# System Fundamental Basic Operation

1. To exit python - exit()

A screenshot of a computer program

Description automatically generatedList

1. To check your folder path – pwd

A black screen with white text

Description automatically generated

1. To check the folder path similarly – ls

A screen shot of a computer

Description automatically generated

1. 当在exited的情况下再To enter python – python .
2. **Activate the virtual environment**

A screen shot of a computer program

Description automatically generated

1. How to close the VS Code

A screenshot of a computer

Description automatically generated

1. When already under one folder but want change the folder location

A screenshot of a computer program

Description automatically generated

cd – to change the folder

cd .. – to return to parents

cd .. \XXXXX direct to other folder under the same parent folder.

cd ..\..\XXXX direct to other folder under two parent folders above

A screen shot of a computer code

Description automatically generated

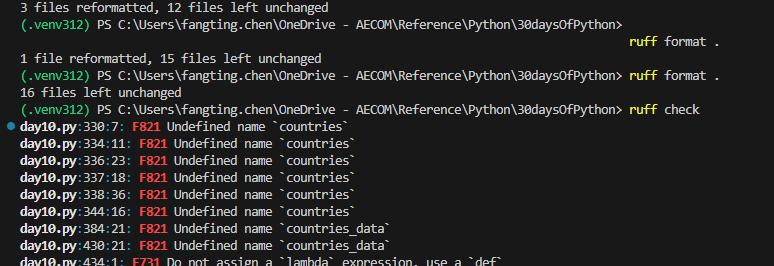
How to activate the virtual environment under the parent folder – also can use .. to locate the folder

A screenshot of a computer program

Description automatically generated

1. Format the script file by ruff format

And use ruff check to see the potential issues



1. Ctrl+L clear the terminal screen
2. For notebook

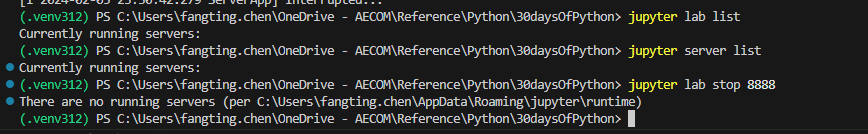
To launch by typing “jupyter lab”

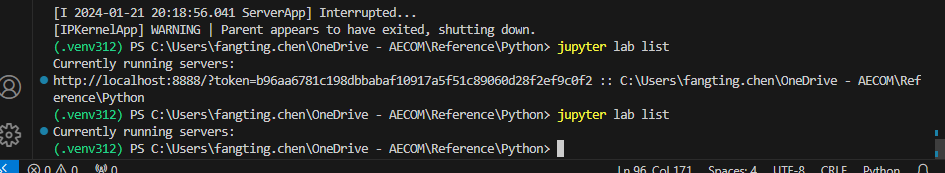


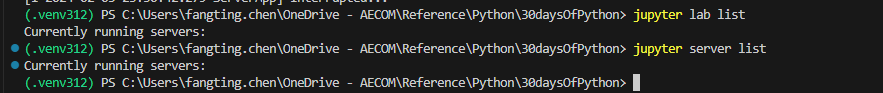
To type Ctrl+c to interrupt the jupyter labjupyter

When running servers still got link: then meaning hvnt been closed fully,

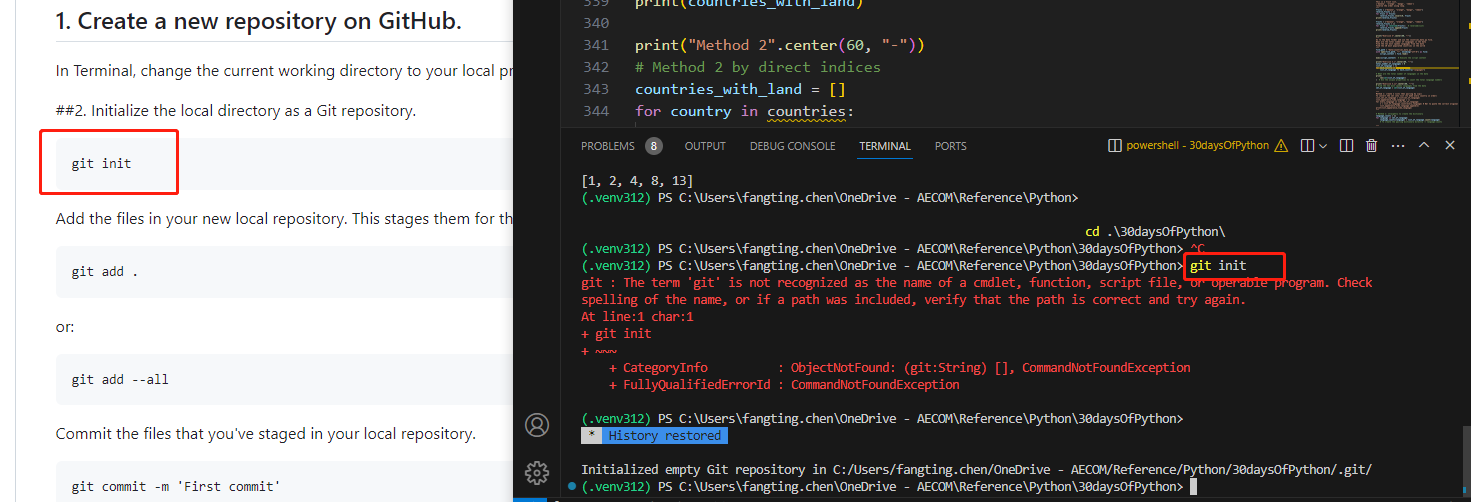
If nothing , meaning closed fully







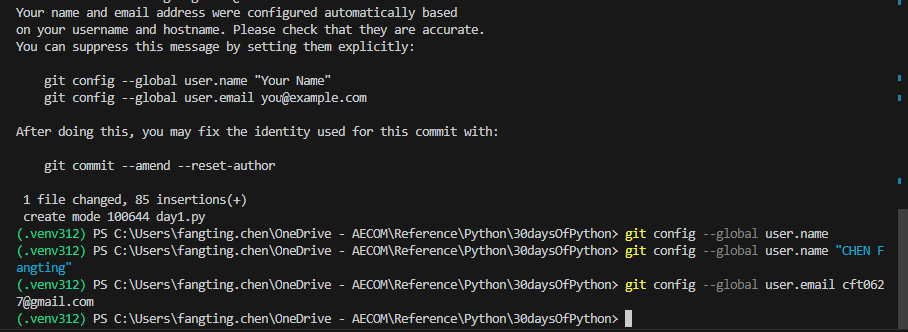
1. How to launch git in the local CS Code
   1. Git init



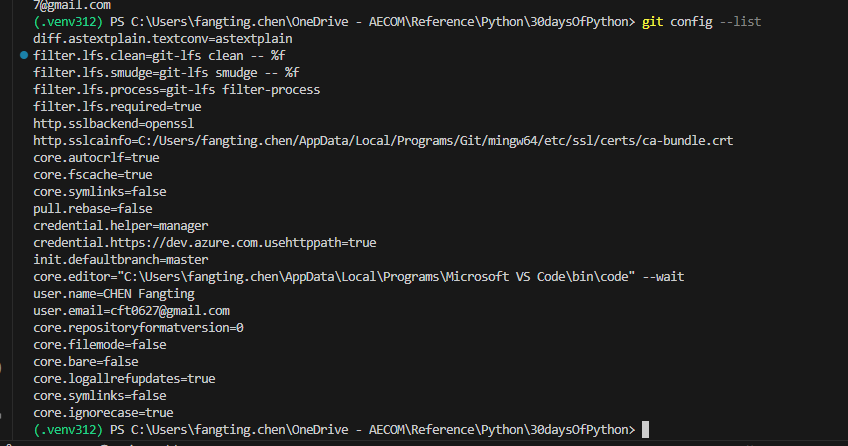
* 1. Git config – to set up your person configuration

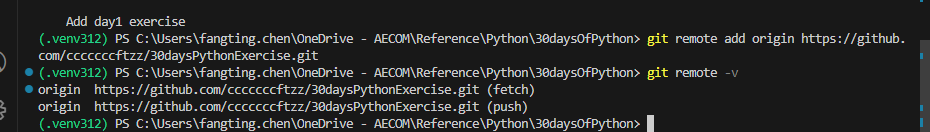
Git config –global user.name

Git config –global user.email

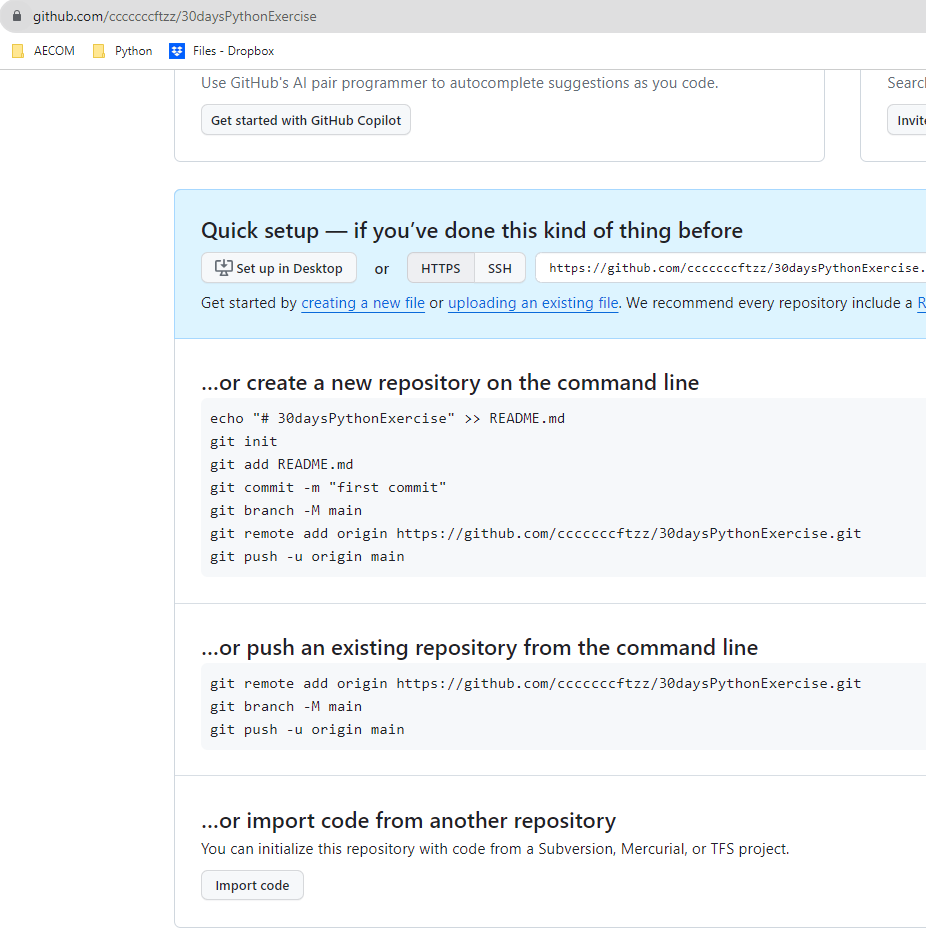


* 1. Check your own configuration info by git config --list

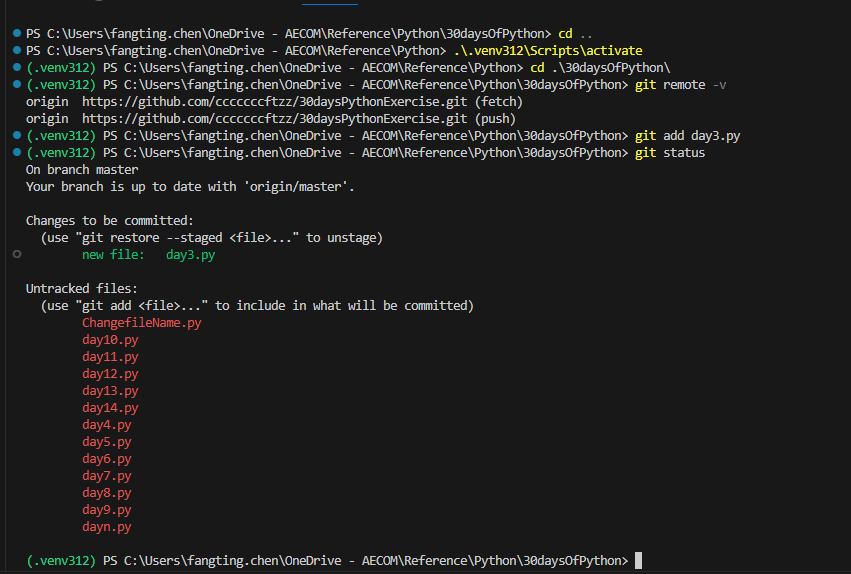




* 1. -u to add on tracking function when pushing the new file

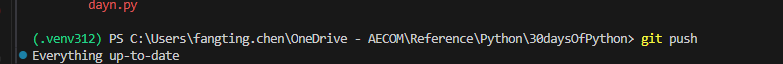


* 1. Check untracked files by git status



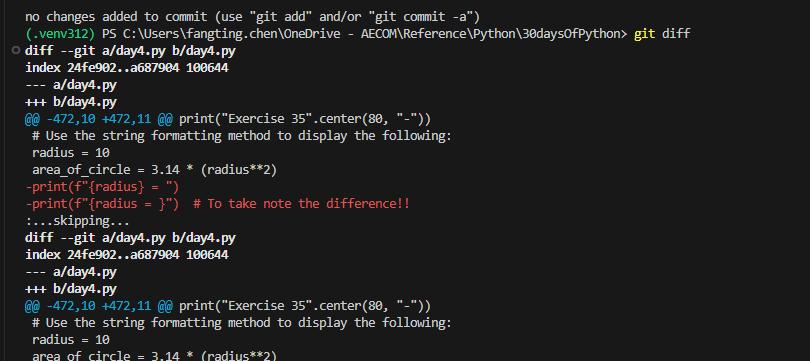
* 1. If I continue the git push without adding the commit,

cloud and my local file are up-to-date then it will show everything up-to-date



* 1. To download the cloud file by typing ‘git pull’
  2. To check whether any difference between the existing cloud and the modified file

By using git diff



* 1. How to push the modified files

Same as adding new file, just adding the modified file again by

Git add XXXX.py

Git commit

Git push

* 1. Can add few commits along with individual files, then push tgt

Git add

Git commit

Git add

Git commit

Then git push tgt

To make sure all the changes under one task will be updated in one time.

* 1. Git clone <url>

to clone others’ repository

# Day 1 of Python - Introduction

## 3 Types of Brackets

1. () are used for Methods function. Or to limit and prioritize a localized computation.

优先算一个局部的运算,通常意义中的()

#print(f"3^2={3\*\*{1+2}}") CANNOT WORK, only can use () to set another scope inside {}

print(f"d) 3^2={3\*\*(1+2)}")

print(f"d2) 3^2.5={3\*\*(1+(0.5+1))}")

1. [ ] are used For List/arrays are used for list, or to locate one argument in a list by index
2. { } are used to set scope to carry out the methods algorithm.

Typically for f-string. Only {} inside the f-string can use for still carrying on the algorithm.

\*\*However, if already inside a f-string {}, then still need another prioritized algorithm, then need () inside {}, as shown above in a)

A screen shot of a computer program

Description automatically generated

Also to take note the f-string {} scope use different

radius = 10

area\_of\_circle = 3.14 \* (radius \*\* 2)

print (f'{radius} = ')

print (f'{radius = }')

Result:

10 =

radius = 10

## Different types of Scripts

### Number

Integer, Float, Complex number

### String

### Booleans

### List, Tuple & Set

List - ordered collection, modifiable, allows to different data type. Allows duplicate members.

['Banana', 10, False, 9.81] # different data types in the list - string, integer, boolean and float

Tuple – ordered, unmodifiable/ immutable collection, allows to different data type, Allows duplicate members.

'Asabeneh', 'Pawel', 'Brook', 'Abraham', 'Lidiya') # Names

Set - unordered, modifiable, allows to different data type. only unique items.

{3.14, 9.81, 2.7} # order is not important in set.

### Dictionary

unordered collection of data in a key value pair format.

No duplicate keys!

Dont forget to give ","

person\_info = {

    'first\_name' : 'Fang Ting',

    'last\_name':'Chen',

    'country':'China',

    'age':'30',

    'is\_married':False

}

## No difference between "" and '' if inside a method to define a variation.

X1 = int(input("X1"))

Y1 = int(input('Y1'))

# Day 2 of Python – Variable

1. The complex number j, got value, but rarely used.
2. max(), min() function can use for few arguments, also can use for one list consisted of few arguments.

print(max(20,30,40,50)) #50

print(min(20,30,40,50)) #20

print(min([20,30,40,50])) #20

print(sum([20,50])) #70

1. Invalid Variable Name: "-". Start with Number, Special figure like @$%
2. When quoting a variable, don’t put any ‘ ‘, otherwise it becomes a string.

print(type(person\_info))

print(type('person\_info'))

1. If only wanna insert a new line, just give the following two ways,

Method 1 – Just give one line

print('\n') #how to insert a new line when printing the multiple variables.

Method 2 – Just to print multiple variables

first\_name = 'Fang Ting'

last\_name = 'Chen'

space = ' '

full\_name = first\_name + space + last\_name

print (last\_name +'\n'+ full\_name)

1. To format a number type with specified digits in f-string.

print(f'{num\_flo:.4f}')

But to take note the location to add the :.4f

print (f'{a / b = :.2f}')

print (f'{a / b = }:.2f')

Result:

a / b = 1.33

a / b = 1.3333333333333333:.2f

1. To give a break line with the title

print('Exercise Level 2'.center(50, "-")) #50 is the total specified string length

print('Exercise Level 2.7'.ljust(35, "~"))

print('Exercise Level 2.7'.rjust(35, "~"))

1. “%”, “//”

% to get the remainer

// : floor division

1. Importing a library, such as math which involves few common math constants.

radius\_circle = int(input('Please enter the radius of the circle = '))

import math # math.pi

area\_of\_circle = math.pi \* (radius\_circle \*\* 2)

print(f'{area\_of\_circle = :.2f}')

# Day 3 of Python – Operator

1. #|,^,<<=,>>= for bit operation no need to learn first

A screenshot of a cell phone

Description automatically generated

1. True == 1 ONLY

False == 0

1. When finding a substring inside string, don’t give [], Otherwise it works as an argument as List

print('on' in ['python'] and 'on' in ['dragon'])

'''

Typical wrong mistake due to wrong type of the argument

when it is in [] meaning it will be one list

'on' is not inside as it only one argument 'python' and 'dragon'

'''

print('on' in 'python' and 'on' in 'dragon')

rmr must type all the when just starting the learning!! Most of the time you wont spot those easy mistakes by typing them yourselves.

1. No need to assign bool() when the sentence is already a judgment

whether\_number\_three\_even = bool (number\_three % 2 == 0) No need to assign bool() because already a bool type

1. The use of “if … else “

print('To juedge a even number')

number\_three = int (input ('To give a number to judge it is even number or odd number '))

#whether\_number\_three\_even = bool (number\_three % 2 == 0)  No need to assign bool() cuz already a bool type

whether\_number\_three\_even = number\_three % 2 == 0

print (f'The number {number\_three} {"is" if whether\_number\_three\_even is True else "is not"} is an even number ')

1. All the input are the String type (even it is a figure!!) Don’t forget to convert the input before computing them

working\_hours = int(input('Enter Working Hours'))  #Dont forget to convert the input to Int!!!

rate\_per\_hour = int(input ('Enter rate per hour'))

pay\_of\_person = working\_hours \* rate\_per\_hour

print (f'Your weekly earning is = {pay\_of\_person}')

# Day 4 of Python – String

1. \t: Tab means(8 spaces), if wanna tab in more , just give double : \t\t
2. To print for \n, \t, \\, \', \"

print ('Practice for "\\n", \\t, \\\, \\\', \\\"'.center(80, "-"))

#Result:-

-----------------------Practice for "\n", \t, \\, \', \"------------------------

1. To assign every character inside the string

language = 'Python'

a,b,c,d,e,f = language

1. Reversing Strings by using [::-1] Start&End&Step

"::" means that it will consider the entire string

the negative step '-1 indicates that the slicing should be done in reverse. When step is 2, meaning to print start from 0: 0,2,4

language = 'Python'

print (language)

pto = language[0:6:2] # 0~5 but step is 2, meaning to print 0,2,4

Result Pto

1. When to check a substring in a list with one argument, cannot find. So pay attention on the type of data

challenge = 'thirty days of python'

print (challenge.count('y',7,14)) #1

challenge = ['thirty days of python']

print (challenge.count('of')) #0

1. expandtabs() function to extend for longer strings

tab\_list = 'Name\tAge\tCountry\tCity'

tab\_list\_one = 'Asabeneh\t250\tFinland\tHelsinki'

print (tab\_list.expandtabs(12))

print (tab\_list\_one.expandtabs(12))

Result:

----------------------------------Exercise 34-----------------------------------

Name Age Country City

Asabeneh 250 Finland Helsinki

1. find() VS index() , rfind() VS rindex()

In Python, both find() and index() are methods used to search for a substring within a string and return the lowest index of the substring if found.

key differences in their behavior:

Return Value:

find():If the substring is not found, it returns -1. you are okay with a result of -1 if the substring is not found.

index():If the substring is not found, it raises a ValueError. you want to handle the case when the substring is not found by catching the ValueError

There is one method to handle with ValueError

sentence = "Codin For All"

try:

    index = sentence.index("Coding")

    print("The substring 'Coding' is present at index:", index)

except ValueError as e:

    print("The substring 'Coding' is not found in the string.")

rfind() and rindex() similarly - search for the last occurrence of a substring in a string and return the highest index of the substring if found.

challenge = 'thirty days of python'

print (challenge.rfind('y')) #16, find the index of last occurrence of substring

print (challenge.rfind('th'))#to give 't' index which is 17

sub\_string = 'da'

print (challenge.index(sub\_string)) #To return the index of "d", 7

print (challenge.index(sub\_string,0,12)) #If cannot find any between (0,12), it will return error

\* When the substring is not completely included in the specified length, so meaning cannot find. See sample below

challenge = 'thirty days of python'

sub\_string = 'da'

#print (challenge.rindex(sub\_string,0,8)) #error. "a" index is 8, but the substring is not completely included in the specified length, so cannot find still

print (challenge.rindex(sub\_string,0,9)) #7

\*\*capital also different when specifying the string to be found

1. Checking alphabet, number and digits

isalnum(): Checks alphanumeric character (a-z and A-Z and 0-9)

isalpha(): Checks if all string elements are alphabet characters (a-z and A-Z)

isdecimal(): Checks if all characters in a string are decimal (0-9)

\*Don’t forget to add () after.isdecimal

isdigit(): Checks if all characters in a string are numbers (0-9 and some other unicode characters for numbers)

challenge = '\u00B2'

print(challenge.isdigit())   # True ??? what is this

isnumeric(): Checks if all characters in a string are numbers or number related (just like isdigit(), just accepts more symbols, like ½)

num = '\u00BD' # ½

print(num.isnumeric()) # True ??? what is this

\*\*When using the isalnum() to check if alphanumeric number. Space is not alphanumeric

challenge = 'thirty days of python'

print(challenge.isalnum()) # False, space is not an alphanumeric character

1. join(), strip(), split(), replace()

join(): Returns a concatenated string

web\_tech = ['HTML', 'CSS', 'JavaScript', 'React']

result = ' '.join(web\_tech) # Inserting a space when joining the list member

#HTML CSS JavaScript React

strip(): Removes all given characters starting from the beginning and end of the string

challenge = 'thirty days of pythooonnn'

print(challenge.strip('noth')) # 'irty days of py'

replace(): Replaces substring with a given string

challenge = 'thirty days of python'

print(challenge.replace('python', 'coding')) # 'thirty days of coding'

split(): Splits the string, using given string or space as a separator

challenge = 'thirty days of python'

print(challenge.split()) # ['thirty', 'days', 'of', 'python']

1. Typical method to create an abbreviation for a string

phrase = "Python For Everyone"

acronym = ''.join(word[0].upper() for word in phrase.split())

#Don't understand why for loop to be returned everytime when using join function.

!! still are not familiar with the for loop use. To check later

# Day 5 of Python - List (Modifiable, ordered)

1. Create empty list

Lst = list()

OR

Lst = []

1. Accessing list item

first\_fruit = fruits[0] # we are accessing the first item using its index

1. Modify / adding/ inserting/ checking / removing / deleting/ clearing

Modifying: fruits[0] = 'avocado'

Checking: does\_exist = 'banana' in fruits  
**append ():** lst.append(item)

complete\_list\_a\_b = a.append(b) .append() is modifying the list in the original list!!! Cannot run out this row

a.append(b)

append is only workable for adding one item into the original list from the last index

however extend is to combine two ‘list’ types tgt  
Inserting: lst.insert(index,item)  
remove(): lst.remove(item)

Pop(): removes the specified index:

lst.pop(index)

OR

lst.pop() #last item

Deleting:

del lst[index]

Or del lst

Clear: Lst.clear() # only to clear the content but keep the list variable

Copy: lst\_copy = lst.copy()

1. Joining

List3= list1 + list2

list1.extend(list2) #Adding list 2 into list 1

‘+=’ similar to extend, meaning combine two lists tgt

1. Counting items in a list

lst.count (item)

1. Finding index of an item

fruits = ['banana', 'orange', 'mango', 'lemon']  
print(fruits.index('orange')) # 1

1. Lst.reverse()

ages = [22, 19, 24, 25, 26, 24, 25, 24]  
ages.reverse()  
print(ages) # [24, 25, 24, 26, 25, 24, 19, 22]

1. Sort() & sorted()

Sort(): sorting the list itslef

lst.sort() # ascending  
lst.sort(reverse=True) # descending

Sorted(): returns the ordered list without modifying the original list

The properties of sort:

print (it\_companies)

sort\_it\_companies = it\_companies.sort()

print (sort\_it\_companies)

print (it\_companies.sort())

'''

Cannot work, after the original list is modified,

The sort() is not a new sorted list and no value to be assigned to the new list.

Same as using print () function

The correct way to sort() is as shown below

'''

it\_companies.sort()

print(it\_companies)

1. Range [2,-2] also can work

it\_companies = ['Facebook', 'Google', 'Microsoft', 'Apple', 'IBM', 'Oracle', 'Amazon']

it\_companies = it\_companies[2:-2]

# Day 6 of Python - Tuple (Unmodifiable, ordered)

1. tuple(): to create an empty tuple
2. count(): to count the number of a specified item in a tuple
3. Accesing tuple items: similar as list

first\_item = tpl[0]

1. index(): to find the index of a specified item in a tuple
2. join two or more tuples and to create a new tuple

fruits = ('banana', 'orange', 'mango', 'lemon')  
vegetables = ('Tomato', 'Potato', 'Cabbage','Onion', 'Carrot')  
fruits\_and\_vegetables = fruits + vegetables

1. Since tuple is not immutable

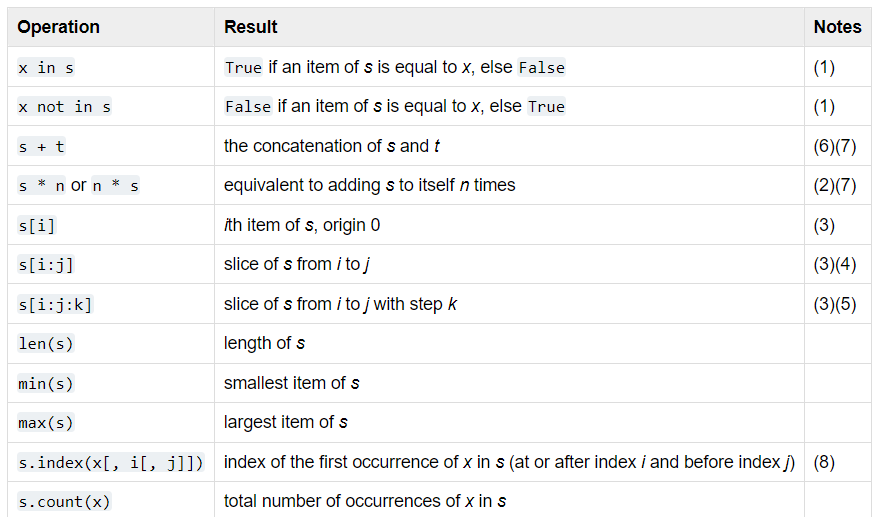
fruits[0] = 'apple' # TypeError: 'tuple' object does not support item assignment.

1. Changing Tuple to list

family\_list = list (siblings) # Change Turple to list

family\_tuple = tuple(family\_list) # Change list to Turple

1. The common sequence operation all applicable to tuple, See the official documentation from python



Including “+”: s + t, the concatenation of s and t

When using “+” for two tuples, the original tuple is not modified, instead a new tuple is created.

Example 1: Normal for loop

result\_list = []

for i in range(11):

tuple\_values = (i,)

for power in range(6):

tuple\_values += (i\*\*power,) # Here every loop will create a new tuple which like “adding” to the previous existing tuple, it is not modifying the existing tuple

(i\*\*power,) ‘,’ here means creates a tuple with a single element, which is the result of i\*\*power and then concentrates. It does not mean any real ‘,’ format

result\_list.append(tuple\_values)

print(result\_list)

Result:

[(0, 1, 0, 0, 0, 0, 0),

(1, 1, 1, 1, 1, 1, 1),

(2, 1, 2, 4, 8, 16, 32),

(3, 1, 3, 9, 27, 81, 243),

(4, 1, 4, 16, 64, 256, 1024),

(5, 1, 5, 25, 125, 625, 3125),

(6, 1, 6, 36, 216, 1296, 7776),

(7, 1, 7, 49, 343, 2401, 16807),

(8, 1, 8, 64, 512, 4096, 32768),

(9, 1, 9, 81, 729, 6561, 59049),

(10, 1, 10, 100, 1000, 10000, 100000)]

Example 2: list comprehensive

output = [(i,) + tuple((i \*\* power) for power in range(0,6)) for i in range(0,11) ]

Result are the same as example 1

A new tuple is created in every iteration of the outer loop.

The expression (i,) + tuple(i\*\*power for power in range(6)) creates a new tuple by concatenating the tuple (i,) with the tuple generated by the generator expression.

Therefore, even in the list comprehension, you are creating new tuples at each step.

1. To create a tuple but start with a given member (a,)

(1,) is the way to create a tuple with one member

* If (1) meaning it is just an integer.

output = [(i,) + tuple((i \*\* power) for power in range(0,6)) for i in range(0,11) ]

Here defined the first member in tuple is i

# Day 7 of Python - Set (Unordered, no duplicate member)

\*数学中 是集合的概念.

1. 集合中元素的特性

确定性

给定一个集合，任给一个元素，该[元素](https://baike.baidu.com/item/%E5%85%83%E7%B4%A0/9563210?fromModule=lemma_inlink)或者属于或者不属于该集合，二者必居其一，不允许有模棱两可的情况出现。

互异性

一个集合中，任何两个元素都认为是不相同的，即每个元素只能出现一次。有时需要对同一元素出现多次的情形进行刻画，可以使用[多重集](https://baike.baidu.com/item/%E5%A4%9A%E9%87%8D%E9%9B%86/0?fromModule=lemma_inlink)，其中的元素允许出现多次。

无序性

一个集合中，每个元素的地位都是相同的，元素之间是无序的。集合上可以定义序关系，定义了序关系后，元素之间就可以按照序关系排序。但就集合本身的特性而言，元素之间没有必然的序。

1. 元素与集合的关系

属于

如果元素a在集合A中，就说a属于A，记作a∈A。

不属于

如果元素a不在集合A中，就说a不属于A，记作a∉A。

1. Creating an empty set

fruits = set () #Different from Tuple tuple()

fruits = {'banana', 'orange', 'mango', 'lemon'}

1. Accessing Items in a Set

We use loops to access items.

1. Converting List to Set: set()

fruits = ['banana','orange','mango','lemon']

fruits\_set = set(fruits)

1. Adding Items to a Set

add : ()

fruits = {'banana', 'orange', 'mango', 'lemon'}  
fruits.add('lime')

update():

1. take one list argument

st = {'item1', 'item2', 'item3', 'item4'}  
st.update(['item5','item6','item7'])

it\_companies = {'Facebook', 'Google', 'Microsoft', 'Apple', 'IBM', 'Oracle', 'Amazon'}

it\_companies.update('Lazada','Tiktok','Shoppee') #Typical Mistakes, update() if including few arguments, It will take as **individual character**

# {'h', 'S', 'Apple', 'Google', 'a', 'z', 'Oracle', 'IBM', 'Amazon', 'L', 'p', 'i', 't', 'd', 'Twitter', 'Microsoft', 'Facebook', 'o', 'e', 'k', 'T'}

it\_companies = {'Facebook', 'Google', 'Microsoft', 'Apple', 'IBM', 'Oracle', 'Amazon'}

it\_companies.update(['Lazada','Tiktok','Shoppee'])

# {'Apple', 'Google', 'Tiktok', 'Oracle', 'IBM', 'Amazon', 'Shoppee', 'Microsoft', 'Facebook', 'Lazada'}

1. Take one set (as one argument) into another set

fruits.update (vegetables) #here vegetables also only considered as one argument

print (fruits) # To add in vegtables into set fruits

union():

fruits = {'banana','orange','mango','lemon'}

vegetables = {'tomato','potato','cabbage'}

print (fruits.union(vegetables))

**Difference between .union() and .update()**

# Union is for creating a new set, didn’t change the original two sets

1. Remove/ pop / del/ clear

remove(): lst.remove('item’)

disgard(): it\_companies.discard('Lazada') # disgard() will not pop out the Error even if the result didnt find in the set

Pop(): remove a random item from the set:

lst.pop()

#in set, remove a random item due to unordered; in list, it means removing the last item

Deleting:

del lst

Clear: Lst.clear() # only to clear the content but keep the list variable

Copy: lst\_copy = lst.copy()

1. Intersection items (交集）

whole\_numbers = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}  
even\_numbers = {0, 2, 4, 6, 8, 10}  
whole\_numbers.intersection(even\_numbers) # {0, 2, 4, 6, 8, 10}

1. Subset & superset （子集和并集）

whole\_numbers = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

even\_numbers = {0, 2, 4, 6, 8, 10}

whole\_numbers.issubset(even\_numbers) #A.issubset(B) -> A是不是B得子集

# False, because it is a super set

whole\_numbers.issuperset(even\_numbers) #A.issuperet(B) -> A是不是B得并集

#True

1. Difference between two sets （补集）

whole\_numbers = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}  
even\_numbers = {0, 2, 4, 6, 8, 10}  
whole\_numbers.difference(even\_numbers) # A.difference(B) -> (A-B) or (A\B)

#{1, 3, 5, 7, 9}

1. Symmetric Difference Between Two Sets

whole\_numbers = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}  
some\_numbers = {1, 2, 3, 4, 5}  
whole\_numbers.symmetric\_difference(some\_numbers) # {0, 6, 7, 8, 9, 10} # it means (A\B)∪(B\A)

1. Joining set (是不是有交集）

even\_numbers = {0, 2, 4 ,6, 8}  
even\_numbers = {1, 3, 5, 7, 9}  
even\_numbers.isdisjoint(odd\_numbers) # True, because no common item

# Day 8 of Python - Dictionary (modifiable, ordered, unique keys)

1. Creating an empty dictionary

dct = {} #Creating an empty dictionary

fruits = set () #Creating an empty set

1. Dictionary can store list and sub-dictionary
2. To count the nos of keys. By len()
3. Accessing an item in dictionary by key
   1. By index

print (dct['address']) # f cannot get the key in the dictionary, by using index it will return error

\*To index one sub-dictionary by key and index for the value

print (dct['skills'][2]) # To give index only after using dct['key'][index]

* 1. get()

print (dct.get('city')) # By using get, even the key is not available in the dictionary, it returns None

1. Adding / modifying
2. Adding:

person['job\_title'] = 'Instructor'  
modifying:

dct = {'key1':'value1', 'key2':'value2', 'key3':'value3', 'key4':'value4'}

dct['key1'] = 'value-one'

1. checking key in dictionary

print('key2' in dct) # True

1. Removing Key and Value Pairs from a Dictionary

person.pop('first\_name') # Removes the firstname item  
person.popitem() # Removes the last item  
del person['is\_married'] # Removes the is\_married item

1. Changing Dictionary to a List of Items by item()

dct = {'key1':'value1', 'key2':'value2', 'key3':'value3', 'key4':'value4'}

print(dct.items()) # dict\_items([('key1', 'value1'), ('key2', 'value2'), ('key3', 'value3'), ('key4', 'value4')])

The result is a list of Mutiple tuples

1. Clear / delete/ copy

print(dct.clear()) # None

del dct

dct\_copy = dct.copy() # {'key1':'value1', 'key2':'value2', 'key3':'value3', 'key4':'value4'}

1. Getting Dictionary Keys as a List

keys = dct.keys() # dict\_keys(['key1', 'key2', 'key3', 'key4'])

1. Getting Dictionary Values as a List

values = dct.values() # dict\_values(['value1', 'value2', 'value3', 'value4'])

# Day 9 – Conditionals

1. If few syntax:

|  |  |  |  |
| --- | --- | --- | --- |
| if: | if: | if: | If: |
|  | else: | elif: | elif: |
|  |  | else: | elif: |
|  |  |  | else: |

# Day 10 – While & for Loops

1. **The difference between for and while (Thru application)**
2. **Initialization and Condition:**

For Loop: It typically has three parts - initialization, condition, and increment/decrement. The loop continues to execute as long as the condition is true.

python

Copy code

for i in range(5): # Initialization, Condition, and Increment are combined

While Loop: It has only a condition. The loop continues to execute as long as the condition is true.

So here, don’t forget to update the iteration value in while loop, bc it will not updated within the loop without coding. Otherwise it will lead to an infinite repetition of the same condition

1. **Structure:**

For Loop: It is often used when the number of iterations is known beforehand.

While Loop: It is used when the number of iterations is not known beforehand, and the loop should continue until a certain condition is met.

1. **Use Cases:**

For Loop: Ideal when you know the number of times you want to iterate, such as iterating over elements in a list or running a loop a specific number of times.

While Loop: Suitable when you want to repeat a block of code until a certain condition is no longer true, and the number of iterations is not predetermined.

1. **While syntax:**

|  |  |  |  |
| --- | --- | --- | --- |
| while : | while: | while: | while: |
|  | else: | if: | if: |
|  |  | break | continue |
|  |  | else: | else: |
|  |  | \*exit the loop immediately including else code when reach break row | \*skip the rows below inside the loop, continue to next iteration |

**In the while loop when there is no break condition: No difference without or without ‘else:’**

count = 0

while count < 6:

    print (count) #0,1,2,3,4,5

    count = count + 1 #1,2,3,4,5,6

else:

    print (count) # 6

count = 0

while count < 6:

    print (count) #0,1,2,3,4,5

    count = count + 1 #1,2,3,4,5,6

print (count) # 6

**While + if break + else, there is a difference by giving or not giving ‘else:’**

Since the code break, so it will not run else.

count = 0

while count < 6:

    if count == 5:

        break # Here once the loop break, it wont execute the else clause, either.

    print(count) #0,1,2,3,4

    count = count + 1 #1,2,3,4,5

print(count) #5

count = 0

while count < 6:

    if count == 5:

        break

    print(count) #0,1,2,3,4

    count = count + 1 #1,2,3,4,5

else:

    print(count) #will not run this row

**However, The use of ‘else’ in a while loop is a matter of coding style and preference.**

1. **For syntax:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| for: | for: | for: | for: | for: | for: |
|  | else: | if: | if: | for: | pass |
|  |  | break | continue |  |  |
|  |  | Exit the loop immediately when reach break row | Skip the rows below inside the loop, continue to next iteration |  | For future statements usually.  Don’t execute any code here |

1. All the loop result are automatically give a new line
2. When using for loop for set, the loop result are not in order
3. To access a file under the same folder

file\_path = 'data/countries.py'

with open(file\_path, 'r') as file:

    script\_content = file.read()

exec(script\_content) # Execute the script content

file\_path = "data/countries\_data.py"

with open(file\_path, "r", encoding="utf-8") as file:

script\_content = file.read()

exec(script\_content)  # Read the file

1. When using loop to access the list each member:

We prefer to using the direct indices.

countries\_with\_land = []

for country in countries:

    if 'land' in country:

        countries\_with\_land.append (country)

print (countries\_with\_land)

1. Common way to count a pair of data related by using dictionary:

language\_counts = {} #Dont forget to create and define an empty dic

for language in set\_of\_language:

    language\_counts[language] = list\_of\_language.count(language)

    # To create the pairing dictionary directly -> language:counts

1. How to sort a list of tuple by using the specified index member in each of tuple

list\_appearance\_time\_language = list(language\_counts.items())

list\_appearance\_time\_language.sort(key=lambda x: x[1], reverse=True)

# To sort by the second element in the tuple, and descending

print("Top 10 most spoken languages:")

for language, count in list\_appearance\_time\_language[:10]:

    print(f"{language}: {count}")

lambda : is one unnamed function name (when people don’t define that function specifically), usually only quoted for the sorting

“lambda x” meaning to do a unnamed function to x , x stands for the each tuple inside the list

X[1] refers to the second member in each of the tuple, i.e. 2,5, by descending

(‘Chinese’:2, ‘English’:5)

1. Built in function ‘sort()’:

sort(\*, key=None, reverse=False)

* change the original list

Sorted() :

* returns the ordered list without modifying the original list

student\_tuples = [

**...**  ('john', 'A', 15),

**...**  ('jane', 'B', 12),

**...**  ('dave', 'B', 10),

**...** ]

**>>>** sorted(student\_tuples, key=**lambda** student: student[2]) *# sort by age*

[('dave', 'B', 10), ('jane', 'B', 12), ('john', 'A', 15)]

* Sorts are guaranteed to be stable. That means that when multiple records have the same key, their original order is preserved.

data = [('red', 1), ('blue', 1), ('red', 2), ('blue', 2)]

**>>>** sorted(data, key=itemgetter(0))

[('blue', 1), ('blue', 2), ('red', 1), ('red', 2)]

* If using Sorted({dic{}} for dictionary, it will ONLY consider the keys and return a sorted list of those keys

print (sorted({1: 'D', 2: 'B', 13: 'B', 8: 'E', 4: 'A'})) #[1, 2, 4, 8, 13]

* If wanna print out the key,value both by sorting the key

my\_dict = {1: 'D', 2: 'B', 13: 'B', 8: 'E', 4: 'A'}

# Using a list comprehension to create a list of key-value pairs sorted by keys

sorted\_items = sorted(my\_dict.items())

# Printing the sorted key-value pairs

for key, value in sorted\_items:

print(key, value)

# Day 11 – function

1. Define a function without parameter also need to give ()

def add\_two\_nums():

…

add\_two\_nums() #Run the code

1. If have more than one parameter, if passing the arguments with key and value, then order does not matter.

def add\_two\_numbers (num1, num2):

total = num1 + num2

print(total)

print(add\_two\_numbers(num2 = 3, num1 = 2)) # Order does not matter

1. Function with default values.

def calculate\_age (birth\_year,current\_year = 2021):

age = current\_year - birth\_year

return age;

print('Age: ', calculate\_age(1821))

def weight\_of\_object (mass, gravity = 9.81):

weight = str(mass \* gravity)+ ' N' # the value has to be changed to string first

return weight

print('Weight of an object in Newtons: ', weight\_of\_object(100)) # 9.81 - average gravity on Earth's surface

print('Weight of an object in Newtons: ', weight\_of\_object(100, 1.62)) # gravity on the surface of the Moon

1. Arbitrary number of arguments :

def sum\_all\_nums(\*nums):

def sum\_all\_nums(\*nums):

total = 0

for num in nums:

total += num # same as total = total + num

return total

print(sum\_all\_nums(2, 3, 5)) # 10

1. When there is no return value, and the printing function inside the function, then just run the function without printing syntax()

#Method 1: print the function while return value still None

def generate\_groups (team,\*args):

    print(team)

    for i in args:

        print(i)

print(generate\_groups('Team-1','Asabeneh','Brook','David','Eyob'))

#To take note there is a 'None' value when running the function done.

Result:

Team-1

Asabeneh

Brook

David

Eyob

None

#Method 2: Directly run the function and no need print return value

generate\_groups('Team-1','Asabeneh','Brook','David','Eyob')

Result:

Team-1

Asabeneh

Brook

David

Eyob

1. Pass the function as parameter:

def cube\_number (n):

    return n \*\* n

def do\_something(f, x):

    return f(x)

print(do\_something(cube\_number, 3)) # 27

1. Reminder of the digits format

print (f'{area\_of\_circle(1):.2f}') # Need keep the :.2f format with the variable

1. To use isinstance to check the format type of an object

        if isinstance(num, (int, float)): # isinstance to check the object are certain format.

        # isinstance(object, classinfo) : a checking object, certain format type

1. Raise + Error exception warning – can tell user whats the issue

        raise ValueError ("The x coordinates for the two points should be different!")

        # Raise + error exception warning to give the warning to user

1. To check a variable empty or not

If ‘value’ is empty, then 'value' itself means False

Then 'not value' means True

So not value returns True and it is one of the types mentioned

def is\_empty(value):

    if value is None:

        return True

    elif isinstance(value, str) and value.strip() == '':

        return True

    elif isinstance(value, (list, set, tuple, dict)) and not value:

        print(not value) #True

        '''

        If value is empty, then 'value' itselves means False

        Then 'not value' not value means True

        So not value returns True and it is one of the types mentioned

        '''

        return True

    else:

        return False

1. The function, module, calss, instance, class method, attribute relationship

**Module:**

A module can contain functions, which are blocks of reusable code.

**Class:**

In addition to attributes and methods, a class can also have class-level functions, known as class methods. Class methods are associated with the class itself rather than instances.

**Instance:**

A class can have instance methods, which are functions defined within the class and are intended to be called on instances of that class.

Functions can also be used outside the class to perform various operations.

**Class Method:**

Class methods are functions associated with a class. They take the class itself as their first parameter (cls) and can be called on the class rather than instances.

**Attribute:**

Functions can be used to define methods within a class, which are essentially functions associated with instances or the class itself.

Now, let's summarize the **relationships**:

Module and Function:

A module can contain functions, providing a way to organize and encapsulate related functionality.

Functions within a module can be called from other modules or scripts.

Class and Function:

A class can have methods, which are functions associated with instances or the class itself.

These methods define the behavior of instances of the class.

Instance and Function:

Functions can be used as instance methods, defining operations that can be performed on individual instances.

Class Method and Function:

Class methods are functions associated with the class itself.

They are defined using the @classmethod decorator and take the class as their first parameter (cls).

Attribute and Function:

Functions can be used to define methods within a class, which can be seen as functions associated with attributes.

# Day 12 – Module

* + - 1. Module usually save in individual .py file.
      2. To import an entire file, just by using importing and the file name only

But when need quote any function need giving the filename in front as well. See sample below

A screenshot of a computer screen

Description automatically generated

* + - 1. To import a function under one module:

from mymodule import generate\_full\_name, sum\_two\_nums, person, gravity

Or also can rename them:

from mymodule import generate\_full\_name as fullname, sum\_two\_nums as total, person as p, gravity as g

* + - 1. Built-in module

**OS Module**

# import the module

import os

# Creating a directory

os.mkdir('directory\_name')

# Changing the current directory

os.chdir('path')

# Getting current working directory

os.getcwd()

# Removing directory

os.rmdir()

**Statistics Module**

from statistics import \* # importing all the statistics modules

ages = [20, 20, 4, 24, 25, 22, 26, 20, 23, 22, 26]

print(mean(ages)) # ~22.9

print(median(ages)) # 23

print(mode(ages)) # 20

print(stdev(ages)) # ~2.3

**Math Module**

import math

print(math.pi) # 3.141592653589793, pi constant

print(math.sqrt(2)) # 1.4142135623730951, square root

print(math.pow(2, 3)) # 8.0, exponential function

print(math.floor(9.81)) # 9, rounding to the lowest

print(math.ceil(9.81)) # 10, rounding to the highest

print(math.log10(100)) # 2, logarithm with 10 as base

**String Module**

import string

print(string.ascii\_letters) # abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ

print(string.digits) # 0123456789

print(string.punctuation) # !"#$%&'()\*+,-./:;<=>?@[\]^\_`{|}~

print(string.ascii\_lowercase) # abcdefghijklmnopqrstuvwxyz

print(string.ascii\_uppercase) # ABCDEFGHIJKLMNOPQRSTUVWXYZ

print(string.ascii\_letters)

# abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ

print(string.digits) # 0123456789

print(string.punctuation)

**Random Module**

from random import random, randint

print(random()) # it doesn't take any arguments; it returns a value between 0 and 0.9999

print(randint(5, 20)) # it returns a random integer number between [5, 20] inclusive

**random shuffle**

Shuffle the sequence x in place.

def shuffle\_list(lst):

    random.shuffle(lst)

    return lst

print(shuffle\_list(["ft", "pig", "fat"]))

result:

['fat', 'pig', 'ft']

* + - 1. To import all the function in math module we can use \* .

from math import \*

print(pi) # 3.141592653589793, pi constant

print(sqrt(2)) # 1.4142135623730951, square root

print(pow(2, 3)) # 8.0, exponential

print(floor(9.81)) # 9, rounding to the lowest

print(ceil(9.81)) # 10, rounding to the highest

print(math.log10(100)) # 2

* + - 1. **Import a function and rename it**

from math import pi as PI

print(PI) # 3.141592653589793

* + - 1. **Random module - random.choices**

random.**choices**(*population*, *weights=None*, *\**, *cum\_weights=None*, *k=1*)

Return a k sized list of elements chosen from the population with replacement. If the population is empty, raises IndexError.

def generate\_user\_id():

    random\_id = "".join(random.choices(string.digits + string.ascii\_letters, k=6))

    # Dont forget k=6 is inside the random function , dont put wrong ()

return random\_id

result:

qU7PSg

* + - 1. **f'' String can be used as return result**

def rgb\_color\_gen():

    return f"rgb({randint(0,256)}, {randint(0,256)}, {randint(0,256)})"

    # f'' String can be used as return result

# Day 13 – List comprehension

* + - 1. Syntax

# syntax

[i for i in iterable if expression]

Combined with if expression:

even\_numbers = [i for i in range(21) if i % 2 == 0]

# to generate even numbers list in range 0 to 21

print(even\_numbers)

# [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20]

list\_of\_lists = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

flattened\_list = [ number for row in list\_of\_lists for number in row]

print(flattened\_list)

# [1, 2, 3, 4, 5, 6, 7, 8, 9]

* + - 1. Lambda function

# syntax

x = lambda param1, param2, param3: param1 + param2 + param2

print(x(arg1, arg2, arg3))

see example below:

slope\_between\_two\_points = lambda a, b: (b[1] - a[1]) / (b[0] - a[0])

print(slope\_between\_two\_points((3, 6), (8, 2)))

* + - 1. 3layers of flatten, sample as below:

list\_of\_lists = [[[1, 2, 3]], [[4, 5, 6]], [[7, 8, 9]]]

flattened\_list = [

    item for sublist in list\_of\_lists for subsublist in sublist for item in subsublist

]

# Result [1, 2, 3, 4, 5, 6, 7, 8, 9]

* + - 1. When creating a new list with sublist/dictionary by using list comprehension

Can use the format variable.

for country, capital in sublist. See below

countries = [[("Finland", "Helsinki")], [("Sweden", "Stockholm")], [("Norway", "Oslo")]]

flattened\_list = [

    [country.upper(), country[:3].upper(), capital.upper()]

    for sublist in countries

    for country, capital in sublist

]

# Result: [['FINLAND', 'FIN', 'HELSINKI'], ['SWEDEN', 'SWE', 'STOCKHOLM'], ['NORWAY', 'NOR', 'OSLO']]

Creating a list with subdictionary, see sample below:

countries = [[("Finland", "Helsinki")], [("Sweden", "Stockholm")], [("Norway", "Oslo")]]

output = [

    {"country": country.upper(), "city": capital.upper()}

    for sublist in countries

    for country, capital in sublist

]

# Result: [{'country': 'FINLAND', 'city': 'HELSINKI'}, {'country': 'SWEDEN', 'city': 'STOCKHOLM'}, {'country': 'NORWAY', 'city': 'OSLO'}]

# Day 14 – Higher Order Function

1. Higher order function mainly are :

Function as a Parameter

def sum\_numbers(nums):

    return sum(nums)

def higher\_order\_function(f, lst):

    summation = f(lst)

    return summation

result = higher\_order\_function(sum\_numbers, [1, 3, 5])

print(result)

Function as a Return Value

def square(x):

    return x\*\*2

def cube(x):

    return x\*\*3

def absolute(x):

    if x >= -0:

        return x

    else:

        return -x

def higher\_order\_function(type):

    if type == "square":

        return square

    elif type == "cube":

        return cube

    elif type == "absolute":

        return absolute

result = higher\_order\_function("square")

# result refer to one function, but function need take one argument

print(result(5))

1. Python closures

Python allows a nested function to access the outer scope of the enclosing function. See sample below:

def add\_ten():

ten = 10

def add(num):

return num + ten

return add

# ten is defined outside the add() function but it is still within the enclosing function

closure\_result = add\_ten()

print(closure\_result(5)) # 15

print(closure\_result(10)) # 20

1. Python decorators

A decorator is a design pattern in Python that allows a user to add new functionality to an existing object without modifying its structure.

See sample below:

def uppercase\_decorator(function):

def wrapper():

func = function()

# Typical format a = f() when using a variable to refer to a function

make\_uppercase = func.upper()

return make\_uppercase

return wrapper

@uppercase\_decorator

def greeting():

return 'Welcome to Python'

print(greeting()) # WELCOME TO PYTHON

multiple decorators: applied according to the sequence.

# First Decorator

def uppercase\_decorator(function):

def wrapper():

func = function()

make\_uppercase = func.upper()

return make\_uppercase

return wrapper

# Second decorator

def split\_string\_decorator(function):

def wrapper():

func = function()

splitted\_string = func.split()

return splitted\_string

return wrapper

@split\_string\_decorator

@uppercase\_decorator #This works first

def greeting():

return 'Welcome to Python'

print(greeting())

The decorators closer to the function definition are applied first.

bc the .upper() does not work to list type

so cannot put uppercase wrapper later i.e. first

The comment cannot put directly below the wrapper

Decorators accepting the parameters – to output some outcome relating to one of the original function arguments

def decorator\_with\_parameters(function):

    def wrapper\_accepting\_parameters(para1, para2, para3):

        function(para1, para2, para3)

        print(f"I live in {para3}")

    return wrapper\_accepting\_parameters

@decorator\_with\_parameters  # Wrapper will auto start with a new line

def print\_full\_name(first\_name, last\_name, country):

    print(f"I am {first\_name} {last\_name}. I love to teach")

print(print\_full\_name("FT", "Chen", "Singapore"))

Result:

I am FT Chen. I love to teach

I live in Singapore

1. Built-in higher order function - Map Function

Syntax:-

map(function, iterable)

Rmr to give list() to get the list type result (Default)

numbers\_str = ['1', '2', '3', '4', '5'] # iterable

numbers\_int = map(int, numbers\_str)

print(list(numbers\_int)) # [1, 2, 3, 4, 5]

here if only type:-

print(numbers\_int)

The result will be 0x0000 bc the numbers\_int refer to a function

map with a lambda function

numbers\_squared = map(lambda x : x \*\* 2, numbers)

print(list(numbers\_squared)) # [1, 4, 9, 16, 25]

1. filter Function

sample below: must give a Boolean return value function then use with filter

numbers = [1, 2, 3, 4, 5] # iterable

def is\_odd(num):

if num % 2 != 0:

return True

return False

odd\_numbers = filter(is\_odd, numbers)

print(list(odd\_numbers))

1. Reduce function

Step 1: must import

from functools import reduce

syntax:

To reduce the number of the arguments.

The reduce function applies this function cumulatively to the numbers in the list,

resulting in the product of all the numbers.

See sample below

#Example 1

numbers\_str = ['1', '2', '3', '4', '5'] # iterable

def add\_two\_nums(x, y):

return int(x) + int(y)

total = reduce(add\_two\_nums, numbers\_str)

print(total) # 1+2+3+4+5=15

#Example 2

def mutiple\_two\_nums(x, y):

return int(x) \* int(y)

product = reduce(mutiple\_two\_nums, number\_str)

print(product) # 1\*2\*3\*4\*5=120

1. Explain the difference between map, filter, and reduce.

map: To execute the function to every member in the list and output a list

filter: To execute the conditional to every member and only return the member after filter with True result

reduce: To execute the function to every member cumulatively and only output one final result

1. Explain the difference between higher order function, closure and decorator

higher order function: function can be as a parameter or function as a return value

python closures: Inside a nested function, can access a function within the closing function but outer scope

python decorators: To modify the function result without modifying the original function by writing a small function to define the pattern

1. Chain Multiple operations

# Sample list

numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# Chain multiple operations

result = reduce(

    lambda acc, x: acc + x, filter(lambda x: x % 2 == 0, map(lambda x: x \* 2, numbers))

)

print(result) #2+4+6+...+20=110

map(lambda x: x \* 2, numbers) #2,4,6,...,20

filter(lambda x: x % 2 == 0,lst) #All the numbers are even

reduce(lambda acc, x: acc + x,lst) #Accumate all list items

#2+4+6+...+20=110

This type of the multiple chain script is not readable.

So prefer to use list comprehensions or plain loops for better readability

1. Defaultdict function from collections

from collections import defaultdict

countries = [

  'Afghanistan',

  'Albania',

  'Algeria',

  'Andorra',

  'Angola',

]

output\_dic = defaultdict(int)

for country in countries:

    output\_dic[country[0]] += 1 #First character of each country name

print(output\_dic)

# show an example of defaultdict, () can define a type of the value by default

x = defaultdict(int)  # When define the value is the int

print(f"{x['A'] = }") # x['A'] = 0

x["A"] = 5

print(f"{x['A'] = }") # x['A'] = 5

y = defaultdict(list)

print(

    f"{y['A'] = }"

)  # When define the value is the list, the empty key value is the empty list, Result is y['A'] = []

The advantage compared with if defining a dictionary by self, you need do the additional steps as following:

* 1. You need define an empty dictionary

# output\_dic = {}

* 1. specify the initial type of value of the values

output\_dic[country[0]] = 1

output\_dic = {}

for country in countries:

    if country[0] not in output\_dic:

        output\_dic[country[0]] = 1

    else:

        output\_dic[country[0]] += 1

print(output\_dic)

1. sort by multiple key for dictionary

sorted & lambda function

for country in countries:

    sorted\_countries = sorted(

        countries\_data, key=lambda x: (x["name"], x["capital"], x["population"])

    )

print(countries\_data[:3])

# Day 15 – Type of Errors

1. SyntaxError

print'Hello World'

#SyntaxError

1. NameError

# print(age)

# Age is not defined

1. IndexError

name = ['a','b','c']

# print(name[3])

# Dont hv index 3 in name list

1. ModuleNotFoundError

# import maths

#ModuleNotFoundError: No module named 'maths'

import math #Correct module name then can work

1. AttributeError

import math

# math.PI

#AttributeError: module 'math' has no attribute 'PI'. Did you mean: 'pi'?

print(math.pi) #Can work

1. KeyError

user = {

'name' : 'Fang Ting',

'age' : '30',

'skills' : 'bodybuilding'

}

print(user['name'])

#print(user['Name'])

#KeyError: 'Name'

1. TypeError"

a = 4 + '3'

print(a)

TypeError: unsupported operand type(s) for +: 'int' and 'str'

1. ValueError

output = int('12a')

print(output)

ValueError: invalid literal for int() with base 10: '12a'

1. ImportError

#from math import power

#ImportError: cannot import name 'power' from 'math' (unknown location)

from math import pow #Correct Power function name

output = pow(2,3)

print(output)

# Day 16 – Python Date Time

1. Dir()

dir() : to get the list of names in the namespace of a module, class, or instance. It returns a list of valid attributes for the given object.

1. Datetime module:

['MAXYEAR', 'MINYEAR', 'UTC', '\_\_all\_\_', '\_\_builtins\_\_', '\_\_cached\_\_', '\_\_doc\_\_', '\_\_file\_\_', '\_\_loader\_\_', '\_\_name\_\_', '\_\_package\_\_', '\_\_spec\_\_', 'date', 'datetime', 'datetime\_CAPI', 'time', 'timedelta', 'timezone', 'tzinfo']

1. datetime, date, time (class in datetime) to get the date & time info

ex. Datetime - To get the datetime by using each attribute: year, month, day, hour, minute, second, microsecond

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

from datetime import datetime

now = datetime.now()

print(now) # 2021-07-08 07:34:46.549883

day = now.day # 8

month = now.month # 7

year = now.year # 2021

hour = now.hour # 7

minute = now.minute # 38

second = now.second

timestamp = now.timestamp()

print(day, month, year, hour, minute)

print('timestamp', timestamp)

print(f'{day}/{month}/{year}, {hour}:{minute}') # 8/7/2021, 7:38

1. Formatting date output using strftime

from datetime import datetime

# current date and time

now = datetime.now()

t = now.strftime("%H:%M:%S")

print("time:", t) # time: 01:05:01

time\_one = now.strftime("%m/%d/%Y, %H:%M:%S")

# mm/dd/YY H:M:S format

print("time one:", time\_one) # time one: 12/05/2019, 01:05:01

time\_two = now.strftime("%d/%m/%Y, %H:%M:%S")

# dd/mm/YY H:M:S format

print("time two:", time\_two) # time two: 05/12/2019, 01:05:01

strftime symbols:

A screenshot of a computer

Description automatically generated

1. String to time using strptime

from datetime import datetime

date\_string = "5 December, 2019"

print("date\_string =", date\_string) # date\_string = 5 December, 2019

date\_object = datetime.strptime(date\_string, "%d %B, %Y")

print("date\_object =", date\_object) # date\_object = 2019-12-05 00:00:00

1. Using date from datetime

from datetime import date

d = date(2020, 1, 1)

print(d)

print('Current date:', d.today()) # 2019-12-05

# date object of today's date

today = date.today()

print("Current year:", today.year) # 2019

print("Current month:", today.month) # 12

print("Current day:", today.day) # 5

1. Using time to represent time

from datetime import time

# time(hour = 0, minute = 0, second = 0)

a = time()

print("a =", a)

# time(hour, minute and second)

b = time(10, 30, 50)

print("b =", b)

# time(hour, minute and second)

c = time(hour=10, minute=30, second=50)

print("c =", c)

# time(hour, minute, second, microsecond)

d = time(10, 30, 50, 200555)

print("d =", d)

1. Time(hour = xx, minute = xx, second = xx ) class

Same way with or without keyword

b = time(10, 30, 50)

print(f'{b}') #10:30:50, can directly to use b to print the time instead of the syntax above

c = time(hour = 10, minute = 30, second = 50)

print(f'{c}') #10:30:50

1. f-string cannot use in time() class

from datetime import time

a = time()

print(f'{a = }') #This syntax cannot work, the output is a = datetime.time(0, 0)

print(f'{a}') # 00:00:00

In the case of a time object, it seems that the default \_\_format\_\_ method is not suitable for the f-string formatting.

1. Difference Between Two Points in Time Using timedelta

class datetime.timedelta attributes:

(days=0, seconds=0, microseconds=0, milliseconds=0, minutes=0, hours=0, weeks=0)

from datetime import timedelta

t1 = timedelta(weeks=12, days=10, hours=4, seconds=20)

t2 = timedelta(days=7, hours=5, minutes=3, seconds=30)

t3 = t1 - t2

print("t3 =", t3)

1. Difference Between Two dates Using date class

#Method 1: Using date function

today = date.today()

target\_date = date(1970, 1, 1)

time\_to\_target\_date = today - target\_date

print(f'{time\_to\_target\_date}')

# Day 17 – Exception Handling

# Day 18 – Regular Expression

1. Regular expression

A group of text boxes

Description automatically generated

# Day 19 – File Handling

1. Common practice, to use the path starts with r’’ to ensure that backslashes (\) are treated as literal characters and not as escape characters. This is particularly useful when working with Windows file paths, where backslashes are used as path separators.

# Regular string - may cause issues with backslashes in Windows paths

file\_path = 'C:\Users\your\_username\Documents\file.txt'

# Use of raw string literal to avoid issues with backslashes

with open(r'C:\Users\your\_username\Documents\file.txt', 'r', encoding='utf-8') as file:

# Using forward slashes (works on Windows too)

with open('C:/Users/your\_username/Documents/file.txt', 'r', encoding='utf-8') as file:

# Using double backslashes in a regular string

with open('C:\\Users\\your\_username\\Documents\\file.txt', 'r', encoding='utf-8') as file:

1. open() function, few modes’ use

**open('filename', mode)** # mode(r, a, w, x, t,b) could be to read, write, update

* "r" - Read - Default value. Opens a file for reading, it returns an error if the file does not exist
* "a" - Append - Opens a file for appending, creates the file if it does not exist
* "w" - Write - Opens a file for writing, creates the file if it does not exist
* "x" - Create - Creates the specified file, returns an error if the file exists
* "t" - Text - Default value. Text mode
* "b" - Binary - Binary mode (e.g. images)

1. read() function

f = open('./files/reading\_file\_example.txt') # The path starts from the same directory

lines = f.readlines()

read(): read the whole text as string. If we want to limit the number of characters we want to read, we can limit it by passing int value to the read(number) method.

Read(10): Only read the first 10 characters of the file

readline(): read only the first line

readlines(): read all the text line by line and returns a list of lines

splitlines(): get all the lines as a list #

['This is an example to show how to open a file and read.', 'This is the second line of the text.']

1. Delete files

import os

if os.path.exists('./files/example.txt'):

os.remove('./files/example.txt')

else:

print('The file does not exist')

1. **Changing dictionary to JSON**

import json

person = {

"name": "Asabeneh",

"country": "Finland",

"city": "Helsinki",

"skills": ["JavaScrip", "React", "Python"]

} # python dictionary

person\_json = json.dumps(person, indent=4) # indent could be 2, 4, 8. It beautifies the json

print(type(person\_json))

print(person\_json)

1. **Changing JSON to Dictionary**

import json

person\_json = '''{

"name": "Asabeneh",

"country": "Finland",

"city": "Helsinki",

"skills": ["JavaScrip", "React", "Python"]

}''' #Json file

# let's change JSON to dictionary

person\_dct = json.loads(person\_json)

print(type(person\_dct)) #dictionary

1. **File with csv Extension**

Sample csv file:

"name","country","city","skills"

"Asabeneh","Finland","Helsinki","JavaScript"

import csv

with open('./files/csv\_example.csv') as f:

csv\_reader = csv.reader(f, delimiter=',') # w use, reader method to read csv

line\_count = 0

for row in csv\_reader:

if line\_count == 0:

print(f'Column names are :{", ".join(row)}')

line\_count += 1

else:

print(

f'\t{row[0]} is a teachers. He lives in {row[1]}, {row[2]}.')

line\_count += 1

print(f'Number of lines: {line\_count}')

1. **File with xlsx Extension**
2. **File with xml Extension**

# Day 20 – PIP (Preferred installer Program)

1. To check if pip is installed

Pip – version

My pip version :pip 23.2.1

1. Installing packages using 'pip <packagename>'
2. Uninstalling packages using

pip uninstall <packagename>

1. To check all installed pip

pip list

A screenshot of a computer program

Description automatically generated

1. To show information about a package

Pip show <packagename>

A screen shot of a computer

Description automatically generated

1. To generate the current pip packages as a requirement file:

pip freeze

A screen shot of a computer

Description automatically generated

1. To install pip a package but still need import that particular function

Yes, that's correct. When you install a new Python package using pip, you need to import the corresponding module in your Python script or module to use its functionality. The import statement is necessary to make the functions, classes, or other elements from the installed package available in your code.

1. **To create a package:**

Created a mypackage in the same folder directory and created two .py files and one init.py file

The init.py is essential for the folder to be recognized by Python as a package.

Now the mypackage folder are treated as one package alr

# Day 21 – Classes and Objects

1. To create a class

class Person:

    pass

print(Person)

1. To create an object of the class

#Creating an object

p = Person()

print(p) #<class '\_\_main\_\_.Person'>

1. To create a class for a constructor and function

Constructor: use constructor function to make our class more useful: \_\_init\_\_

Function: to give a function involving those defined attributes of the instance of the class

#More parameters for constructor function

class Person:

def \_\_init\_\_(self, firstname, lastname, age, country, city):

# \_\_init\_\_ is the Constructor, name, age.. is the attributes to class

        self.age = age

        self.firstname = firstname

        self.lastname = lastname

        self.country = country

        self.city = city

    def person\_info(self): #Define a method (function) to the class

        return f'{self.firstname} {self.lastname} is {self.age}. She lives in {self.city}, {self.country}.'

person\_one = Person('FangTing', 'Chen', 18, 'China', 'Shanghai')

print(person\_one.person\_info()) #instance.method

# FangTing Chen is 18. She lives in Shanghai, China.

# Day 22– Web Scrapping

1. HTML Tags basic general structures

<!DOCTYPE html>

<html>

<head>

<title>Page Title</title>

</head>

<body>

<div class="my-div">

<h1>My Div Content</h1>

<p>This is some text inside the <code>&lt;div&gt;</code>.</p>

</div>

</body>

</html>

**The intro for common tags:**

1. <!DOCTYPE html>: This declaration defines that the document is an HTML5 document.
2. <html>: The root element of an HTML page. All other elements are nested within this tag.
3. <head>: Contains meta information about the HTML page, such as the title.
4. <title>: Specifies a title for the HTML page (displayed in the browser’s title bar or tab).
5. <body>: Defines the document’s body and contains all visible content (headings, paragraphs, images, links, etc.).
6. We define a <div> with the class name “my-div
7. <h1>: Represents a large heading.
8. <p>: Represents a paragraph.

An HTML element consists of a start tag, content, and an end tag. For example:

1. The main procedures to parse a HTML Website

‘Html.parser’ : The 'html.parser' string refers to a specific parser module in Python that is used for parsing HTML (HyperText Markup Language) and XHTML (eXtensible HyperText Markup Language) formatted text files.

1. HTML Tags basic general structures for one with table

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Sample Table</title>

</head>

<body>

<table>

<tr>

<th>Company</th>

<th>Contact</th>

<th>Country</th>

</tr>

<tr>

<td>Alfreds Futterkiste</td>

<td>Maria Anders</td>

<td>Germany</td>

</tr>

<tr>

<td>Centro comercial Moctezuma</td>

<td>Francisco Chang</td>

<td>Mexico</td>

</tr>

</table>

</body>

</html>

In this example:

<table> defines the entire table.

<tr> represents a row within the table.

<th> defines header cells (usually used for column headings).

<td> represents data cells (actual content) within the rows.

# Day 23 – Virtual Environment

1. To install virtualenv

Pip install virtualenv

1. To create a folder under a project folder

C:\Users\User\Documents\30DaysOfPython\flask\_project>python -m venv venv

1. To check the venv by

Ls

A screenshot of a computer

Description automatically generated

1. To activate

C:\Users\User\Documents\30DaysOfPython\flask\_project> venv\Scripts\activate

1. To deactivate



1. One project using one virtual environment

# Day 24 – Statistics

1. Import numpy

import numpy as np

1. Complex number application in the array

#To check the size of any array

x = np.array([1, 2, 3], dtype = np.complex128)

print(x) #[1.+0.j 2.+0.j 3.+0.j]

'''The dtype parameter specifies the data type of the array.

np.complex128 indicates that the array should contain complex numbers with 128-bit precision.

Since the array x contains complex numbers, each number is printed in the format a+bj,

where a is the real part and b is the imaginary part.

In this case, the imaginary part is 0 for all elements,

so it's just a+0j, which simplifies to a'''

print(x.itemsize) #16

'''To return the size of a single array element in bytes. np.complex128 has a size of 128 bits,

which is equivalent to 16 bytes (1 byte =  8 bits).

1. The difference between array and list
2. Arrays support vectorized operations, while lists don’t.
3. Once an array is created, you cannot change its size. You will have to create a new array or overwrite the existing one.
4. Every array has one and only one dtype. All items in it should be of that dtype.
5. An equivalent numpy array occupies much less space than a python list of lists.
6. numpy arrays support Boolean indexing.
7. Convert between list and numpy array
8. Convert from list to array
   1. Keep integer

python\_list = [1,2,3,4,5]

numpy\_array\_from\_list = np.array(python\_list)

print(numpy\_array\_from\_list) # array([1, 2, 3, 4, 5])

* 1. Convert to float

numpy\_array\_from\_list\_float = np.array(python\_list, dtype=float)

print(numpy\_array\_from\_list\_float) #array[1. 2. 3. 4. 5.]

* 1. Convert to Boolean

numpy\_bool\_array = np.array([0, 1, -1, 0, 0], dtype=bool)

print(numpy\_bool\_array)#array[False True True False False]

* 1. **Create as 2D array**

two\_dimensional\_list = [[0,1,2], [3,4,5], [6,7,8]]

numpy\_two\_dimension\_list = np.array(two\_dimensional\_list)

print(numpy\_two\_dimension\_list)

1. Convert from array to list

np\_to\_list = numpy\_array\_from\_list.tolist()

1. Creating numpy array from tuple

python\_tuple = (1, 2, 3, 4, 5)

numpy\_array\_from\_tuple = np.array(python\_tuple)

print(f'{numpy\_array\_from\_tuple}') #[1 2 3 4 5]

1. **Size** of a numpy array

numpy\_array\_from\_list = np.array([1, 2, 3, 4, 5])

two\_dimensional\_list = np.array([[0, 1, 2],

                                 [3, 4, 5],

                                 [6, 7, 8]])

print({numpy\_array\_from\_list.size}) #{5}

print({two\_dimensional\_list.size}) #{9}

1. Converting the data type inside array

numpy\_int\_arr = np.array([1, 2, 3, 4], dtype='float')

1. Position 2D array by index

two\_dimension\_array = np.array([(1, 2, 3), (4, 5, 6), (7, 8, 9)])

first\_column = two\_dimension\_array[:,0] #array([1, 4, 7])

before\_first\_row = two\_dimension\_array[:0] # array([], shape=(0, 3), dtype=int32)

until\_first\_row = two\_dimension\_array[:1] #array([[1, 2, 3]])

second\_column = two\_dimension\_array[:,1] #array([2, 5, 8])

third\_column = two\_dimension\_array[:,2] #array([3, 6, 9])

1. Slicing

two\_dimension\_array = np.array([(1, 2, 3), (4, 5, 6), (7, 8, 9)])

first\_two\_rows\_and\_columns = two\_dimension\_array[0:2, 0:2]

print(first\_two\_rows\_and\_columns)

'''

[[1 2]

 [4 5]]'''

1. reverse the rows and array

two\_dimension\_array = np.array([(1, 2, 3), (4, 5, 6), (7, 8, 9)])

print(two\_dimension\_array[1::])

'''

[[4 5 6]

 [7 8 9]]'''

1. To reserve the row and column positions

two\_dimension\_array = np.array([(1, 2, 3), (4, 5, 6), (7, 8, 9)])

print(two\_dimension\_array[::-1, ::-1]) #reverse the row and column position both

'''

[[9 8 7]

 [6 5 4]

 [3 2 1]]

'''

1. To create default value array, zeros or ones

numpy\_zeroes = np.zeros((3,3),dtype=int,order='C')

print(numpy\_zeroes)

'''

[[0 0 0]

 [0 0 0]

 [0 0 0]]

'''

numpy\_ones = np.ones((3,3), dtype=int, order='C')

print(numpy\_ones)

'''

[[1 1 1]

 [1 1 1]

 [1 1 1]]

'''

1. Reshape

first\_shape = np.array([(1, 2, 3), (4, 5, 6)])

print(first\_shape)

reshaped = first\_shape.reshape(3, 2)

print(reshaped)

'''

[[1 2 3]

 [4 5 6]]

'''

1. Flatten

#Flatten

flattened = reshaped.flatten()

print(flattened) #[1 2 3 4 5 6]

1. Horizontal stack & vertical stack

print("Horizontal stack".center(80, "-"))

np\_list\_one = np.array([1, 2, 3])

np\_list\_two = np.array([4, 5, 6])

print(np\_list\_one + np\_list\_two) #[5 7 9]

print(f'{np.hstack((np\_list\_one, np\_list\_two))}') #[1 2 3 4 5 6]

print("Vertical Stack".center(80, "-"))

print(f'{np.vstack((np\_list\_one, np\_list\_two))}')

'''

[[1 2 3]

 [4 5 6]]

'''

1. Generate random number
2. Method 1 : by random.random() or random.randint()

random\_float = np.random.random(5) #random() method, () stands for how many array number to be generated

print(random\_float) #Generating as an array

random\_int = np.random.randint(0, 11, size=(4, 2)) #4 rows, 2 cols array with random integer between 0~10

print(random\_int)  #[4 4 7 8]

1. Method 2 : np.random.normal(mu, sigma, size)

print("Generating Random Numbers from random module".center(80, "-"))

normal\_array = np.random.normal(79, 15, 80)

[ 90.80079252  78.28596759  78.02159718  76.41257498  71.60812818

  74.58027718  72.07787582  77.02227712  65.50652843  95.75141089

  97.90239649  54.271475    94.60833075  77.70214143 100.44400474

  78.00670482  76.81492309 107.82887331  92.99657298 100.78859393

…]

#The set of number mean value is 79, the standard deviation is 15, and total 80 numbers

1. To generate the diagram to visualize a set of data by matplot and seaborn module
2. Sample 1: numpy.logspace()

logspace\_example = np.logspace(2.0, 4.0, num=4)

print(logspace\_example) #[  100.           464.15888336  2154.43469003 10000.        ]

#it generates 4 numbers spaced evenly on a logarithmic scale between 10^2 and 10^4

A screenshot of a math problem

Description automatically generated

import matplotlib.pyplot as plt

#This module provides a MATLAB-like plotting framework in Python.

import seaborn as sns

#seaborn is a statistical data visualization library built on top of Matplotlib.

# Compute the values

start = 2.0

stop = 4.0

num = 4

indices = np.arange(num)

values = 10\*\*(start + indices \* (stop - start) / (num - 1))

# Plotting

plt.figure(figsize=(8, 6))

plt.plot(indices, values, marker='o', linestyle='-')

plt.title('Values on Logarithmic Scale')

plt.xlabel('Index')

plt.ylabel('Value')

plt.xticks(indices)

plt.grid(True)

plt.show()

A screen shot of a graph

Description automatically generated

1. Sample 2: np.random.normal(mu, sigma, samples)

mu = 28

sigma = 15

samples = 100000

x = np.random.normal(mu, sigma, samples)

ax = sns.displot(x)

ax.set(xlabel='x', ylabel='y')

plt.show()

A graph of a number of numbers

Description automatically generated with medium confidence

1. Numpy. Arrange(), similar as range() function for python list

#Normal list: range(starting, stop, step)

lst = range(0, 11, 2)

for i in lst:

    print(i) #0, 2, 4, 6, 8, 10

#To get as an Array: numpy.arrange(starting, stop, step)

whole\_numbers = np.arange(0, 20, 1)

print(whole\_numbers) #[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19]

odd\_numbers = np.arange(1, 20, 2)

print(odd\_numbers) #[ 1  3  5  7  9 11 13 15 17 19]

1. Numpy. Linspace() and numpy.logspace()

#numpy.linspace()

#To create 10 values from 1 to 5 evenly spaced.

linspace\_example = np.linspace(1.0, 5.0, num=10)

print(linspace\_example) #[1.         1.44444444 1.88888889 2.33333333 2.77777778 3.22222222 3.66666667 4.11111111 4.55555556 5.        ]

#Not to include the last value in the interval but still with 10 spaces

linspace\_example = np.linspace(1.0, 5.0, num=10, endpoint=False)

print(linspace\_example)

#numpy.logspace() in python with example

logspace\_example = np.logspace(2.0, 4.0, num=4)

print(logspace\_example) #[  100.           464.15888336  2154.43469003 10000.        ]

#it generates 4 numbers spaced evenly on a logarithmic scale between 10^2 and 10^4

1. To find the max/min value among each row or column

two\_dimension\_array = np.array([(1, 2, 3), (4, 5, 6), (7, 8, 9)])

'''

np.max(): This function takes multiple arrays as arguments and returns the maximum of all the elements.

np.amax(): This function is an alias for np.max(). It behaves exactly the same way as np.max().

'''

print(f'Row with minimum: {np.amin(two\_dimension\_array, axis=0)}')

print(f'Column with maximum: {np.amax(two\_dimension\_array, axis=1)}') #Row with maximum: [3 6 9]

1. To create repeating sequences

a = [1, 2, 3]

#Repeat whole of 'a' two times

print(f'Tile: {np.tile(a, 2)}') #Tile: [1 2 3 1 2 3]

#Repeat each element of 'a' two times

print(f'Repeat: {np.repeat(a, 2)}') #Repeat: [1 1 2 2 3 3]

1. Stats.mode

print(f'{stats.mode(np\_normal\_dis) = }') #ModeResult(mode=3.4011733054941393, count=1)

'''the mode represents the value in the dataset that occurs with the highest frequency.

count:the count of occurrences of the mode value, which is 1 in this case.

when there are multiple values with the same maximum frequency,

the function returns the smallest value among them as the mode.'''

1. The dot product/ matrix multiplication/ determinant

#numpy.dot(): Dot Product

f = np.array([1, 2, 3])

g = np.array([4, 5, 6])

# [1, 2, 3] \* [4, 5, 6]

result = np.dot(f,g)

print(result) # 1\*4+2\*5 + 3\*6 = 32

print("NumPy Matrix Multiplication with np.matmul()".center(80, "-"))

#Matmul: matruc project of two arrays

h = [[1, 2], [3, 4]]

i = [[5, 6], [7, 8]]

result = np.matmul(h, i)

print(result)

'''

[1, 2]     [5, 6]           [1x5+2x7, 1x6+2x8]         [19, 22]

[3, 4]     [7, 8]    ->     [3x5+4x7, 3x6+4x8]    ->   [43, 50]

'''

#for 2D array [[a, b], [c, d]] is ad - bc:

i = [[5, 6], [7, 8]]

result = np.linalg.det(i)

print(result) #-2

#For a stack of matrices

a = np.array([ [[1, 2], [3, 4]], [[1, 2], [2, 1]], [[1, 3], [3, 1]] ])

'''

[[1, 2]     [1, 2]     [1, 3]

[3, 4]]     [2, 1]     [3, 1]

'''

result\_shape = a.shape

print(result\_shape) #(3, 2, 2)

result= np.linalg.det(a)

print(result) #[-2. -3. -8.]

1. To set array with value by positioning

z = np.zeros((8, 8))

z[1::2, ::2] = 1

print(z)

'''

[[0. 0. 0. 0. 0. 0. 0. 0.]

 [1. 0. 1. 0. 1. 0. 1. 0.]

 [0. 0. 0. 0. 0. 0. 0. 0.]

 [1. 0. 1. 0. 1. 0. 1. 0.]

 [0. 0. 0. 0. 0. 0. 0. 0.]

 [1. 0. 1. 0. 1. 0. 1. 0.]

 [0. 0. 0. 0. 0. 0. 0. 0.]

 [1. 0. 1. 0. 1. 0. 1. 0.]]

step 1: 1::2 -> first element stands for row number rules: means from 2nd row, step 2rows, i.e. 2nd row, 4th row, 6th rows, 6th rows

Then ::2 -> second element stands for column number rules: from 1st col, step 2ols, i.e, 1st col, 3rd col, 5th col, 7th col

according to both positioning criteria, then to set the value as 1

'''

z[::2, 1::2] = 1

print(z)

'''

[[0. 1. 0. 1. 0. 1. 0. 1.]

 [1. 0. 1. 0. 1. 0. 1. 0.]

 [0. 1. 0. 1. 0. 1. 0. 1.]

 [1. 0. 1. 0. 1. 0. 1. 0.]

 [0. 1. 0. 1. 0. 1. 0. 1.]

 [1. 0. 1. 0. 1. 0. 1. 0.]

 [0. 1. 0. 1. 0. 1. 0. 1.]

 [1. 0. 1. 0. 1. 0. 1. 0.]]

 '''

# Day 26 – Python for web

1. To create a folder under the current directory : mkdir <foldername>