## basics

### REPL

Read Eval Print Loop

### keywords

and or not

True False None

if elif else

pass

while for

break continue

def return

## environments & editors

IDLE

pycharm

visual studio

vscode

vi & python

jupyter

(anaconda)

spyder

## installation

python.org

anaconda.com

linux command line (ubuntu)

sudo apt install python3

sudo apt-get install python3-pip

## Data Types

int

float

str

complex

5 + 6j

### bool

True

False

### False

0

0.0

""

[ ]

{ }

( )

False

None

## comments

# single line comment

doc-strings used as multi line comment

## operators

### arithmetic

+

-

\*

/

%

// floor division

\*\* power of

=

no pre increment or decrement

### logical

and

or

not

### relational (conditional)

<

>

<=

>=

==

!=

### bitwise

&

|

!

^

<<

>>

### membership

in

not in

### identity

is

is not

## Functions

### general

print()

type()

help()

### sequences (iterable)

len()

### cast (convert)

int()

float()

str()

complex()

list()

tuple()

set()

## strings

text data

double or single quotes

index

negative

slice

no IndexError (out of bound)

step

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| p | a | n | d | a |
| 0 | 1 | 2 | 3 | 4 |
| -5 | -4 | -3 | -2 | -1 |

immutable

### string functions

|  |  |
| --- | --- |
| capitalize() | Returns a copy of the string with its first character capitalized and the rest lowercased. |
| casefold() | Returns a casefolded copy of the string. Casefolded strings may be used for caseless matching. |
| center(width[, fillchar]) | Returns the string centered in a string of length width. Padding can be done using the specified fillchar (the default padding uses an ASCII space). The original string is returned if width is less than or equal to len(s) |
| count(sub[, start[, end]]) | Returns the number of non-overlapping occurrences of substring (sub) in the range [start, end]. Optional arguments start and end are interpreted as in slice notation. |
| endswith(suffix[, start[, end]]) | Returns True if the string ends with the specified suffix, otherwise it returns False. suffix can also be a tuple of suffixes. When the (optional) start argument is provided, the test begins at that position. With optional end, the test stops comparing at that position. |
| expandtabs(tabsize=8) | Returns a copy of the string where all tab characters are replaced by one or more spaces, depending on the current column and the given tab size. T |
| find(sub[, start[, end]]) | Returns the lowest index in the string where substring sub is found within the slice s[start:end]. Optional arguments start and end are interpreted as in slice notation. Returns -1 if sub is not found. |
| format(\*args, \*\*kwargs) | Performs a string formatting operation. The string on which this method is called can contain literal text or replacement fields delimited by braces {}. |
| index(sub[, start[, end]]) | Like find() but raises a ValueError when the substring is not found (find() returns -1 when the substring isn't found). |
| isalnum() | Returns True if all characters in the string are alphanumeric and there is at least one character. Returns False otherwise. |
| isalpha() | Returns True if all characters in the string are alphabetic and there is at least one character. Returns False otherwise. |
| isdigit() | Returns True if all characters in the string are digits and there is at least one character. Returns False otherwise. |
| islower() | Returns True if all cased characters in the string are lowercase and there is at least one cased character. Returns False otherwise. |
| isnumeric() | Returns True if all characters in the string are numeric characters, and there is at least one character. Returns False otherwise. |
| isspace() | Returns True if there are only whitespace characters in the string and there is at least one character. Returns False otherwise.  Whitespace characters are space, tab, new line, vertical tab etc. |
| istitle() | Returns True if the string is a titlecased string and there is at least one character (for example uppercase characters may only follow uncased characters and lowercase characters only cased ones). Returns False otherwise. |
| isupper() | Returns True if all cased characters in the string are uppercase and there is at least one cased character. Returns False otherwise. |
| join(iterable) | will be discussed during the session |
| lower() | Returns a copy of the string with all the cased characters converted to lowercase. |
| partition(sep) | Splits the string at the first occurrence of sep, and returns a 3-tuple containing the part before the separator, the separator itself, and the part after the separator. If the separator is not found, it returns a 3-tuple containing the string itself, followed by two empty strings. |
| replace(old, new[, count]) | Returns a copy of the string with all occurrences of substring old replaced by new. If the optional argument count is provided, only the first count occurrences are replaced. For example, if count is 3, only the first 3 occurrences are replaced. |
| rfind(sub[, start[, end]]) | Returns the highest index in the string where substring sub is found, such that sub is contained within s[start:end]. Optional arguments start and end are interpreted as in slice notation. This method returns -1 on failure. |
| rindex(sub[, start[, end]]) | Like rfind() but raises ValueError when the substring sub is not found. |
| rpartition(sep) | Splits the string at the last occurrence of sep, and returns a 3-tuple containing the part before the separator, the separator itself, and the part after the separator. If the separator is not found, it returns a 3-tuple containing the string itself, followed by two empty strings. |
| rsplit(sep=None, maxsplit=-1) | Returns a list of the words in the string, using sep as the delimiter string. If maxsplit is given, at most maxsplit splits are done, the rightmost ones. If sep is not specified or is set to None, any whitespace string is a separator. |
| rstrip([chars]) | Return a copy of the string with trailing characters removed. The chars argument is a string specifying the set of characters to be removed. If omitted or set to None, the chars argument defaults to removing whitespace. |
| split(sep) | Returns a list of the words in the string, using sep as the delimiter string. |
| splitlines() | Returns a list of the lines in the string, breaking at line boundaries. |
| startswith(prefix[, start[, end]]) | Returns True if the string starts with the specified prefix, otherwise it returns False. prefix can also be a tuple of prefixes. When the (optional) start argument is provided, the test begins at that position. With optional end, the test stops comparing at that position. |
| strip([chars]) | Returns a copy of the string with leading and trailing characters removed. The chars argument is a string specifying the set of characters to be removed. If omitted or set to None, the chars argument defaults to removing whitespace. |
| swapcase() | Returns a copy of the string with uppercase characters converted to lowercase and vice versa. |
| title() | Returns a title-cased version of the string. Title case is where words start with an uppercase character and the remaining characters are lowercase. |
| translate(table) | will be discussed in the intermediate session |
| upper() | Returns a copy of the string with all the cased characters converted to uppercase. |

### escape sequences

"\n" new line

"\t" tab

"\b" back space

"\\" \

"\"" "

"\r" carriage return

"\v" vertical tab

## list

anything (all and any data type)

allows duplicates

index

negative

slice

step

no index error

mutable

when ‘=’ is used

create reference

nesting

any level deep

### list functions

append, insert. extend

pop(), pop(num), remove, clear

sort, reverse, sort(reverse=True)

copy

ndex, count

explanation of list within list

x = [a, b, c, d]

x.insert(x)

[a, b, c, [a, b, c, d], d]

## tuple

anything (all and any data type)

allows duplicates

index

negative

slice

step

no index error

immutable

nesting

any level deep

### tuple functions

count, index

## set

no duplicates

only unique

unordered

~~index, slice~~

members:

immutable

(numbers, strings, pure tuples)

by itself set is mutable

no nesting

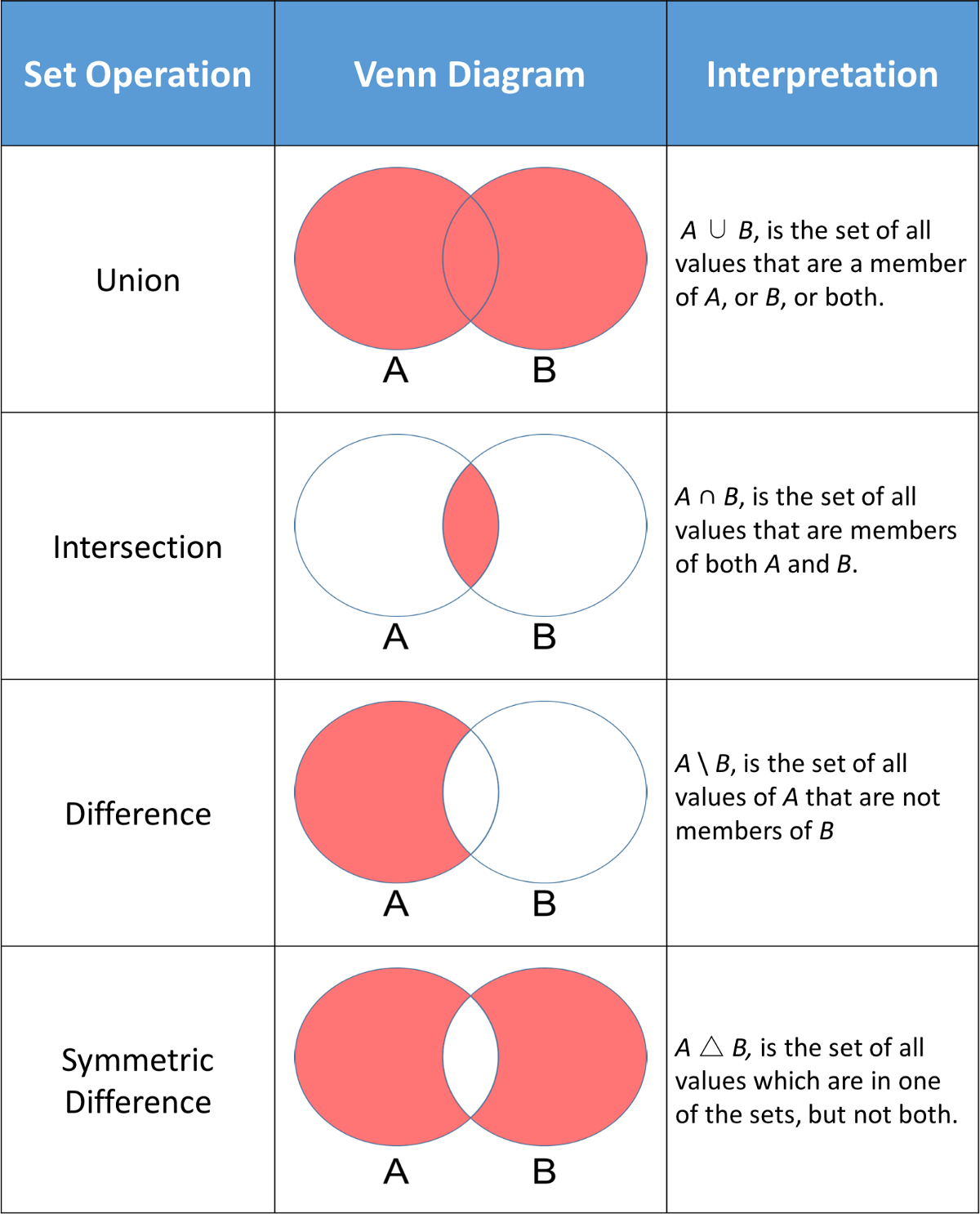
### set operations

| union

& intersection

- difference

^ symmetric difference



### set functions

## dict

key : value

indexed

keys

ordered (3.7 +)

keys:

immutable

strings, numbers, pure tuples

unique

values:

anything

modifiable

by itself is mutable

### dict functions

clear, pop, popitem

update

copy

## flow control

### if else

if

elif

else

### while

while

break

continue

while-else

### for

for

for with sequences

for with range

break

continue

for-else

## functions (user defined)

def

indentation

no overloading

latest definition is used

arguments (parameters)

any number

return

default None

any number

default arguments (values)

always to the right most

named arguments

arbitrary (variable number)

args

kwargs

high order functions

pure functions

global

## module

executes the entire module once

does not re-import

import importlib

importlib.reload(colour)

### ways to import

1. import colour

colour.blue()

colour.red()

colour.purple()

1. from colour import blue, red

blue()

red()

~~purple()~~

1. import colour as c

c.blue()

c.red()

c.purple()

1. from color import blue as b

b()

1. from colour import \*

blue()

red()

purple()

## files

fa = open()

fa.read(n)

fa.realine()

fa.readline(7)

fa.readlines()

fa.write()

fa.readable()

fa.writable()

fa.tell()

fa.seek()

fa.close()

fa.closed

### file open modes

r read

w write

(always creates a new file)

a append

r+ read and write

b binary

explanation of hash table

1 “apple”

2 “banana”

3 “plum”

4 [7, 8, 9, 10, 22]

5 [