Python cheatsheet

Global AI Hub

IMPORTING

import numpy	– imports numpy library
import numpy as np	– imports numpy library with alias np
from numpy import array	– imports only array function / method from numpy library
from numpy import *	– imports all modules from numpy library

COMMENTING

```
# This is a comment line therefore won't be interpreted by Python.
'''
This is a multiline comment.
Therefore won't be interpreted by Python.
'''
```

DATA TYPES

int( )	– 8, -17, 65536
float( )	– 0.2384, -7.32, 17.0
complex( )	– 5+2j, -3+4j, -7-2j
str( )	– "This is a string"
bool( )	– True, False

USER INPUT

```
my_int   = int(input( ))
my_float = float(input( ))
my_comp  = complex(input( ))
my_str   = input( )
my_bool  = bool(input( ))
```

default for input() is str, therefore no need to write str(input())

DEFINING VARIABLE

my_variable = value	– defines a variable

FUNCTIONS

`def my_func(num1, num2):`	– defines function name and arguments
`    result = num1 + num2`	– defines an operation inside (there may be many)
`    return(result)`	– returns the result of the function
`my_func(6, 7)`	– calls the function

STRING METHODS

my_str = "Global AI Hub"	– creates a string
my_str.capitalize( )	– converts the first character to upper case
my_str.upper( )	– converts a string into upper case
my_str.lower( )	– converts a string into lower case
my_str.title( )	– converts the first character of each word to upper case
my_str.count("a")	– returns the occurrence number of specified character
my_str.index("b")	– returns the position of specified character
my_str.isalpha( )	– returns True if all characters in string are alphabetic
my_str.isnumeric( )	– returns True if all characters in string are numeric
my_str.islower( )	– returns True if all characters in string are in lower case
my_str.isupper( )	– returns True if all characters in string are in upper case
my_str.split(" ") seperator	– splits the string at the specified separator, and returns a list
my_str.strip( )	– removes spaces at the beginning and at the end

TRY - EXCEPT - ELSE - FINALLY

`try:`
`    print(x/y)`
	– try block lets you test a code for error

`except AnyErrorName:`
`    print("Division by 0 is not defined!")`
	– statements inside except runs if there's any error

`finally:`
`    print("Close all the resources here!")`
	– statements inside except runs in any case

COLLECTION DATA TYPES

LIST

mutable, ordered (accessing by index is possible)
may contain non-unique elements

my_list = [10, 23, 44, 56, 25, 34]	– creates a list
my_list.append(78)	– adds an element to the end of the list
my_list.insert(3, 4) (index element)	– adds an element to the given index
my_list.pop(2) index	– deletes the element at the given index
my_list.remove(10) element	– deletes the given element
my_list.reverse( )	– reverses the elements of the list
my_list.sort( )	– sorts the list elements in ascending order
my_list.clear( )	– removes all elements
my_list.count(23)	– returns the occurrence number of specified element
my_list.copy( )	– returns a copy of the list
my_list.extend(iterable) list, set, tuple etc.	– adds the elements of any iterable, to the end of the list
my_list.index(10)	– returns the index of the given element

TUPLES

non-mutable, ordered (accessing by index is possible)
may contain non-unique elements

my_tuple = (10, 23, 44, 56, 25, 34)	– creates a tuple
my_tuple.count(44)	– returns the occurrence number of specified element
my_tuple.index(44)	– returns the index of specified value

DICTIONARIES

mutable, non-ordered (accessing by index is not possible)
may contain non-unique values but it's better to use only unique keys

my_dict = {"elon":4098, "anita": 4782, "jurgen": 4139}	– creates a dictionary
my_dict.get("elon") or my_dict["elon"]	– returns value of key "elon"
my_dict["rita"] = 4322	– adds "rita":4322 key-value pair
my_dict.pop("rita")	– removes "rita":4322 key-value pair
my_dict.popitem( )	– removes last inserted key-value pair
my_dict.clear( )	– removes all elements
my_dict.copy( )	– returns a copy of the dictionary
my_dict.items( ), my_dict.keys( ), my_dict.values( )	– returns pairs, keys and values

SETS

mutable, non-ordered (accessing by index is not possible)
may contain only unique elements

my_set = {1.0, "AI", (1,2,3)}	– creates a set
my_set.add("HUB")	– adds an element to set
my_set.remove("AI")	– removes an element from the set
my_set.pop	– removes a random element from the set
my_set.clear( )	– removes all elements
my_set.copy( )	– returns a copy of the set
my_set.difference(your_set)	– returns a set containing the difference between two sets
my_set.intersection(your_set)	– returns a set containing the intersection between two sets

LIST INDEXING AND SLICING

General Notation: numbers[start:stop:step]

Indices	0	1	2	3	4	5
Elements	10	23	44	56	25	34
Negative indices	-6	-5	-4	-3	-2	-1

[a:b] a is inclusive (included) b is exclusive (not included)

INPUT	ACTION	OUTPUT
my_list[1]	– element in the given index	– 23
my_list[2:]	– list after the given index	– [44, 56, 25, 34]
my_list[:5]	– list before the given index	– [10, 23, 44, 56, 25]
my_list[2:5]	– list between given indices	– [44, 56, 25]
my_list[-1]	– last item in the list	– 34
my_list[-2:]	– last 2 items in the list	– [25, 34]
my_list[::-1]	– list in reverse	– [34, 25, 56, 44, 23, 10]
my_list[::-2]	– 1 of every 2 elements in reverse	– [34, 56, 23]
my_list[::2]	– 1 of every 2 elements	– [10, 44, 25]

LOOPS

for iterates through a collection:

```
for element in collection:
    print(element)
```

for

while executes statement when case is True:

```
while num_of_tries < 3:
    password = input("Password: ")
```

while

CONDITIONAL STATEMENTS

IF ELSE

```
if age < 18:
    print("Kid")
elif (age >= 18) and (age < 65):
    print("Adult")
elif age >= 65:
    print("Senior")
else:
    print("Error")
```

if **elif** **elif** **else**

BONUS TECHNIQUES

General Notation: range[start:stop:step]

INPUT	OUTPUT
print(*range(0,12,2))	– 0 2 4 6 8 10
print(*range(8))	– 0 1 2 3 4 5 6 7

LIST COMPREHENSION

[x+1 for x in range (8) if x % 2 == 1]

x+1: output — for x in range (8): collection — if x % 2 == 1: condition

result: [1, 3, 5, 7]

name, age, major = "Andrew", 45, "AI"	– assigns variables at the same time
print(f"Hello my name is {name}, I'm {age} years old.")	– prints variable & text at the same time
int_evenness_check = lambda x : x % 2 == 0	– creates a function in one line
if (n:=len([1, 2 , 3, 4, 5]))>3: print(f'List is too long ({n} elements, expected <= 3)')	– assigns and returns a value in a single expression [walrus operator :=]

FILE OPERATIONS

Modes

r	– opens a file for reading
w	– opens a file for writing
a	– opens a file for appending
b	– opens in binary mode
x	– creates a file
+	– opens a file for updating (reading and writing)

TXT — In case of w (write) and a (append), new file is created if it doesn't exist. — CSV

Examples

file = open("filename.txt", "a")	– opens file for appending
file = open("filename.txt", "rb")	– opens file for reading in binary mode
file_contents = file.read()	– reads the content of the file
file_contents = file.readlines()	– returns a list of lines of file
file.close()	– don't forget to close the file!

Another method for file operations:

with open("filename.txt", "a") as f: file_content = f.read()	– closing process will be done automatically