## Basics

interpreted

REPL

Read Eval Print Loop

Data = everything is an object

### extensions & types

.py

.sh

#!/usr/bin/python3

.ipynb

python notebook files

jupyter

cell

shift + enter to execute

spyder

### comments

# single line

doc strings used as multiline

""" """

''' '''

## installation

### ubuntu

$sudo apt install python

$sudo apt install python3-pip

$pip install flask

## Data Types

### basic

int

float

str

complex

4+6i

4+6j

### bool

True

False

### False

0

0.0

[ ]

''

{ }

False

None

## Key Words

and or not

in is

False True None del

if else elif

for while break continue

def return

global nonlocal

import as

with

pass

try except finally else raise

## Functions

### general

print

type

input

eval

### sequences

len

min

max

sum

all

any

sorted

### convert (cast)

int

float

str

list

tuple

set

## Operators

### arithmetic

+

-

/

\*

% modulo (remainder)

// floor division

\*\* power (raised to)

= assignment

### conditional (relational)

<

>

<=

>=

==

!=

### logical

and

or

not

### membership

in

not in

### identity

is

is not

## strings

text data

ascii, unicode

index

negative index

error prone

slice

no indexerror

step

| p | u | s | h | p | a |
| --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 |
| -6 | -5 | -4 | -3 | -2 | -1 |

immutable

### 

### string functions:

upper, lower, capitalize, title

index, rindex, count

isalpha, isalnum, isdigit

replace

split, splitlines

join

… … …

… … …

### escape sequences:

\n new line

\t tab

\b backspace

\\ \

\’ ‘

\” “

\r, \v, \f

### doc strings:

""" """

''' '''

used by some programmers as multiline comment

works only in scripts

used by all programmers for documentation

has to be first line of

class

method

function

modules

\_\_doc\_\_

## Data Structures

## list

any kind of data

all kinds of data

duplicates allowed

index

negative index

slice

step

nesting

mutable

### list functions

sort, sort(reverse=True)

reverse

index, count

insert, append, extend

remove, pop, clear

copy

## tuple

any kind of data

all kinds of data

duplicates allowed

index

negative index

slice

step

nesting

immutable

### tuple functions

index, count

## sets

no duplicates

unordered

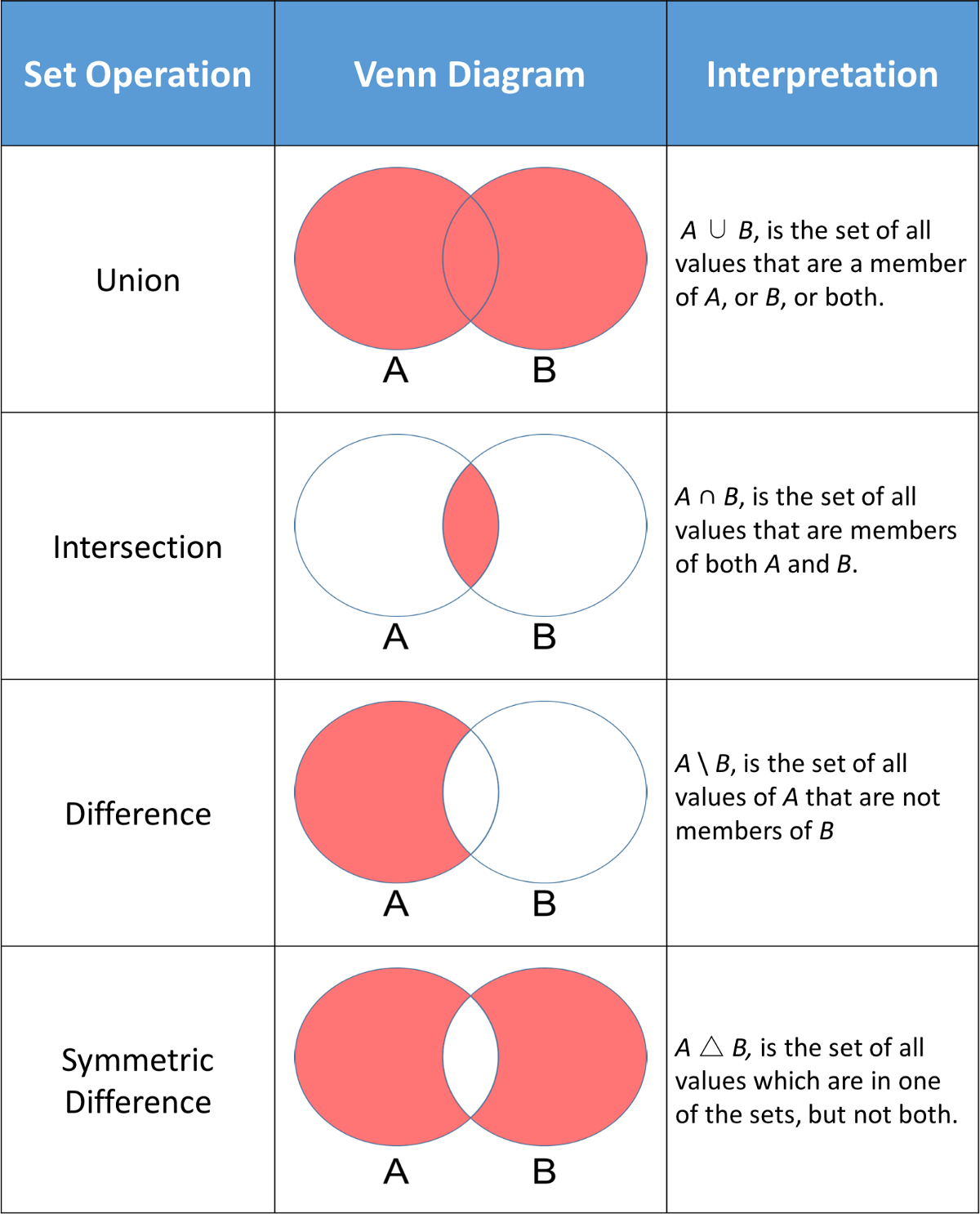
no index

no slice

sets are mutable

members have to be immutable

no nesting



functions:

… … …

## dict

key:value pairs

keys:

immutable

unique

values:

anything

dict themselves are mutable

nesting is possible (as values)

unordered 3.6 & before

ordered 3.7 & later

functions:

values, keys

items

## flow control

### if else

if

elif

else

~~switch case~~

### while

while

~~do while~~

else with while

### for

with sequences

with range

for, break, continue, else

else with for

## User Defined Functions

def

no function overloading

latest definition stays

default arguments (values )

always to the right most

named arguments

return

can return multiple values

default is None

\*args

\*\*kwargs

global

### higher order functions

## Modules

\_\_builtin\_\_

\_\_builtins\_\_

standard library:

<https://docs.python.org/3/library/index.html>

extra packages:

<https://pypi.org>

ways to import:

1. import colours

colours.blue()

colours.yellow()

colours.green()

1. from colours import blue, yellow

blue()

yellow()

~~green()~~

1. from colours import \*

blue()

yellow()

green()

1. import colours as c

c.blue()

c.yellow()

c.green()

1. from colours import blue as b

b()

## OOPs

DRY Don’t repeat Yourself

class

self

\_\_init\_\_()

called by default

### encapsulation

\_\_object

private

~~protected~~

~~inner~~

do not get inherited

### inheritance

multilevel

C++, Java, C#, Python, Ada, Algol…….

multiple

C++, Python

private do not get inherited

### polymorphism

function overloading

C++, Java, C#

operator overloading

C++, Python

magic methods

operator precedence doesn't change

can not create new operators

\_\_add\_\_

friday + saturday

class assumed is of the left data

\_\_radd\_\_

friday + saturday

class assumed is of the right data

chaining of operators

### magic methods

\_\_add\_\_

\_\_mul\_\_

\_\_eq\_\_

\_\_lt\_\_

## File Handling

open

with open

### modes:

r read

w write

(always opens file like a new files)

a append

r+ read & write

b binary

### functions

open

close

read

read(n)

readline

readlines

write

writeline

readable

writable

attribute:

closed

## Regular Expressions

### Meta Characters

^ if starts with a certain character

$ if ends with ascertain character

. any single character

except new line

[ ] set of characters to match

[a-z] all lower

[ \_-] space/underscore/hyphen

[0-9] all digits

[a-zA-Z] all alphabets

[0-39] [01239]

[^ ] invert of characters to match

[^a-z] anything except lower case alphabets

\* zero or more occurrences of the pattern left to it

+ one or more occurrences of the pattern left to it

? zero or one occurrence of the pattern left to it

| alteration

{ } number of repetitions

{2, 4} min 2 times , max 4 times

\ escape various characters (and metacharacters) \

### Special sequences

\d any digit

\D any non digit

\s whitespace character

space, tab, new line, \v, \r, \f

\S invert of \s

any non whitespace

\w alphanumeric (including underscore)

[a-zA-Z\_]

\W invert of \w

re.match

re.findall

re.sub

re.subn

re.split

## Exception Handling

try except as finally else raise

### points:

1. try followed by except (always)
2. default except always at the end
3. nesting
4. custom exceptions

derived from Except class

1. raise exceptions
2. can be received like objects

### situations:

1. no exception

else block, finally block, code continues

1. exception raised, handled

except block, finally block, code continues

1. exception raised, not handled

finally block, code stops

## Concurrency

### multi threading

threading

threading.Thread

start

no priority

prints get distorted

## Test

### test runners

unittest

pytest

nose2

blocktest

### unittest

#### Messages:

OK .

FAIL F

ERROR E

skip s

expected failure x

#### options:

-m multiple (custom choice)

-q quiet

-v verbose

-f stop on first fail

#### methods:

setUp() before each test case

initialising

tearDown() after each test case

@classmethod

setUpClass() before any test cases in a class

@classmethod

tearDownClass() after any test cases in a class

@timeout\_decorator.timeout()

#### skip methods

@unittest.skipIf()

@unittest.skipUnless()

@unittest.skip()

@unittest.expectedFailure

#### asserts:

assertEqual()

assertNotEqual()

assertTrue(expr)

expression should be True

assertFalse()

assertIs()

assertIsNot()

assertIn()

assertNotIn()

assertGreater()

assertGreaterEqual()

assertLess()

assertLessEqual()

assertAlmostEqual()

default 7 decimal places

assertNotAlmostEqual()

assertListEqual(la, lb)

assertTupleEqual(ta, tb)

assertDictEqual()

self.assertRegex

### pytest:

look for files starting with name test\_

-v verbose

-k match a string

-m match a marker

-n number of worker threads-

@pytest.mark.markername

@pytest.mark.parametrize

@pytest.mark.skip

@pytest.mark.xfail

## Parked topics

### (if time permits)

~~higher order functions~~

decorators

### to be done

packages

\_\_main\_\_

if \_\_name\_\_ == “\_\_main\_\_”

~~@classmethod~~

~~@staticmethod~~

magic mock and monkey patch in pytest?

## 