## features:

user friendly

create data anywhere

assign any data type anywhere, anytime

platform independent

extensive libraries

comprehensive documentation

### background translators:

C language Cpython (translates & uses C libraries)

Java language Jython (converts code for JVM)

…

…

…

## Data Types:

other languages: int varx = 30;

python: varx = 30

### Numbers:

int integers

float decimal places

complex (10 + 7j)

### bool:

True

False

## operators:

### arithmetic:

+ add

- subtracts

\* multiplication

/ division

% remainder

// floor division

\*\* power of (exponent)

### relational, comparative:

< less than

> greater than

<= less than or equal to

>= greater than or equal to

== equal to

!= not equal to

### logical:

and

or

not

### identity:

is

is not

### membership:

in

not in

### False:

value: 0

0.0

bool False

None (similar to void)

empty ‘ ‘ [ ] { }

## strings:

text data

double quotes or single quotes

(triple double quotes or triple single quotes help us with multi line strings)

index

negative index

slicing :

upper range not included

no errors for out of bound

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| o | n | s | e | m | i |
| 0 | 1 | 2 | 3 | 4 | 5 |
| -6 | -5 | -4 | -3 | -2 | -1 |

immutable

functions:

upper, lower

count, index

isupper, islower

isalpha, isdigit, isalnum

strip, rstrip, lstrip

….

….

ascii

### escape sequences:

\n new line

\t tab

\b backspace

may not be supported:

\v vertical tab

\r carriage return

\f form feed

### raw strings:

attach ‘r’ at the start of string

### comments:

# single line

""" .. """ multi lines

### doc strings:

triple double quotes or triple single quotes

create documentation

\_\_doc\_\_

## functions:

### general:

print

type

round

len

min()

max()

sum()

any()

all()

### conversion:

list()

tuple()

set()

int()

float()

str()

## Data Structures:

### lists:

similar to collections of java, STL of C++

heterogeneous data

index

negative index

slicing

mutable

nesting

functions:

append, insert

remove, pop, clear

reverse, sort, sort(reverse=True)

count, index

extend

copy

references

### tuples:

heterogeneous data

index

negative index

slicing

**immutable**

nesting

functions:

count, index

### set:

unique members (no duplicates)

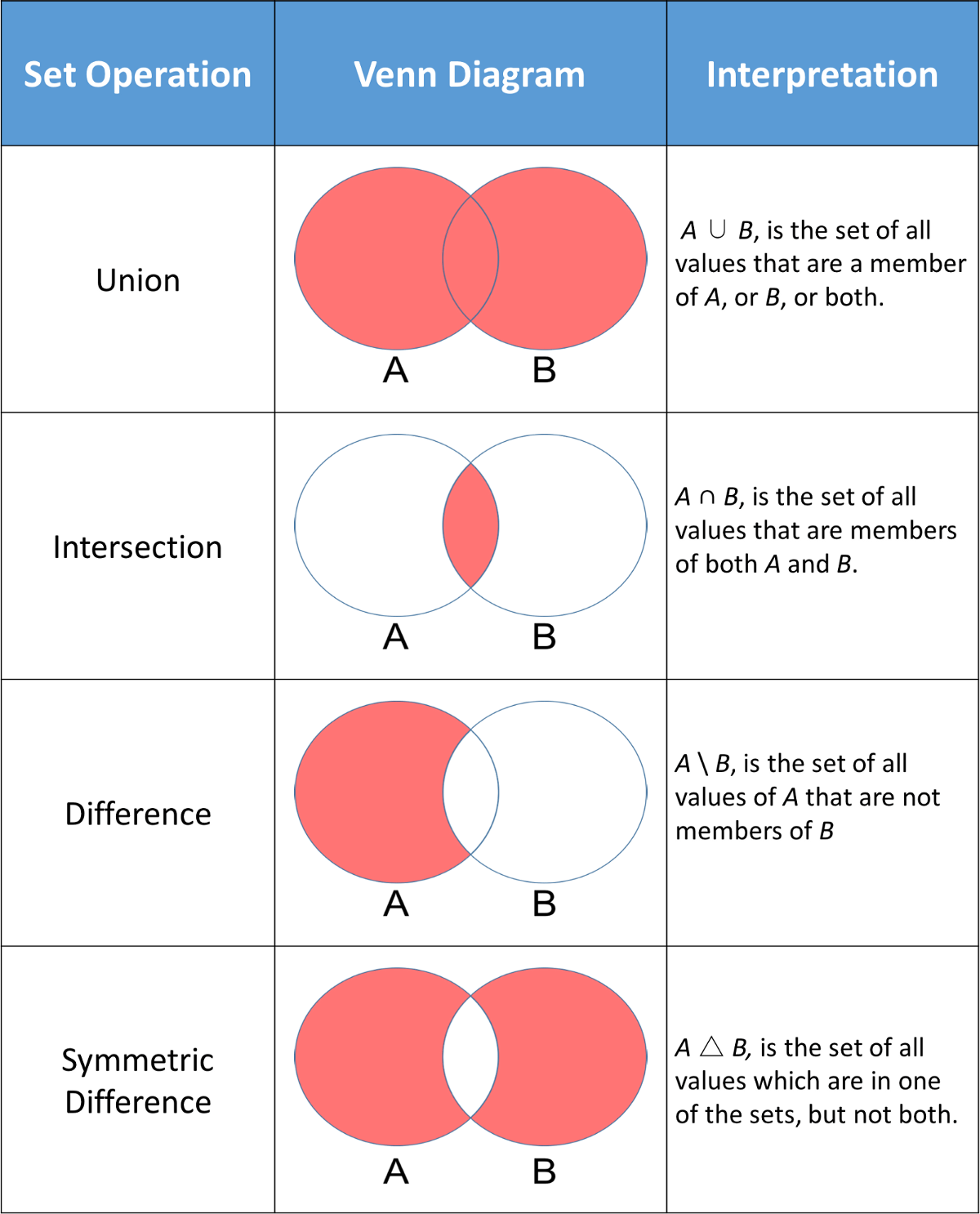
unordered

sets by themselves are mutable

members of sets have to be immutable

functions:

…..



### dictionary:

pairs of key & value

unordered

index

keys:

unique (no duplicates)

immutable

values:

anything

dictionaries by themselves are mutable

functions:

update

## Flow Control:

: colon

indentation (tab)

pass

~~switch case~~

~~do while~~

### if else

if

elif

else

### while:

while

break

### for:

for

break

continue

else

## User Defined Functions

procedures

routines

subroutines

methods

functions

pass:

arguments, parameters

any number of parameters

positional arguments

default values to arguments

default arguments to right most

named arguments

variable number of arguments

\*args = tuple

variable number of keyword arguments

\*\*kwargs = dict

return:

can return anything

returns None by default

can return any number of values

recursion :

function calling itself

controlled by an exit policy

## memory:

C, C++

|  |  |  |
| --- | --- | --- |
| location | value | name of variable |
| 6784 | 30 | varx |
| 6788 | 24 | vary |
| 6792 | 30 | varz |
| 6796 | 101 | vara |

python

|  |  |  |
| --- | --- | --- |
| location | value | name of variable |
| 6784 | 30 | varx, varz |
| 6788 | 24 | vary |
| 6792 | 100 |  |
| 6796 | 101 | vara |
| 7890 | [varx, vary, 23, 33] | lista, listb |
| 6888 | “hello” | stra |

## modules:

inbuilt

os

sys

math

datetime

time

statistics

requests

json

openpyxl

5 ways of importing:

1. import colour

colour.blue()

colour.yellow()

colour.green()

1. from colour import blue, yellow

yellow()

blue()

~~green()~~

1. from colour import \*

yellow()

blue()

green()

1. import colour as c

c.blue()

c.yellow()

c.green()

1. from colour import blue as b

b()

points to remember:

1. does not dynamically update
2. does not fetch again
3. can be reloaded using importlib.reload

### os:

os.path.exist()

os.path.isfile()

os.path.isdir()

os.mkdir()

### special functions:

map

filter

lambda

# OOP:

class

encapsulation

inheritance

polymorphism

private

# exception handling:

try except finally else

nesting is allowed

outer except can handle an inner except

scenarios:

1. exception occurred, handled

except block

finally

continues

1. exception occurred, not handled

finally

stopped the execution

1. exception never occurred

else

finally

continues

# concurrent programming:

(parallel)

multithreading

multiprocessing

multiprocessing

ta = multiprocessing.Process(target=funca)

tb = threading.Thread(target=funcb)

ta.start()

tb.start()

ta.join()

tb.join()

# External interfaces

### files:

open()

f.close()

f.read()

f.readline()

f.readlines()

f.write()

f.tell()

f.seek()

f.closed

f.readable()

f.writable()

modes:

r read

w write

a append

r+ read & write

w+ read & write

### Databases:

oracle, MySQL cx\_oracle

SQL server pyodbc

sqlite sqlite3

### sqlite3:

1. connection
2. cursor
3. queries (transaction)

### json:

javascript object notation

data about the data

(meta data)

multilevel dictionary

## (web) Frameworks:

django

flask

cherryPy

web2py

turbogears

grok

pyramid

HTML

CSS

JS

## requests:

HTTP protocol

client server application-level protocol

probably runs over TCP/IP

asymmetric request-response client-server protocol

stateless protocol

client server model

requests:

purpose of needing requests: HTTP

How:

client & server communicate using text messages

few headers & a message body

GET & POST

to speak to the cloud (example WMS )

GET method:

most common HTTP request method

client uses to request (or “get” ) a piece of resource from an HTTP server

retrieve

not-to-modify

example: GET /index.html HTTP/1.0

POST method:

send additional data to the server

example: submitting a form

HTML form method=”post”

PUT

ask the server to store data

DELETE

ask the server to delete data

response status code:

1xx informational

request received, server is continuing the process

2xx success

request received successfully, understood, accepted & serviced

3xx redirection

4xx client error

5xx

1.

make a request

2.

pass parameters in URLs

3.

response content

response.text in unicode

response.content content (bytes)

response.json() json format

response.raw : to get raw socket response

4.

custom header

5.

complicated POSTS

6.

POST a multipart-encoded file

7.

timeout

8.

Errors and Exceptions

requests.exceptions.RequestException