreed

→ argument (type of variable can be any, str, int, float, list etc)

↓ attribute name

# OPERATIONS / ACTIONS ---> Methods.
def bark(self):
    print("WOOF")

my_dog = Dog(mybreed = 'Huskie')
my_dog.breed
Huskie

my_dog.species
'mammal'

→ class attribute. This is going to be same regardless of instance of my_dog.

def bark(self):
    print("WOOF! My name is {}".format(self.name))

Methods can take outside arguements
def bark(self, number):
    print("WOOF! My name is {} and the number is {}".format(self.name, number))

Global attributes can be accessed using
1) self.species or
2) Circle.species ~~inside the class~~ ~~inside the cl~~

Scanned by CamScanner

```
class Circle():
    # CLASS OBJECT ATTRIBUTE
    pi = 3.14

    def __init__(self, radius=1):
        self.radius = radius
        self.area = radius * radius * Circle.pi
                                      or
                                      self.pi

    # METHOD
    def get_circumference(self):
        return self.radius * Circle.pi * 2
```

Inheritance and Polymorphism

Inheritance - is a way to form new classes using classes that have already been defined.

Advantages

- Ability to reuse code
- Reduces complexity of a program

```
class Animal():          — Base Class
    def __init__(self):
        print("Animal Created")
    def who_am_i(self):
        print("I am an animal")
    def eat(self)
        print("I am eating")
```

Scanned by CamScanner

Passing base class as argument means I am going to inherit or derive from this base class

derived class ← class Dog(Animal): Dog inherits Animal

instance of dog class is created here

```
    def __init__(self):
        Animal.__init__(self)
        print("Dog Created")
    def who_am_i(self):
        print("I am a dog!")
    def bark(self):
        print("WOOF!")
mydog = Dog()
```

instance of animal class created while instance of dog class created as I am inheriting Animal class

in case you want to overrite animal class method.

steps executed →
① init method of dog called
② " " " animal "
③ Prints 'Animal Created
④ Prints Dog created

gives [ANIMAL CREATED
Dog Created

```
mydog.eat()
mydog.who_am_i()
```

→ derived from base class. and this is the main idea behind inheriance

gives [I am a dog

Polymorphism:-

- Different object classes can share same method name
- Those methods can be called from same place even though a variety of different objects might be passed in.

```
class Dog():
    def __init__(self,name):
        self.name = name
    def speak(self):
        return self.name + "says woof!"

class Cat():
    def __init__(self,name):
        self.name = name
    def speak(self):
        return self.nam + " says meow!"
```

Scanned by CamScanner

```
niko = Dog("niko")
felix = Cat("felix")
```
created 2 instances of above classes

```
print (niko.speak())
niko says woof!

print (felix.speak())
felix says meow!
```

There are few different ways to demonstrate polymorphism

①
```
for pet in [niko, felix]:
    print (type(pet))        → what type of class it is
    print (type(pet.speak()))
```

```
< class '__main__.Dog' >
< class 'str' >
< class '__main__.Cat' >
< class 'str' >
```

This is example of polymorphism as both Niko and Felix share the same method name speak.

② 2nd alternative — using func^n.

```
def pet_speak (pet):
    print (pet.speak())

pet_speak(niko)
niko says woof!
pet_speak (felix)
felix says 'meow!
```

Scanned by CamScanner

Abstract Class → class of which object can't be created
→ serve as base class only.

```
class Animal():
    def __init__(self, name):
        self.name = name
    def speak(self):
        raise NotImplementedError("Subclass must implement this abstract method")
```

* Animal class is not supposed to be instantiated

```
myAnimal = Animal()
myAnimal.speak()
```

→ gives error as this class is not supposed to be instantiated. It is expecting you to inherit the animal class and then override speak method.

```
class Dog(Animal):
    def speak(self):
        return self.name + " says woof!"

class Cat(Animal)
    def speak(self):
        return self.nam + " says meow!"

fido = Dog("fido")
isis = Cat("Isis")
print(fido.speak())
fido says woof!
print(Isis.speak())
Isis says Meow!
```

Scanned by CamScanner

Special methods / Magic methods / ~~dsue their~~

- How to use built in functions such as length or print with my own user defined objects.
- They are called so as they use double underscores
- class Book():

```
    def __init__(self, title, author, pages):
        self.title = title
        self.author = author
        self.pages = pages

    def __str__(self):  → special func"
        return f"{self.title} by {self.author}"

b = Book('Python rocks', 'Jose', 200)
print(b) → gives book object loc" in memory
                → asks string representation of b
```

② <__main__.Book object at 0x099325>

Therefore, if we try to cast b into string, we get a string version of ②

Instead we can use special method to that ~~string call~~ related to that string call

```
def __str__(self).
```

∴ If there is ever any func" that asks (print(b)) for the string representation of book class then it returns whatever this method returns.

```
print(b)
Python rocks by Jose
str(b)
Python rocks by Jose
```

Scanned by CamScanner

To delete a variable from memory

del b
↳ variable name

def __del__(self):
print ("A book object has been deleted")

IMP. coor1 = (3,2) & coor2 = (4,5)
These are coordinates in the form of tuple and passed as parameters.
Tuple Unpacking is effective way of getting x,y co-ordinates.

x_1, y_1 = self.coor1
x_2, y_2 = self.coor2

Django and flask library are used for web development of Python

~~OOO challenge~~ Modules and Packages

- PyPi is a repository for open-source third party Python packages
- pip install → command can be used on cmd line to install these packages.
- pip → to download packages directly from the PyPi repository.

① - cmd > pip install package name
Ex. pip install colorama → This command is for installing colorama package.

② ~~Go the~~ At cmd > python.
③ >> from colorama import init
>> init()
>> from colorama import Fore
>> print (Fore.RED + "some red text")

Scanned by CamScanner

To install openpyxl from PyPi.

```
> pip install openpyxl
> python
>> import openpyxl
```

Packages - are collection of modules.

file location(\ \ ) > python myprogram.py

Ex:-

```
Desktop > MyMainPackage
          |-> __init__.py
          |-> some_main_script.py
          |-> [SubPackage]
                |-> __init__.py
                |-> mysubscript.py
          |-> myprogram.py
          |-> mymodule.py
```

```
from MyMainPackage import some_main_script
from MyMainPackage.SubPackage import mysubscript
some_main_script.report_main()
mysubscript.sub_report()
```

O/P Hey I am in some_main_script in main package
Hey I am inside mysubscript

Scanned by CamScanner

if __name__ == "__main__":

* if name is equal to main block of code that is usually at the end a larger .py script files.

- used ~~to know whether they are called directly from cmd or being~~ used to know whether the file in which this line of code is written is called directly or for some other .py file

Scanned by CamScanner

Errors and Exception Handling

Following are 3 keywords :

1. try - This is the block of code to be attempted
2. except - Block of code will execute in case there is an error in try block.
3. finally - A final block of code to be executed, regardless of an error

Ex1

```
try :
    # WANT TO ATTEMPT THIS CODE
    # MAY HAVE AN ERROR
    result = 10 + 10
except: → If there is error in try block, we will run except block
    print ("Hey it looks like you aren't adding correctly")
else : → If there is no error, we run this block of code.
    print ("Add went well!")
    print (result)
```

Ex2.

```
try:
    f = open('testfile', 'w')
    f.write ("Write a test file")
except TypeError :
    print ("There was a type error!")
except OSError :
    print (" You have an OSS Error')
finally :
    print ("I always run")
```

* Both else and finally can also be used simultaneously but generally it is avoided.

Scanned by CamScanner

Unit testing of Python code files

- To ensure unit & Integration testing
- To ensure changes in one python file do not impact other python files/code

Two main testing tools (though several out there)

- pylint - Look at code and report back possible issues, styling issues or may be if there is some invalid code
- unittest - Allows to test programs and check if you are getting desired outputs.

* Python has a set of style convention rules known as "PEP 8"

① cmd >> pylint module.py

② Unit Testing - Unit test allows you to write your own test program.
Basically, there are 2 scripts:

ⓐ - your own program.
ⓑ - test script for program created at ⓐ

hello
give error

Scanned by CamScanner

Decorators

Adding new functionality to old code/func^n

- This can be done in 2 ways.

① Either you can add the extra code to old func^n but then we cannot call original func^n.

② To create brand new func^n where you copied all the old code and then added new code to that. The problem here is that you had to recreate the entire func^n over again even though it's just a copy and paste.

Prob:- What if we want to remove that extra functionality

* We would need to delete it manually or make sure to have the old func^n.

~~Is there a~~

Decorators provides the ~~p~~ solu^n to above problem.

- It allows to tag on extra functionality to an already existing func^n.
- Use @ operator and is placed on top of the original func^n.

```
@some_decorator
def simple_func():
    # Do simple stuff
    Return something.
```

→ connects to some extra code

```
def hello()
    return "Hello!"
```

① hello()
→ 'Hello!'

② hello
→ <function main.hello>

③ greet = hello → Here we are not actually executing func^n But we can actually assign func^n to other variables and then execute them off that variable

④ greet()
→ 'Hello!'

⑤ del hello

⑥ greet() → Even though we deleted the name Hello the name greet is still pointing to original func^n obj.
→ 'Hello!'

* It is important to know that functions are objects that can be passed into other objects.

Passing in a function within another func^n or calling functions within other func^n.

```
def hello(name = 'José'):
    print ('The hello() function has been executed!')
    def greet():
        return '\t This is the greet() func inside hello!'
    def welcome():
        return '\t This is welcome() inside hello'
    print (greet())
    print (welcome())
```

* Scope of greet() and welcome() is limited to hello func^n
* If we try calling them outside hello, it'll give error

Scanned by CamScanner

```
# def hello(name = 'Jose')
    print('Hello function has been executed')
    def greet():
        return '\t This is greet function inside hello'
    def welcome():
        return '\t This is welcome inside hello'

    print("I am going to return a func")
    if name == 'Jose':
        return greet
    else:
        return welcome
```

① my_new_func = hello('Jose')
↳ The hello() func has been executed!
I am going to return a function!!
Here greet() is assinged to my_new_fun

② my_new_fun
<function_main__.hello.<locals>.greet>

③ my_new_fun()
'\t This is the greet() func inside hello!'
greet() assigned in ① gets executed here and it returns the

④ print(my_new_func())
This is greet() func inside hello!
* Here we are returning a func^n (greet()) within another func^n (hello())

Scanned by CamScanner

```
# def Cool():
    def super_cool():
        return 'I am very cool!'
    return super_cool

① some_func = cool()
② some_func
↳ <function __main__.cool.<locals>.super_cool>
③ some_func()
↳ 'I am very cool'

# ① def hello():
    return 'Hi Jose'!'

② def other(some_def_func):
    print('other code runs here!')
    print(some_def_func())

④ hello
↳ <function __main__.hello>
⑤ hello()
↳ 'Hi Jose!'
⑥ other(hello)
↳ Other code runs here!
↳ Hi Jose!
```

Scanned by CamScanner

contains extra functionality to decorate original func"

```
# ** def new_decorator(original_func):
        def wrap_func():
            print('some extra code before the origin func')
            original_func()
            print('some extra code after original func')
        return wrap_func
```

↳ returns the wrapped version of original func

→ original function

① def func_needs_decorator():
    print("I want to be decorated!")

② func_needs_decorator()
↳ I want to be decorated!!

③ new_decorator(func_needs_decorator)

** decorated_func = new_decorator(func_needs_deco

④ decorated_func()

*** ↳ Some extra c------ before original func
I want to be ---deco.
Some ------ after original "

IMP: ~~line marked ** can be avoided by~~
~~and~~ This will call line marked ** above and execute wrap func" and type same o/p as ***
Benefit of decorator is you can anytime comment it ~~out as per your convenience.~~

ⓐ new_decorator → ON-OFF SWITCH

```
def func_needs_decorator():
    print("I want to be decorated").
```

① func_needs_decorator()

IMP:- We can be using web framework or someone else library and just adding in these new decorator to maybe render new web site or point to another page.

Scanned by CamScanner

They are commonly used in web frameworks such as jingle or flask

Flask - web framework
- Allows to create web pages in python
- Decorators are heavily used.

Django - more heavy duty
- more popular web framework for Python

Framework - s/w library
- provides generic functionality which can be extended by the programmers to build application.
- Flask and Django are examples of frameworks intended for web development

Diff b/w framework & API

Framework is distinguished from a simple library or API. API is a piece of s/w that developer can use in his application. A framework is more encompassing your entire application is structured around the framework (i.e. it provides the framework around which you build your software).

Generators with Python

Generator func^n → Allow us to write a func^n that can send back a value and then later resume to pick up where it left off.

- Allow us to generate a sequence of values over time instead of having to create entire sequence and hold it in memory
- when generator func^n is compiled they become object that supports an iteration protocol.

Scanned by CamScanner

- That means when they are called in code they don't actually return a value and then exit.
- Automatically suspend and resume their execution and state around the last point of value generation.
- Adv is that instead of having to compute an entire series of values up front and hold in memory, the generator computes one value waits until next value is called for
- Ex:- range() doesn't produce an list in memory for all the values from start to stop.
  * just keeps track of last number and the step size, to provide flow of numbers
  * If user needs the list, they have to transform the generator to a list

```
list(range(0,10))
```

```
# def create_cubes(n):
    result = []
    for x in range(n):
        result.append(x**3)
    return result
```

→ creates list of cubes and stores in memory

create_cubes(10)

[0,1,8,27,64] → using normal func^n, we're keeping this entire list in memory. This may be useful when you want entire list.

Another perspective

```
for x in create_cubes(10):
    print(x)
```

→ Here we are not storing complete list in memory. We needed one value at a time to print. We just need previous value and then using formula to get the next value to generate these cubes.

Op
0
1
8
27
64

Using generator comprehension

```
my_list = [1,2,3,4,5]
gencomp = (item for item in my_list if item>2)
for item in gencomp:
    print(item)
```

o/p 3 4 5

```
def create_cubes(n):
    for x in range(n):
        yield x**3
```

→ don't have this list in memory. Instead of list yielding the values as they come.

```
for x in create_cubes(10):
    print(x)
```

↳ is a generator and generating values as you need them.

```
list(create_cubes(4))
```

[0,1,8,27,64] → Does not store cubes in memory. Rather it creates on the fly and keep adding to list.

Fibonnaci Sequence (USING GENERATOR)

```
def gen_fibon(n):
    a=1
    b=1
    for i in range(n):
        yield a
        a,b = b,a+b

for number in gen_fibon(4):
    print(number)
```

↳ This is memory efficient

WITHOUT USING GENERATOR

```
def gen_fibon(n):
    a=1
    b=1
    output = [ ]
    for i in range(n):
        output.append(a)
        a,b = b,a+b
    return output
```

↑ This is not memory efficient as hold everything in memory

NEXT & ITER FUNCTION

```
def simple_gen():
    for x in range(3):
        yield x

for number in simple_gen():
    print(number)
```

o/p 0
1
2

Scanned by CamScanner

```
g = simple_gen()
g
<generator object simple_gen at 0x00x000>

print(next(g))
0
print(next(g))
1
print(next(g))
2
print(next(g))
```

— This is what generator object is doing internally when you call yield keyword.

↓ gives error 'Stop Iteration' because after yielding all the values next calls a stop iteration error, and this error informs us that all the values have been yielded.

→ Using for loop we do not get this error. because it automatically catches this error and stops calling next.

Iter() — allows to automatically iterate through a normal object that we may not expect.

Ex. `s = 'hello'`

```
for letter in s:
    print(letter)
```

O/P
h
e
l
l
o

★ String itself is not iterable object. To make it iterable we pass it into iter()

next(s) → gives error that 'str' object is not a iterator

Scanned by CamScanner

s_iter = iter(s) → This is how we can convert objects that are iterable into iterators themselves

next (s_iter)
'k'

next (s_iter)
'e'

IMP:- 'yield' keyword can be used to create our generators.

COLLECTIONS MODULE

Collections module is a built-in module that implements specialized container datatypes providing alternative to Python's general purpose built-in containers.

Counter → is a dict subclass
→ Helps count hashable objects.
→ Inside of its elements are stored as dictionary keys and the counts of the objects are stored as the value.

```
from collections import Counter
l = [1,1,1,1,2,2,3,4,5,2,2,2,3,3,4,4,4,12],
Counter(l)
Counter({2:5, 1:4, 4:4, 3:3, 12:1})
```

* Similar exercise can be done with strings also

```
c = Counter(l)
c.most_common(2) → gives 2 most common nos.
[(2,5), (1,4), (4,4)]
```

Common functions in Counter

sum(c.values) → total no. of all counts (17)

c.clear() → reset all counts.

Scanned by CamScanner

list(c) → list unique elements [2, 5, 1, 12, 4, 3]

set(c) → convert to a set {1, 2, 3, 4, 5, 12}

{2:5, 1:4, 4:4, 3:3, 12:1} dict(c) → convert to a regular dictionary

c.items() → convert to a list of (elem, cnt) pairs

dict_items([(2,5), (1,4), (4,4), (5,3), (12,1)])

Counter(dict(list-of-pairs))

→ convert from a list of (elem, cnt) pairs

c.most_common()[:-n-1:-1]

→ n least common elements

c+ = Counter() → remove zero and -ve counts.

defaultdict

→ It is a dictionary like object which provides all methods provided by dictionary but takes first argument (default_factory) as default data type for the dictionary.

→ Using defaultdict is faster than doing the same using dict.set_default method.

★ A defaultdict will never raise a KeyError. Any key that does not exists get the value returned by the default factory.

```
from collections import defaultdict
d = {'k1':1}
d = defaultdict(object).
d['one']
<object at 0x40xx>
for item in d:
    print item
```

★ d does not have any keys. It will assign whatever the factory condition here for that value to

★ So, we can this to initialise default values

o/p one ~~This is the value returned by default factory.~~

d = defaultdict(lambda:0)

d['one'] → 0 — since there is no assignment for d['one'] it always returns zero. It did not throw any error, it goes by the default that you pass.

d['two'] = 2

d

→ defaultdict(<function <lambda> at 0x00002148>, {'two':2, 'one':0})

★★ defaultdict will never throw key error. Instead it will initialise the default value that we place when you create the default dictionary.

Ordered Dict

- is a dictionary subclass that remembers the order in which its contents are added.

★ Normal dictionary is just the mapping, it doesn't retain the order.

★ ~~Order~~ Regular dictionary only looks at contents when testing for equality, order dictionary also consider order of the items as they were added.

from collections import OrderedDict.

d = OrderedDict()

Normal Dict.	Ordered Dict.
d1 = {}	d1 = OrderedDict()
d1['a'] = 1	d1['a'] = 1
d1['b'] = 2	d1['b'] = 2
~~d2['b'] = 2~~	d2 = OrderedDict()
d2 = {}	d2['b'] = 2
d2['b'] = 2	d2['a'] = 1
d2['a'] = 1	Print d1 == d2
Print d1 == d2	FALSE
TRUE	

Scanned by CamScanner

namedtuple

t = (1,2,3)
t[0]
1

We can use indexing to grab elements from the tuple. However, remembering which index should be used is hard to remember or if tuple has large no. of values. This might lead to errors.

namedtuple ~~Named~~ - To avoid these errors, it assigns names, as well as numerical index, to each member in the tuple.

```
from collections import namedtuple
Dog = namedtuple('Dog', 'age breed name')
sam = Dog(age=2, breed='Lab', name='Sammy')
```

is the object class Dog,

2 arguments
- ① Name of class
- string, with the various attribute fields

```
sam
Dog(age=2, breed='Lab', name='Sammy')
sam.age
2
sam[0]
2
```

namedtuple is just like a normal tuple, except that
① it allows names for the various fields.
② it creates a new object type

Datetime

```
import datetime
t = datetime.time (5,25,1)
                   (Hr, min, sec)
print t
↳ 05:25:01
t.hour
↳ 25
print datetime.time.min
↳ 00:00:00
print datetime.time.resolution
↳ 0:00:00.000001
```

Time

→ Time values are represented with the time class

→ Attributes (hour, minute, second, microsecond)

→ can also include timezone information

→ time instance only holds values of time, and not a date associated with a time.

```
today = datetime.date.today()
print today
↳ 2018-12-29
today.timetuple()
datetime.date.min
↳ datetime.date(1,1,1)
d1 = datetime.date(2015,3,11)
print (d1)
↳ 2018-12-29
d2 = d1.replace(year=1990)
d2
↳ datetime.date(1990,12,29)
```

→ represents today's date using today() class method.

Date

→ attributes (year, month & day)

Arithmetic on time differences.

```
d1-d2
↳ datetime.timedelta(9131)
```

→ difference in days.

Scanned by CamScanner

Python Debugger

PDB → Python's built-in debugger module

```
import pbd

x = [1,3,4]
y = 2
z = 3
result = y + z
print result
pdb.set_trace()
result2 = y+x
print result2
```

This allows to basically pause code at the point of the trace and check if anything is wrong.

Key here is we always just set this trace to right before we think error is occurring, come to the where you see error, type pdb.set_trace()

this gives interactive debugging environment.

and then, when you run your code, use the debugger interactive environment there to find the source of your error.

PDB → module implements an interactive debugging environment for Python program

→ includes features to let you pause your program, look at the values of variables, and watch program execution step-by-step.

* Use 'q' to quit the debugger

timeit perform the same code in this line, through 10000 loops, and report back best of 3.

Scanned by CamScanner

Timing your code

- It is important to know how long code is taking to run
- If a particular line of code is slowing down entire project.
- timeit → provides simple way to time small bits of python code.
- has both command line interface as well as callable one.

```
import timeit
'0-1-2-3-.....-99'
"-".join(str(n) for n in range(100))
```

1st method.

used to time this method ← `timeit.timeit('"-".join(str(n) for n in range(100))', number=10000)`

↓ pass entire exp. as string

no. of time you want to run this

Time ~~Took~~ to perform 10,000 loops,

0.247888 ↳ took about .248 secs and it will be diff. depending on computer.

Now, running it as list comprehension

`timeit.timeit('"-".join([str(n) for n in range(100)])', number=10000)`

↳ 2nd method.

0.221066

running as map fun → 3rd method.

`timeit.timeit('"-".join(map(str, range(100)))', number=10000)`

0.1573034 → GIVE BEST RESULT

★★ Above is way to use timeit in ~~you~~ code if you weren't working in a Jupyter notebook.

In Jupyter notebook, the same can be done using built-in magic and defined as

```
%timeit "-".join(str(n) for n in range(100))
```

10000 loops, best of 3: 28.9 µs per loop.

Scanned by CamScanner

Camlin Page
Date

% timeit "-".join([str(n) for n in range(100)])
→ 100,00 loops, best of 3: 26.2 us per loop.

fastest method
% timeit "-".join(map(str, range(100)))
100,000 loops, best of 3: 14.4 us per loop
↳ map was so much faster than the other methods, jupyter iPython ran it 100,000 times instead of just 10000

* If it takes long time to run, jupyter iPython may just run it 3 times. If it's very fast, it will run it more times, to get a better average.

* iPython will limit the amount of real time it will spend on timeit procedure. So if you're running 100,000 loops that would have taken 10 mins, iPython automatically going to reduce the number of loops to something more reasonable, like 100 or 1000 depending on how long it actually takes.

Regular Expressions :- (re)

- text matching patterns.
- used for finding repetition, matching text etc.
- Also parsing problems can be solved

Ex:- (to search some pattern in text)

```
import re
patterns = ['term1', 'term2']
text = 'This is a string with term1, but not the other term
re.search('hello', 'hello world')
```

→ <_sre.SRE_Match at 0x41d487> → gives match object

pattern text

```
for pattern in patterns:
    print 'Searching for "%s" in: \n "%s" ' % (pattern, text),
    # check for match
    if re.search (pattern, text):
        print '\n'
        print 'Match was found. \n'
    else:
        print '\n'
        print 'No match was found. \n'
```

→ if match object exists print match found.

O/P. Searching for "term1" in:
"This is a string with term1, but not the other term"
Match was found.
Searching for "term2" in:
"This is a string with term2, but not the other term"
No Match was found"

```
# print re.search ('h', 'w')
```
↳ None → None is returned when there is no match.

```
# match = re.search (patterns[0], text)
type (match)
```
↳ _sre. SRE_Match → match object
↳ Contains info. about the match, i.e. original input string, regular exp. that was used, location of the match. So we can call methods on this match objects.

```
match.start()
```
↳ 22

```
match.end()
```
↳ 27

Scanned by CamScanner

Split with regular expression

split_term = @

phrase = 'what is your email, is it hello@gmail.com?'

re.split(split_term, phrase)

['what is your email, is it hello', 'gmail.com?']

Finding all instances of a pattern

re.findall('match', 'Here is one match, here is another match')

↳ returns list of all matches.

['match', 'match']

Metacharactess - used along with re to find specific types of patterns.

```
def multi_re_find(patterns, phrase):
    '''
    Takes in a list of regex patters
    Prints a list of all matches
    '''
    for pattern in patterns:
        print('Searching the phrase using the
                re check: %r' % (pattern))
        print(re.findall(pattern, phrase))
        print('\n')
```

format character

value in pattern get assigned to r,

Conversion	Meaning
c	Single character (accepts integer or single chars)
r	String (Converts any python objects using repr())
s	String (Converts " " " " str())
%	No argument is converted, results in a "%" character in the result.

Scanned by CamScanner

Repetition Syntax

There are 5 ways to express repetition in a pattern:

Ex: test_phrase = 'sdsd... sssddd... sddddsddd... dsds... dssss ... sdddd'

test_patterns = [① 'sd*', # s followed by 0 or more d's
② 'sd+', # s followed by 1 or more d's
③ 'sd?', # s followed by 0 or 1 d's
④ 'sd{3}', # s followed by 3 d's
⑤ 'sd{2,3}', # s followed by 2 to 3 d's
]

multi_re_find (test_patterns, test_phrase)

① gives o/p ['sd', 'sd', 's', 's', 'sddd', 'sddd', 'sddd', 'sd', 's', 's', 's', 's', 's', 's', 'sddddd']
② gives o/p ['sd', 'sd', 'sddd', 'sddd', 'sddd', 'sd', 'sddddd']
③ gives o/p ['sd', 'sd', 's', 's', 'sd', 'sd', 'sd', 'sd', 's', 's', 's', 's', 's', 's', 'sd']
④ gives o/p ['sddd', 'sddd', 'sddd', 'sddd']
⑤ gives o/p ['sddd', 'sddd', 'sddd', 'sddd']

Characters Sets

- Characters sets are used when you wish to match any one of a group of characters at a point in the input.
- Brackets are used to construct character set inputs.

Ex: input [ab] searches for occurences of either a or b

test_patterns = [① '[sd]', # either s or d
② 's[sd]+'] # s followed by 1 or more s or d

o/p ① ['s', 'd', 's', 'd', 's', ...]
② ['sdsd', 'sssddd' . . .]

Exclusion.

^ → to exclude terms by incorporating it into the bracket syntax notation.

[^...] → will match any single character not in the brackets.

test_phrase = 'This is a string! But it has punctuation. How can we remove it?'

re.findall('[^!.?]+', test_phrase)

↓ + to check that the match appears at least once.

This basically translates into finding the words.

o/p. ['This',
'is',
'a',
'string',
.
.
.
'it']

Character Ranges

- format used is [start-end]

Ex: [a-f] would return matches with any ~~occurs~~ occurrence of letters b/w a and f.

test_phrase = 'This is an example sentence. Lets see if we can find some letters.'

test_patterns = ['[a-z]+', # sequence of lower case letters
'[A-Z]+', # sequence of upper case "
'[a-zA-Z]+', # sequence of lower or upper case letters
'[A-Z][a-z]+'] # one upper case letter followed by lower case letters.

Scanned by CamScanner

Escape codes

→ special escape can be used to find specific types of patterns in your data, such as digits, non-digits, whitespace and more

Code	Meaning
\d	a digit
\D	a non-digit
\s	whitespace (tab, space, newline etc.)
\S	non whitespace.
\w	alphanumeric
\W	non-alphanumeric

Escapes are indicated by prefixing the character with a backslash. Unfortunately, a backslash must itself be escaped in normal Python strings, and that results in expressions that are difficult to read. Using raw strings, created by prefixing the literal value with r, for creating regular expressions eliminates this problem and maintains readability.

This use of r to escape a backslash is probably one of the things that block someone who is not familiar with regex in Python from being able to read regex code at first.

```
test_phrase = 'This is a string with some numbers
        1233 and a symbol #hashtag'
test_patterns = [①r'\d+',  # sequence of digits
                 ②r'\D+',  # sequence of non-digits
                 ③r'\s+',  # sequence of whitespace
                 ④r'\S+',  # sequence of non-whitespace
                 ⑤r'\w+',  # alphanumeric characters
                 ⑥r'\W+',  # non-alphanumeric
```

Scanned by CamScanner

① o/p ~~Searching~~ ['12333']

② ['This is a string with some numbers ', ' and a symbol # hashtag']

③ [' ', ' ', ' ', ' ', ' ']

④ ['This', 'is', 'a', 'string', 'with', 'some', 'numbers', '1233', 'and', 'a', 'symbol', 'hashtag']

⑤ Same as ④

⑥ [' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#']

String IO – Files we opened files that exist outside of python, and streamed their contents into an in-memory file object.

We can also create in-memory file-like objects within our program that python treats the same way.

Text data is stored in a StringIO object, while binary data would be stored in a BytesIO object. This object can then be used as input or o/p to most functions that would expect a standard file object.

```
import StringIO
message = 'This is just a normal String'
f = StringIO.StringIO(message)
```

↳ Use StringIO method to set a file object

↳ this object we will be able to treat just like file

```
f.read()
f.write
f.seek(0)
f.close
```

↳ Close the object when contents are no longer needed.

Scanned by CamScanner

2**4 = pow(2,4) = 16

pow(2,4,3) = 15

↳ (2**4) % 3

round(3.1) = 3.0

round(3.9) = 4.0

round(3.14159 2, 2) = 3.14

abs(-3) = 3.

hex(246) = '0xf6'

bin(128).

Advanced Strings

s.capitalize()	s = 'hello world'
	o/p Hello World
s.upper()	HELLO WORLD
s.lower()	hello world.
s.count()	2
s.find('o')	4
s.	hello world.
s.center(20,'z')	z = total length used for centre of string. 'zzzzhello worldzzzzz'
print('hello\thi)	'hello hi'
or 'hello\thi'.expandtabs()	
s='hello' s.isalnum()	TRUE. Checks if all characters are alphanumeric
s.isalpha()	→ TRUE
s.islower()	TRUE
s.isspace()	FALSE
s.istitle()	FALSE. Returns TRUE only if s is title case string and there is at least one character in s
s.isupper()	FALSE
s.endswith('o')	TRUE
or s[-1]='o'	

Scanned by CamScanner

separator

s = 'hihihhihiii'

s.partition('i') ('h', 'i', 'hihhihiii')

head tail

★★ partition will separate only at first instance while split will do it at every instance.

Advanced Sets

s = {1, 2}

s.clear() - clears complete set

sc = s.copy() - creates copy of set s as sc

s.add(4) = {1,2,4} - adds 4 to existing set

→ Difference b/w 2 sets

s.difference(sc)

returns 4 as sc was {1, 2} s was {1,2,4}

s should have more elements than sc

sc.difference(s) will not work

→ Difference_update

s1 = {1,2,3}

s2 = {1,4,5}

s1.difference_update(s2)

Now, s1 = {2,3} - returns first set after removing all the elements found in second one.

→ Discard

s1.discard(2) will give {1,3}

However if we discard a number which is not there in s1, it wont give any ~~ex~~ error and return the set as is.

s1.discard(12) will give {1,3}

→ Intersection of 2 sets

s1.intersection(s2) = {1}

S1 = {1,2}
S2 = {1,2,4}
S3 = {5}

→ Intersection update
S1.intersection_update(S2)
S1 = {1}

→ Chk disjoint (another way of intersection)
S1.isdisjoint(S2) → returns false as they have intersection
S1.isdisjoint(S3) → returns ~~Fa~~ True

→ chk if a set is a subset of other
S1.issubset(S2) → returns True

→ chk for superset
S2.issuperset(S1) → True

→ Symmetric difference b/w 2 sets
will return all elements other than the intersection from both the sets (Normal difference gives ~~ab~~ diff from say S1 - S2)
S1.symmetric_difference(S2)
{4}

→ Union of both sets
S1.union(S2) → {1,2,4}

→ Update set
S1.update(S2) has overwritten S1
returns S1 = {1,2,4} copy creates a seperate copy

Advanced Dictionaries

Dictionary Comprehension

→ {x : x**2 for x in range(10)}

Where x → Key
x**2 → Value

→ If values are not based on numbers
{K : v**2 for K, V in zip(['a', 'b'], range(2))}
→ {'a' : 0, 'b' : 1}

→ Dictionaries can be iterated using Iter()

```
for K in d.iteritems():
    print K
```

→ ('K2', 2)
('K1', 1)

→ for K in d.itervalues():
    print K

→ 2
1

similarly with keys

→ Use view methods
d.viewitems()
→ dict_items([('K2', 2), ('K1', 1)])

d.viewkeys()
→ dict_keys(['K2', 'K1'])

d.viewvalues()
can be used for values

Scanned by CamScanner

$l = [1, 2, 3]$

Camlin Page
Date / /

Advanced lists

l.append(4) → [1, 2, 3, 4]
l.count(10) → 0 (# of occurances of 10 in the list)
l.count(1) → 1

x = [1, 2, 3]
x.append([4, 5])
print x → [1, 2, 3, [4, 5]]
append method appends entire object
x.extend([4, 5]) → [1, 2, 3, 4, 5]

l.index(2) → 1
l.index(10) → ValueError: 10 is not in list
l.insert(2, 'inserted') → [1, 2, 'inserted', 3]
index — object that you want to insert.
l.pop() → [1, 2, 'inserted']
l.remove('inserted') → [1, 2]
↳ removes 1st occurence of value
Ex:- l = [1, 2, 3, 4, 5, 3]
o/p → [1, 2, 4, 5, 3]
l.reverse() → [3, 2, 1]
l.sort() → [1, 2, 3]

To extract domain from email ID

domainGet('user@domain.com') → domain.com
where domainGet is a function

Scanned by CamScanner

from imblearn.over_sampling import SMOTE

os = SMOTE()

os.fit_sample(X_train, Y_train)

RFE (Recursive feature elimination)

Sklearn.metrics → ROC_auc_score

ROC_curve

Scanned by CamScanner