

# Dancing with Python 101



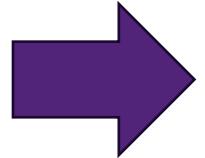
Yoochan Myung

23<sup>rd</sup> Nov 2023

School of Chemistry and Molecular Biosciences  
Postdoctoral Research Fellow

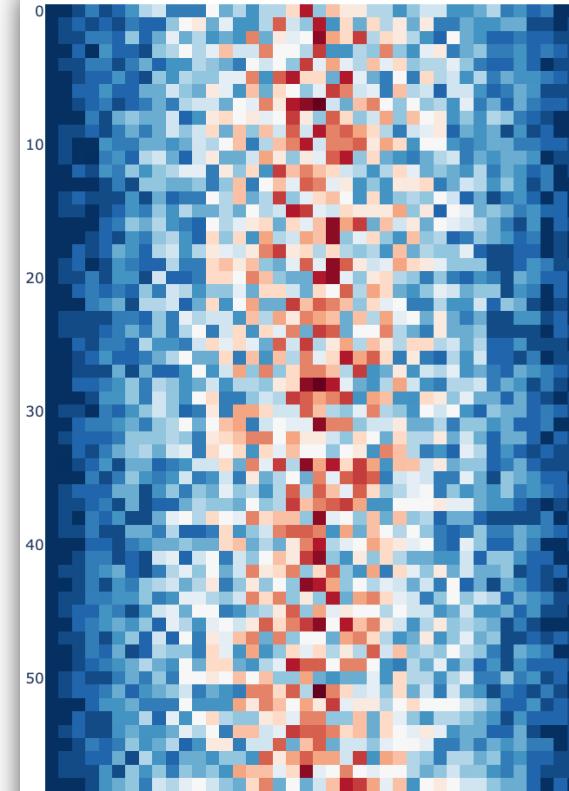
# After this workshop...

inflammation-01.csv



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	2
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	3
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	4
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	5
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	6
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	7
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	8
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	9
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	10
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	11
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	12
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	13
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	14
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	15
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	16
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	17
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	18
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	19
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	20
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	21
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	22
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	23
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	24
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	25
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	26
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	27
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	28
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	29
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	30

file (CSV)



plot



# Tasks

## What's Python?

## How to use Python?

- IDE software and code editors

## How to manage data/information ?

- Data type and structures
- Condition (IF), loop (FOR)

## How to make a Function?

## How to analyse data using Numpy?

## How to plot data?

inflammation-01.csv		Open with Numbers	
0	0	1	3
1	2	4	7
2	3	5	8
3	4	6	9
4	5	7	10
5	6	8	11
6	7	9	12
7	8	10	13
8	9	11	15
9	10	12	17
10	11	13	19
11	12	14	19
12	13	15	17
13	14	16	18
14	15	17	19
15	16	18	20
16	17	19	21
17	18	20	22
18	19	21	24
19	20	22	26
20	21	23	28
21	22	24	30
22	23	25	32
23	24	26	34
24	25	27	36
25	26	28	38
26	27	29	40
27	28	30	42
28	29	31	44
29	30	32	46
30	31	33	48
31	32	34	50
32	33	35	52
33	34	36	54
34	35	37	56
35	36	38	58
36	37	39	60
37	38	40	62
38	39	41	64
39	40	42	66
40	41	43	68
41	42	44	70
42	43	45	72
43	44	46	74
44	45	47	76
45	46	48	78
46	47	49	80
47	48	50	82
48	49	51	84
49	50	52	86
50	51	53	88
51	52	54	90
52	53	55	92
53	54	56	94
54	55	57	96
55	56	58	98
56	57	59	100
57	58	60	102
58	59	61	104
59	60	62	106
60	61	63	108
61	62	64	110
62	63	65	112
63	64	66	114
64	65	67	116
65	66	68	118
66	67	69	120
67	68	70	122
68	69	71	124
69	70	72	126
70	71	73	128
71	72	74	130
72	73	75	132
73	74	76	134
74	75	77	136
75	76	78	138
76	77	79	140
77	78	80	142
78	79	81	144
79	80	82	146
80	81	83	148
81	82	84	150
82	83	85	152
83	84	86	154
84	85	87	156
85	86	88	158
86	87	89	160
87	88	90	162
88	89	91	164
89	90	92	166
90	91	93	168
91	92	94	170
92	93	95	172
93	94	96	174
94	95	97	176
95	96	98	178
96	97	99	180
97	98	100	182
98	99	101	184
99	100	102	186
100	101	103	188
101	102	104	190
102	103	105	192
103	104	106	194
104	105	107	196
105	106	108	198
106	107	109	200
107	108	110	202
108	109	111	204
109	110	112	206
110	111	113	208
111	112	114	210
112	113	115	212
113	114	116	214
114	115	117	216
115	116	118	218
116	117	119	220
117	118	120	222
118	119	121	224
119	120	123	226
120	121	124	228
121	122	125	230
122	123	126	232
123	124	127	234
124	125	128	236
125	126	130	238
126	127	131	240
127	128	132	242
128	129	133	244
129	130	134	246
130	131	135	248
131	132	136	250
132	133	137	252
133	134	138	254
134	135	140	256
135	136	141	258
136	137	142	260
137	138	143	262
138	139	144	264
139	140	145	266
140	141	146	268
141	142	147	270
142	143	148	272
143	144	149	274
144	145	150	276
145	146	151	278
146	147	152	280
147	148	153	282
148	149	154	284
149	150	155	286
150	151	156	288
151	152	157	290
152	153	158	292
153	154	159	294
154	155	160	296
155	156	161	298
156	157	162	300
157	158	163	302
158	159	164	304
159	160	165	306
160	161	166	308
161	162	167	310
162	163	168	312
163	164	169	314
164	165	170	316
165	166	171	318
166	167	172	320
167	168	173	322
168	169	174	324
169	170	175	326
170	171	176	328
171	172	177	330
172	173	178	332
173	174	179	334
174	175	180	336
175	176	181	338
176	177	182	340
177	178	183	342
178	179	184	344
179	180	185	346
180	181	186	348
181	182	187	350
182	183	188	352
183	184	189	354
184	185	190	356
185	186	191	358
186	187	192	360
187	188	193	362
188	189	194	364
189	190	195	366
190	191	196	368
191	192	197	370
192	193	198	372
193	194	199	374
194	195	200	376
195	196	201	378
196	197	202	380
197	198	203	382
198	199	204	384
199	200	205	386
200	201	206	388
201	202	207	390
202	203	208	392
203	204	209	394
204	205	210	396
205	206	211	398
206	207	212	400
207	208	213	402
208	209	214	404
209	210	215	406
210	211	216	408
211	212	217	410
212	213	218	412
213	214	219	414
214	215	220	416
215	216	221	418
216	217	222	420
217	218	223	422
218	219	224	424
219	220	225	426
220	221	226	428
221	222	227	430
222	223	228	432
223	224	229	434
224	225	230	436
225	226	231	438
226	227	232	440
227	228	233	442
228	229	234	444
229	230	235	446
230	231	236	448
231	232	237	450
232	233	238	452
233	234	239	454
234	235	240	456
235	236	241	458
236	237	242	460
237	238	243	462
238	239	244	464
239	240	245	466
240	241	246	468
241	242	247	470
242	243	248	472
243	244	249	474
244	245	250	476
245	246	251	478
246	247	252	480
247	248	253	482
248	249	254	484
249	250	255	486
250	251	256	488
251	252	257	490
252	253	258	492
253	254	259	494
254	255	260	496
255	256	261	498
256	257	262	500
257	258	263	502
258	259	264	504
259	260	265	506
260	261	266	508
261	262	267	510
262	263	268	512
263	264	269	514
264	265	270	516
265	266	271	518
266	267	272	520
267	268	273	522
268	269	274	524
269	270	275	526
270	271	276	528
271	272	277	530
272	273	278	532
273	274	279	534
274	275	280	536
275	276	281	538
276	277	282	540
277	278	283	542
278	279	284	544
279	280	285	546
280	281	286	548
281	282	287	550
282	283	288	552
283	284	289	554
284	285	290	556
285	286	291	558
286	287	292	560
287	288	293	562
288	289	294	564
289	290	295	566
290	291	296	568
291	292	297	570
292	293	298	572
293	294	299	574
294	295	300	576
295	296	301	578
296	297	302	580
297	298	3	

# Why is Programming Language needed?

-1 and 0.5



-0.5

**data (number)**

**calculator (function/method)**

**outcome**

“Ethan”, 174 cm, B,  
...etc



CODE



**data (information)**

**function/method**

# What's Python?

- A **high-level** programming language, 1991 (v 0.9.0) ~ current (v 3.11.4)
- Free and Open-Source

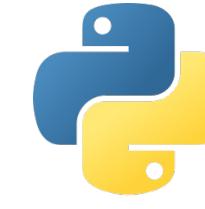
```
print('Hello, World!')
```



high-level language

```
01001000 01100101  
01101100 01101100  
01101111...
```

low-level language



# What's Python?

Why Python over other languages?

- **Easy-of-use:** great readability and straightforward syntax
- **Large and active community:** a vast community such as scientific computing, machine learning, and web development etc.
- Easily get helps via ~~online communities such as Stack overflow, Reddit, StackExchange, chatGPT.~~



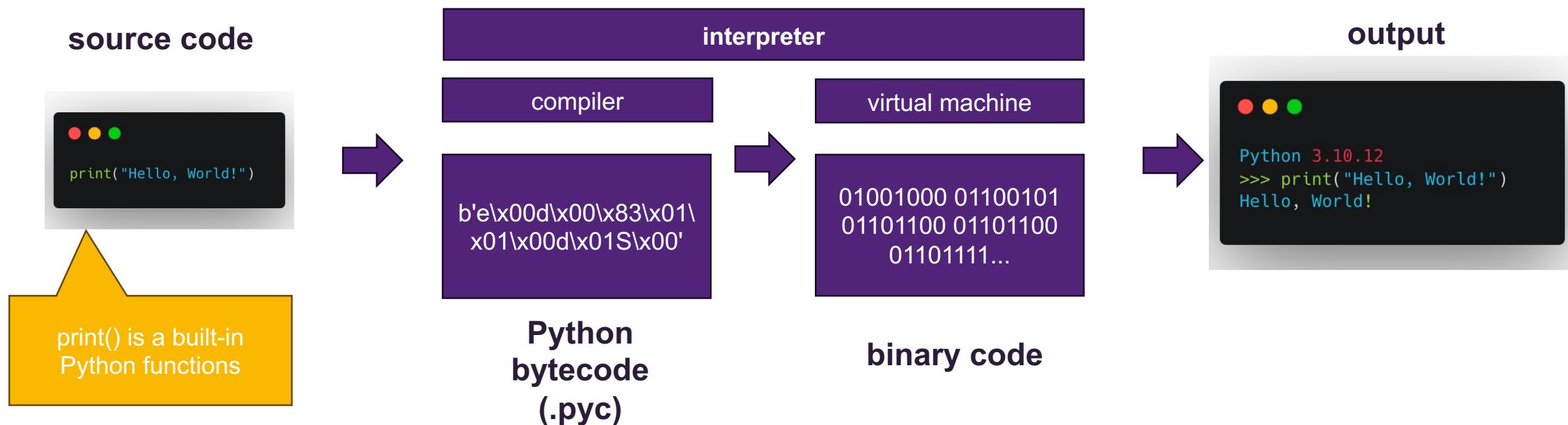
- **Extensive third-party packages:** a rich ecosystem of libraries (further installations are required)  
import
- The **Python Package Index (PyPI)**, the official third-party software repository for Python, provides more than 450,000 Python packages.
- **Pandas** for data manipulation and analysis, **Scikit-learn** for machine learning, **NumPy** and **Scipy** for mathematical and scientific computing, and **Pytorch** for deep learning



# What's Python?

How the Python works?

- **source code**: your code
- **bytecode**: intermediate representation of your code
- **binary code**: two-symbol machine code using “0” and “1”



# What's Python?

Python 2 vs Python 3

**Dropped support for Python 2 in 2020.**

- For storage of strings, Python 2 uses ASCII and Python 3 uses UNICODE.
- In Python2, "Print" is a statement not a function.

**Converting Python 2 codes to Python 3 may need a lot of effort due to the incompatibility above.**



```
ymyung@ymyung-Precision-5820-Tower: ~
ymyung@ymyung-Precision-5820-Tower: ~ (ssh)
(deepk_py27) ymyung@ymyung-Precision-5820-Tower:~$ python
Python 2.7.18 |Anaconda, Inc.| (default, Nov 25 2022, 06:27:37)
[GCC 11.2.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> a = "안녕!"
>>> a
'\xec\x95\x88\xeb\x85\x95!'
>>> type(a)
<type 'str'>
>>> print a
안녕!
>>>
```

```
python (python) python (python) python (python)
python (python) python (python) python (python)
python (python) python (python) python (python)

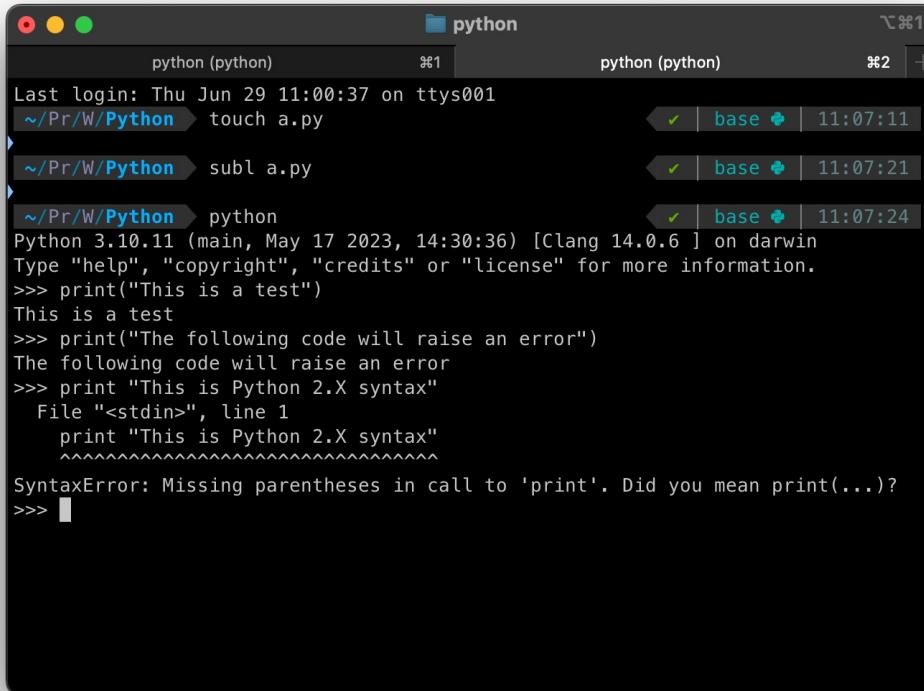
python
Python 3.10.11 (main, May 17 2023, 14:30:36) [Clang 14.0.6 ] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> a = "안녕!"
>>> a
'안녕!'
>>> type(a)
<class 'str'>
>>> print(a)
안녕!
>>> print a
File "<stdin>", line 1
    print a
      ^
      ^
      ^
SyntaxError: Missing parentheses in call to 'print'. Did you mean print(...)?
>>>
```

안녕! = Hello!

*“Consider the third-party packages you intend to use. If none are required, stick with the latest versions available.”*

# How to code?

**via Python IDLE\* (console)**



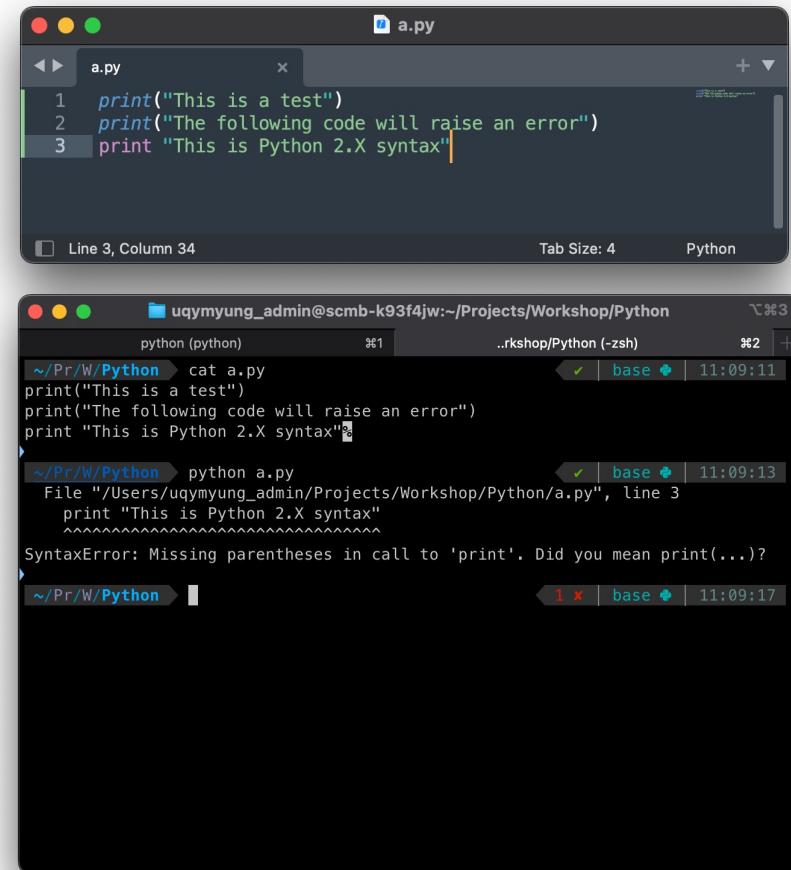
```

python (python) #1
Last login: Thu Jun 29 11:00:37 on ttys001
~/Pr/W/Python touch a.py
~/Pr/W/Python subl a.py
~/Pr/W/Python python
Python 3.10.11 (main, May 17 2023, 14:30:36) [Clang 14.0.6 ] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print("This is a test")
This is a test
>>> print("The following code will raise an error")
The following code will raise an error
>>> print "This is Python 2.X syntax"
  File "<stdin>", line 1
    print "This is Python 2.X syntax"
               ^
SyntaxError: Missing parentheses in call to 'print'. Did you mean print(...)?
>>>

```

real time,  
one-by-one,  
easier to experiment, test small code  
segments

**via Python file execution**



```

a.py
1 print("This is a test")
2 print("The following code will raise an error")
3 print "This is Python 2.X syntax"

Line 3, Column 34
Tab Size: 4 Python

uqymyung_admin@scmb-k93f4jw:~/Projects/Workshop/Python
python (python) #1 .rkshop/Python (-zsh) #2
~/Pr/W/Python cat a.py
print("This is a test")
print("The following code will raise an error")
print "This is Python 2.X syntax"

~/Pr/W/Python python a.py
  File "/Users/uqymyung_admin/Projects/Workshop/Python/a.py", line 3
    print "This is Python 2.X syntax"
               ^
SyntaxError: Missing parentheses in call to 'print'. Did you mean print(...)?
~/Pr/W/Python

```

faster,  
better to write long codes

# Integrated Development Environment (IDE):

1. Syntax highlighting
2. Debugging
3. Integrated Terminal
4. ...



THE UNIVERSITY  
OF QUEENSLAND  
AUSTRALIA



Visual Studio Code

```
tenfold_regressor.py
1 #!/usr/bin/python
2
3 # # # # # Yooschan Hyung - yuchan@gmail.com
4 # # # # #
5
6 import pandas as pd
7 import numpy as np
8 import os
9 import time
10 import sys
11 import math
12 import re
13 import argparse
14 from math import sqrt
15 import statsmodels.stats.api as sms
16 import joblib
17
18 from sklearn import patch_sklearn
19 patch_sklearn()
20
21 from sklearn.ensemble import GradientBoostingRegressor, RandomForestRegressor, AdaBoostRegressor, ExtraTreesRegressor
22 from xgboost.sklearn import XGBRegressor
23 from sklearn.gaussian_process import GaussianProcessRegressor
24 from sklearn.neighbors import KNeighborsRegressor
25 from sklearn.model_selection import KFold
26 from sklearn.metrics import mean_squared_error
27 from sklearn.neural_network import MLPRegressor
28 from sklearn.pipeline import Pipeline
29 from sklearn.compose import make_column_transformer
30 from sklearn.preprocessing import StandardScaler, OneHotEncoder
31 from sklearn.impute import SimpleImputer
32
33 timestamp = time.strftime("%Y%m_%d%H%M")
34
35 def run(algorithm, fname, kfold, training_pd, blind_pd, output_dir, target_label, error_type, n_cores, num_of_shuffling, random_state, no_scaling, save_model):
36     result_M = pd.DataFrame()
37     result_M['algorithm'] = algorithm
38     result_M['fname'] = fname
39     result_M['kfold'] = kfold
40     result_M['training_pd'] = training_pd
41     result_M['blind_pd'] = blind_pd
42     result_M['output_dir'] = output_dir
43     result_M['target_label'] = target_label
44     result_M['error_type'] = error_type
45     result_M['n_cores'] = n_cores
46     result_M['num_of_shuffling'] = num_of_shuffling
47     result_M['random_state'] = random_state
48     result_M['no_scaling'] = no_scaling
49     result_M['save_model'] = save_model
50
51     if sort_features:
52         X_input = X_input.sort_values(headers)
53     else:
54         X_input = X_input[headers]
55
56     if not no_scaling:
57         X_input = X_input.select_dtypes('number')
58
59     if not greedy:
60         X_input = X_input.select_dtypes('object')
61
62     if not save_model:
63         joblib.dump(result_M, os.path.join(output_dir, '{}_{}_transformer.sav'.format(timestamp, fname, algorithm)), compress=True)
64
65     X_input = pd.DataFrame(out, columns=headers)
```



Sublime text

```
tenfold_regressor.py
1 #!/usr/bin/python
2
3 # # # # # Yooschan Hyung - yuchan@gmail.com
4 # # # # #
5
6 import pandas as pd
7 import numpy as np
8 import os
9 import time
10 import sys
11 import math
12 import re
13 import argparse
14 from math import sqrt
15 import statsmodels.stats.api as sms
16 import joblib
17
18 from sklearn import patch_sklearn
19 patch_sklearn()
20
21 from sklearn.ensemble import GradientBoostingRegressor, RandomForestRegressor, AdaBoostRegressor, ExtraTreesRegressor
22 from xgboost.sklearn import XGBRegressor
23 from sklearn.gaussian_process import GaussianProcessRegressor
24 from sklearn.neighbors import KNeighborsRegressor
25 from sklearn.model_selection import KFold
26 from sklearn.metrics import mean_squared_error
27 from sklearn.neural_network import MLPRegressor
28 from sklearn.pipeline import Pipeline
29 from sklearn.compose import make_column_transformer
30 from sklearn.preprocessing import StandardScaler, OneHotEncoder
31 from sklearn.impute import SimpleImputer
32
33 timestamp = time.strftime("%Y%m_%d%H%M")
34
35 def run(algorithm, fname, kfold, training_pd, blind_pd, output_dir, target_label, error_type, n_cores, num_of_shuffling, random_state, no_scaling, save_model):
36     result_M = pd.DataFrame()
37     result_M['algorithm'] = algorithm
38     result_M['fname'] = fname
39     result_M['kfold'] = kfold
40     result_M['training_pd'] = training_pd
41     result_M['blind_pd'] = blind_pd
42     result_M['output_dir'] = output_dir
43     result_M['target_label'] = target_label
44     result_M['error_type'] = error_type
45     result_M['n_cores'] = n_cores
46     result_M['num_of_shuffling'] = num_of_shuffling
47     result_M['random_state'] = random_state
48     result_M['no_scaling'] = no_scaling
49     result_M['save_model'] = save_model
50
51     if sort_features:
52         X_input = X_input.sort_values(headers)
53     else:
54         X_input = X_input[headers]
55
56     if not no_scaling:
57         X_input = X_input.select_dtypes('number')
58
59     if not greedy:
60         X_input = X_input.select_dtypes('object')
61
62     if not save_model:
63         joblib.dump(result_M, os.path.join(output_dir, '{}_{}_transformer.sav'.format(timestamp, fname, algorithm)), compress=True)
64
65     X_input = pd.DataFrame(out, columns=headers)
```



Vim

```
tenfold_regressor.py
1 #!/usr/bin/python
2
3 # # # # # Yooschan Hyung - yuchan@gmail.com
4 # # # # #
5
6 import pandas as pd
7 import numpy as np
8 import scipy as sp
9 import time
10 import sys
11 import os
12 import re
13 import argparse
14 from math import sqrt
15 from scipy import stats
16 import joblib
17
18 from sklearn import patch_sklearn
19 patch_sklearn()
20
21 from sklearn.ensemble import GradientBoostingRegressor, RandomForestRegressor, AdaBoostRegressor, ExtraTreesRegressor
22 from xgboost.sklearn import XGBRegressor
23 from sklearn.gaussian_process import GaussianProcessRegressor
24 from sklearn.neighbors import KNeighborsRegressor
25 from sklearn.model_selection import KFold
26 from sklearn.metrics import mean_squared_error
27 from sklearn import svm
28 from sklearn.neural_network import MLPRegressor
29
30 from sklearn.pipeline import make_pipeline
31 from sklearn.compose import make_column_transformer
32 from sklearn.preprocessing import StandardScaler, OneHotEncoder
33 from sklearn.impute import SimpleImputer
34
35 timestamp = time.strftime("%Y%m_%d%H%M")
36
37 def run(algorithm, fname, kfold, training_pd, blind_pd, output_dir, target_label, error_type, n_cores, num_of_shuffling, random_state, no_scaling, save_model, sort_features, greedy=False):
38     result_M = pd.DataFrame()
39     result_M['algorithm'] = algorithm
40     result_M['fname'] = fname
41     result_M['kfold'] = kfold
42     result_M['training_pd'] = training_pd
43     result_M['blind_pd'] = blind_pd
44     result_M['output_dir'] = output_dir
45     result_M['target_label'] = target_label
46     result_M['error_type'] = error_type
47     result_M['n_cores'] = n_cores
48     result_M['num_of_shuffling'] = num_of_shuffling
49     result_M['random_state'] = random_state
50     result_M['no_scaling'] = no_scaling
51     result_M['save_model'] = save_model
52
53     if sort_features:
54         X_input = X_input.sort_values(headers)
55     else:
56         X_input = X_input[headers]
57
58     if not no_scaling:
59         X_input = X_input.select_dtypes('number')
60
61     if not greedy:
62         X_input = X_input.select_dtypes('object')
63
64     if not save_model:
65         joblib.dump(result_M, os.path.join(output_dir, '{}_{}_transformer.sav'.format(timestamp, fname, algorithm)), compress=True)
66
67     X_input = pd.DataFrame(out, columns=headers)
```



# Jupyter Notebook

- A unique combination of interactive code execution, documentation and data analysis features.
- Run on **Google CoLab** server
- Easy to share and less compatibility issues

A screenshot of a Jupyter Notebook cell in Google Colab. The code contains several syntax errors, notably missing parentheses in print statements. The interface includes a sidebar for file management and a toolbar with various icons. A search bar at the bottom is active, suggesting a search for help or solutions.

```
0s
print("This is a test")
print("The following code will raise an error")
print "This is Python 2.X syntax"

SyntaxError: Missing parentheses in call to 'print'. Did you mean print(...)?


[4] print("This is a test")
print("The following code will raise an error")

This is a test
The following code will raise an error

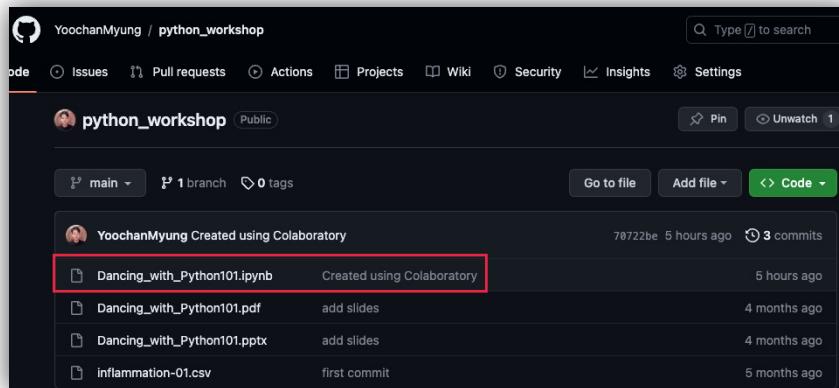
0s
print "This is Python 2.X syntax"

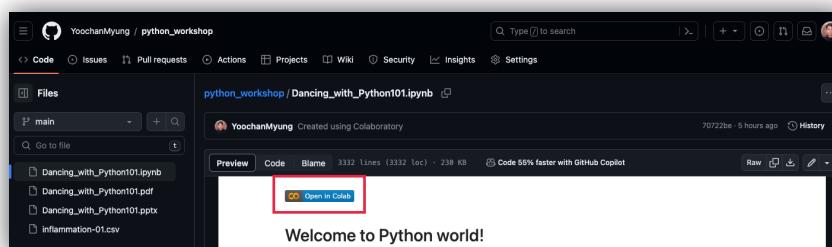
SyntaxError: Missing parentheses in call to 'print'. Did you mean print(...)?
```

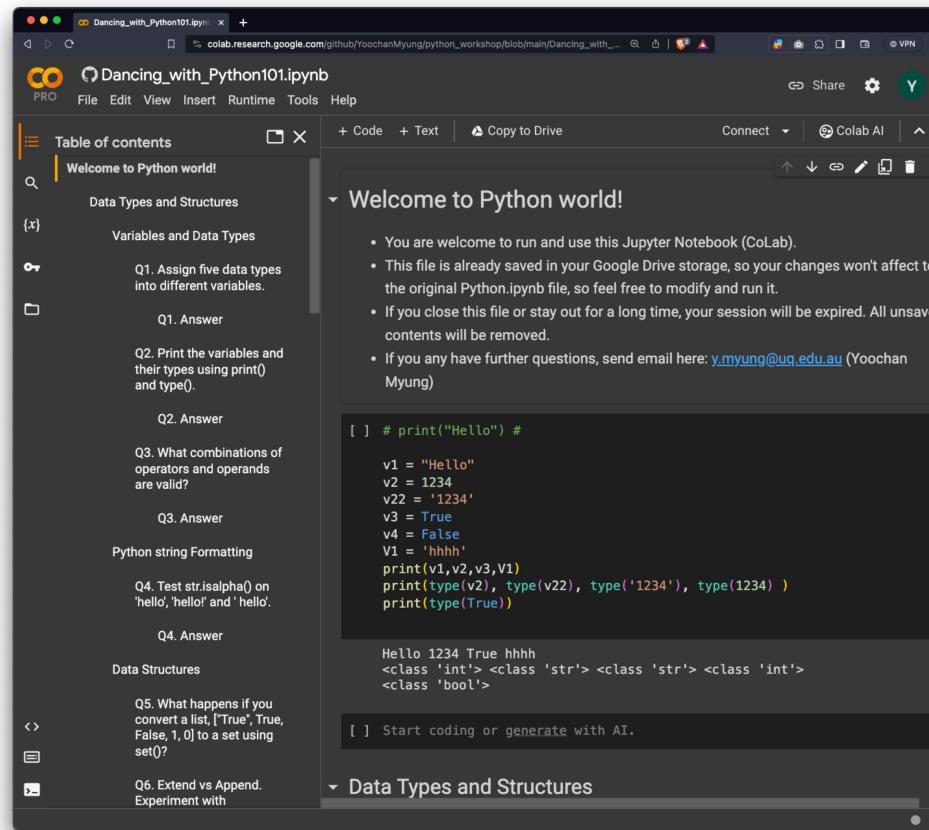
A screenshot of the 'Curated Notebooks' section on Google Colab's website. It features a grid of nine thumbnail cards, each representing a different AI and Machine Learning project. The cards include titles like 'Music Transcription with Transformers', 'Generating Music with Transformers', 'Retraining an Image Classifier', 'Text Classification with Movie Reviews', 'Multilingual Universal Sentence Encoder Q&A Retrieval', 'Create and Train a Custom RL Agent', 'Visualize RL Agent Training on TensorBoard', and 'Hyperparameter Tuning with TensorBoard'. Each card provides a brief description of the project's purpose and how it can be used.

# Coding via Google Colab

[https://github.com/YoochanMyung/python\\_workshop](https://github.com/YoochanMyung/python_workshop)







```
# print("Hello") #
v1 = "Hello"
v2 = 1234
v22 = '1234'
v3 = True
v4 = False
V1 = 'hhhh'
print(v1,v2,v3,V1)
print(type(v2), type(v22), type('1234'), type(1234))
print(type(True))

Hello 1234 True hhhh
<class 'int'> <class 'str'> <class 'str'> <class 'int'>
<class 'bool'>
```

# Class, Function and Method

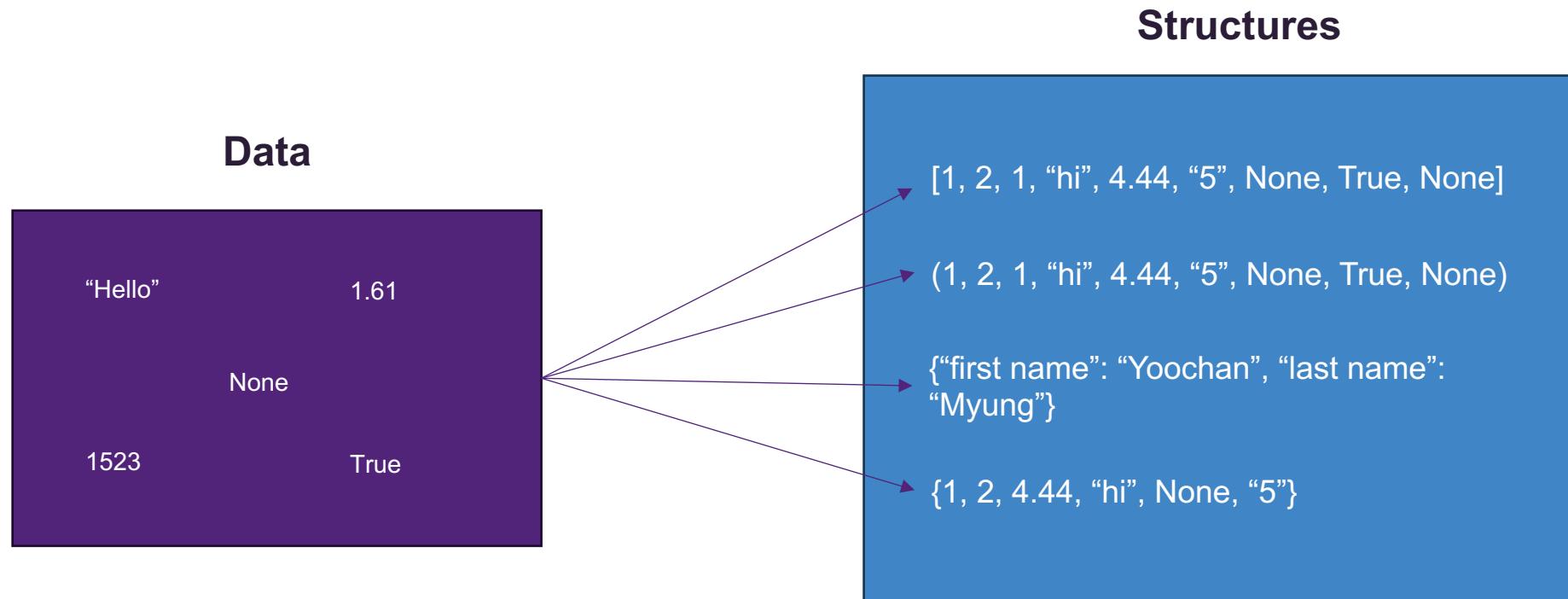
```
● ● ●  
  
def myfunc(name):  
    print("Hello my name is " + name)  
  
class Person:  
    def __init__(self, name, age):  
        self.name = name  
        self.age = age  
  
    def myfunc(self): # Actually, method  
        print("Hello my name is " + self.name)
```

**Function:** A block of code that performs a task.

**Method:** A function attached to an object.

**Class:** A piece of code (object) with properties (data) and behaviors (methods).

# Data Types and Structures



# Data Types and Variables: RULES

**Variables:** we need to assign (=) our data to a variable.

- can contain letters, digits and underscores(\_).
- Case-sensitive (`v1` ≠ `V1`)
- First character must be letter or an underscore(\_).
- Cannot use Python reserved keywords as variable names

## Types of data

- **integer number (int):** 1, 10, 100
- **floating point number (float):** 3.14
- **string (str):** "Hi", "1", "3.14"
- **boolean (bool):** True (1), False (0)
- **NoneType (NoneType):** None

### reserved keywords

False, class, finally, is, return  
 None, continue, for, lambda, try  
 True, def, from, nonlocal, while  
 and, del, global, not, with  
 as, elif, if, or, yield  
 assert, else, import, pass  
 break, except, in, raise

### built-in functions

list, set, dict, tuple, str, int, float,  
 bool

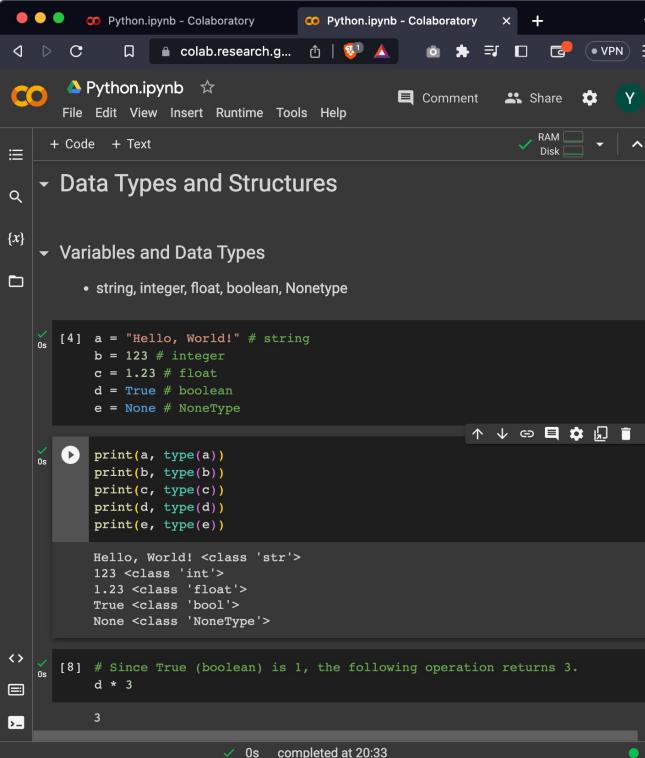
'=' , an assignment operator  
 '==' , a relational operator

# Data Types and Variables

**Task 1.** Assign the **five data types** into different variables

**Task 2.** Print the variables and their types using `print()` and `type()` functions

**Task 2A.** Use 'functions' to convert data types



The screenshot shows a Jupyter Notebook interface with two cells. Cell [4] contains code to assign values to variables a through e based on their data types. Cell [5] contains code to print each variable and its type using the `type()` function. The output shows the variable names followed by their types: string, int, float, bool, and NoneType respectively.

```

[4]: a = "Hello, World!" # string
      b = 123 # integer
      c = 1.23 # float
      d = True # boolean
      e = None # NoneType

[5]: print(a, type(a))
      print(b, type(b))
      print(c, type(c))
      print(d, type(d))
      print(e, type(e))

Hello, World! <class 'str'>
123 <class 'int'>
1.23 <class 'float'>
True <class 'bool'>
None <class 'NoneType'>

```

A function is a block of code which only runs when it called.



The screenshot shows a Jupyter Notebook interface with a cell containing code to demonstrate type annotations. The code defines variables a through d with their respective types: str, int, float, and bool. A callout box on the right explains that this ensures data is explicitly defined with a specific type.

```

a = str("Hello, World!")
b = int(123)
b = int("123") ★
c = float(1.23)
c = float("1.23") ★
# for False
d = bool(0)
d = bool(False)
d = bool(None)
d = bool(100-100)
d = bool(0/100)
# for True
d = bool("ANYTHING except 0,False, None")

```

This ensures that your "data" is explicitly defined with a specific type.

```

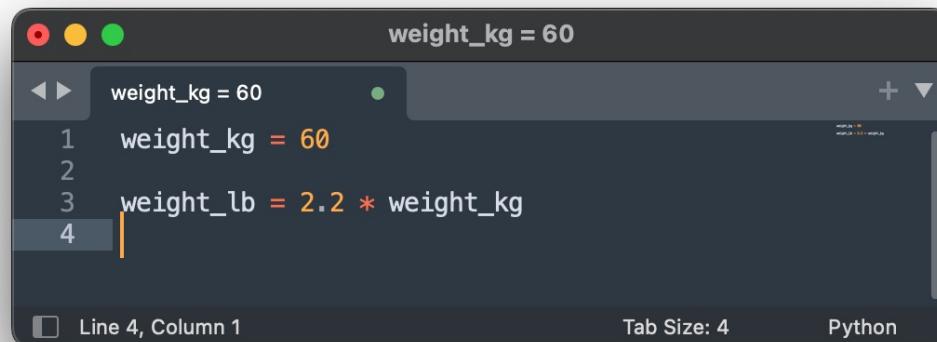
55 <class 'int'>
55 <class 'str'>
1.23 <class 'str'> 1.23 <class 'float'>

```

# Data Types and Variables

## Task 3. What combinations of operators and operands are valid?

Try arithmetic operators (+, -, /, \*) with five data types.



```
weight_kg = 60
1 weight_kg = 60
2
3 weight_lb = 2.2 * weight_kg
4
```

Line 4, Column 1      Tab Size: 4      Python

**float \* int**



```
untitled
1
2 guest_id = '002'
3 guest_id = 'Guest' + gueest_id
4
5
```

Line 5, Column 1      Tab Size: 4      Python

**str + str**

# Python Data Types and Variables

## Task 3. What combinations of operators and operands are valid?

- ‘int or float’ and ‘int or float’ : +, -, /, \*
- ‘str’ and ‘str’: +
- ‘str’ and ‘int’: \*
- `int or float` and `bool` : +, -, \*, / (except division by False (0))
  - because True == 1, False == 0

# Data Types and Operators

Numeric Types – int, float and complex

Operation	Meaning	Result
<code>1 + 4</code>	sum	5
<code>1 - 4</code>	difference	-3
<code>1 * 4</code>	product	4
<code>1 / 4, 11 / 5</code>	quotient of 1 and 4	0.25, 2.2
<code>1 // 4, 11 // 5</code>	floored quotient of 1 and 4	0, 2
<code>1 % 4</code>	remainder of 1 / 4	1
<code>abs(-4)</code>	absolute	4
<code>int(3.14), int(3.9)</code>	convert to integer	3
<code>float(3)</code>	convert to floating point	3.0
<code>pow(2,3)</code>	2 to the power 3	8
<code>2 ** 3</code>	2 to the power 3	8

# String Formatting: Print()

The basic `print()` function does not provide direct control over the decimal point precision of floating-point numbers.

**% operator:**

- `'%s` for string, `'%d` for integer and `'%f` for float.

**`format()` : since Python 3.0**

**f-strings: since Python 3.6**

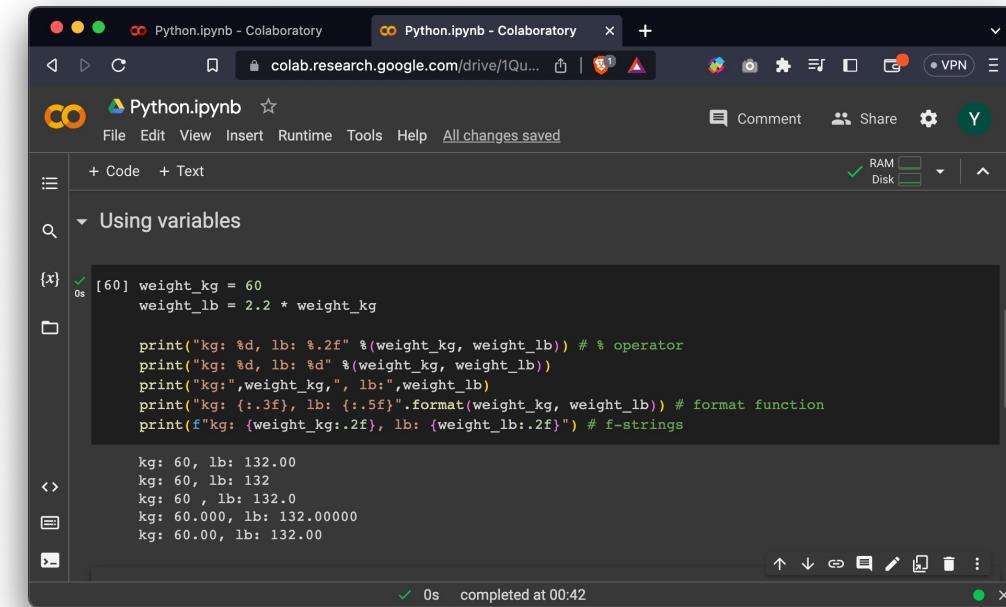
- `{:d}` for integer, `{:f}` for a floating-point number with two decimal places

{:.2f}

`:` the start of the format specifier

`2` specifies the precision or the number of decimal places to display

`f` the type specifier for floating-point numbers



The screenshot shows a Google Colab notebook titled "Python.ipynb". The code cell contains the following Python code:

```
[60] weight_kg = 60
      weight_lb = 2.2 * weight_kg

      print("kg: %d, lb: %.2f" %(weight_kg, weight_lb)) # % operator
      print("kg: %d, lb: %d" %(weight_kg, weight_lb))
      print("kg:", weight_kg, ", lb:", weight_lb)
      print("kg: {:.3f}, lb: {:.5f}".format(weight_kg, weight_lb)) # format function
      print(f"kg: {weight_kg:.2f}, lb: {weight_lb:.2f}") # f-strings
```

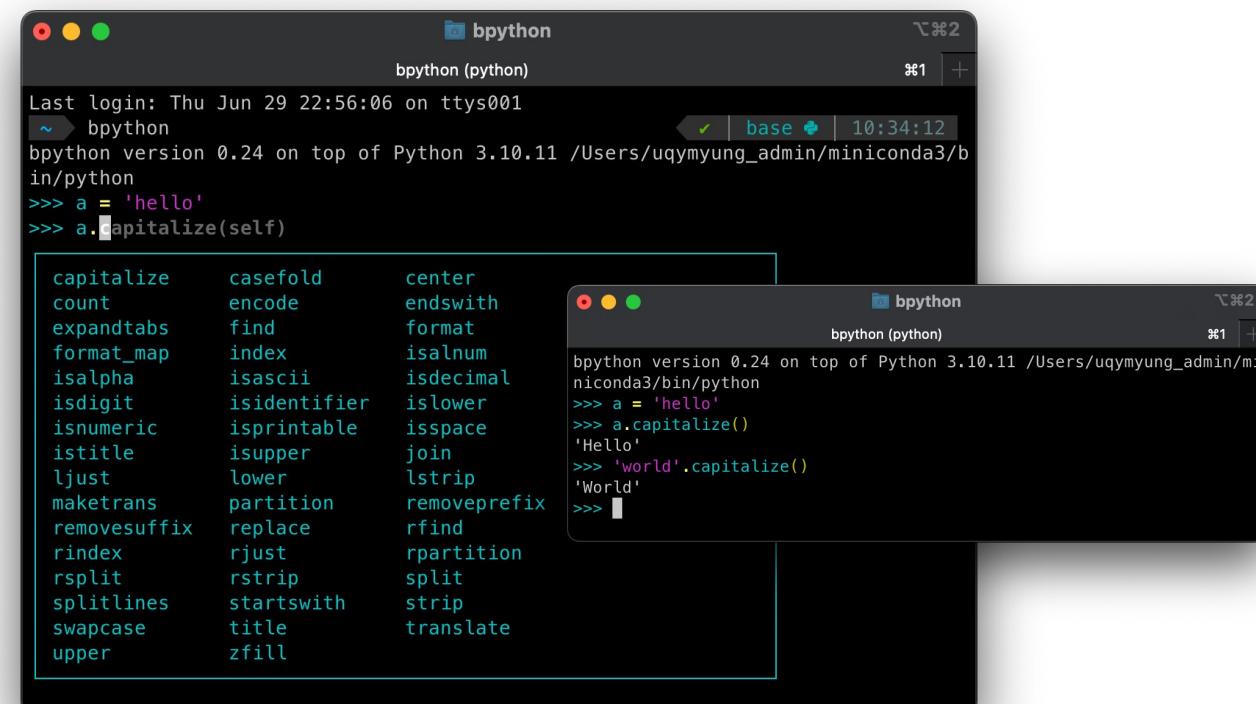
The output of the code is displayed below the code cell:

```
kg: 60, lb: 132.00
kg: 60, lb: 132
kg: 60 , lb: 132.0
kg: 60.000, lb: 132.00000
kg: 60.00, lb: 132.00
```

# String functions

**Task 4.** Test `isalpha()` on ‘hello’, ‘hello!’, ‘hell0’, and ‘ hello’.

function	example	output
<code>str.capitalize(self)</code>		Hello
<code>str.upper(self)</code>		HELLO
<code>str.lower(self)</code>		hello
<code>str.count(obj, sub[, start[, end]])</code>	<code>str.count('l')</code> <code>str.count('l', 2)</code> <code>str.count('l', 3)</code>	2 2 1
<code>str.isalpha(self)</code>		True
<code>str.isnumeric(self)</code>		False
<code>str.strip(chars)</code>		hello
<code>str.lstrip(chars)</code>	<code>str.lstrip('h')</code>	ello
<code>str.rstrip(chars)</code>	<code>str.rstrip('h')</code>	hello
<code>str.startswith(obj, [, start[, end]])</code>	<code>str.startswith('h')</code>	True
<code>str.endswith(obj, [, start[, end]])</code>	<code>str.endswith('h')</code>	False
<code>str.replace(self, old, new, count)</code>	<code>str.replace('l', 'a', 2)</code>	heaaoo



The image shows two side-by-side Python consoles. The left console is titled 'bpython' and shows the following session:

```
Last login: Thu Jun 29 22:56:06 on ttys001
~> bpython
bpython version 0.24 on top of Python 3.10.11 /Users/uqymyung_admin/miniconda3/bin/python
>>> a = 'hello'
>>> a.capitalize(self)
```

The right console is also titled 'bpython' and shows:

```
bpython version 0.24 on top of Python 3.10.11 /Users/uqymyung_admin/miniconda3/bin/python
>>> a = 'hello'
>>> a.capitalize()
'Hello'
>>> 'world'.capitalize()
'World'
>>> 
```

A vertical list of string methods is overlaid on the right side of the left console's window, with a blue box highlighting the `capitalize` method.

# TIME FOR A BREAK

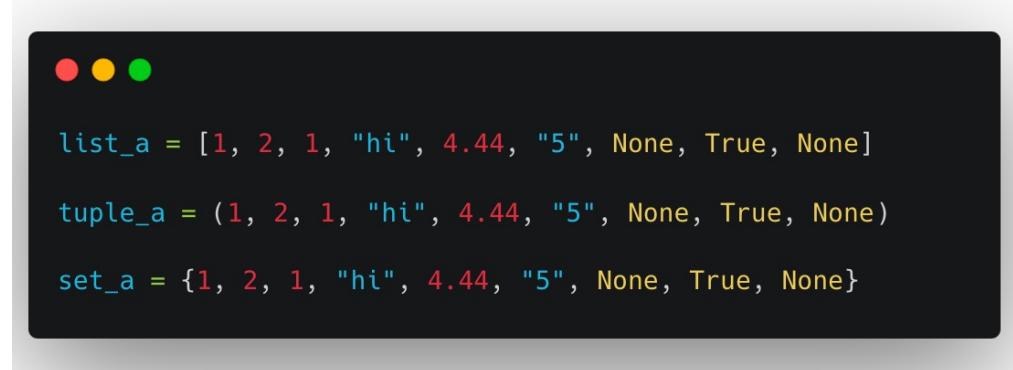
**Please be ready for using Google CoLab and ChatGPT!**

[https://github.com/YoochanMyung/python\\_workshop](https://github.com/YoochanMyung/python_workshop)

<https://openai.com/blog/chatgpt>

# Data Structures

- **List:** ordered and mutable collections of elements
  - [1, 2, 1, "hi", 4.44, "5", None, True, None]
- **Tuple:** ordered and immutable collections of elements
  - (1, 2, 1, "hi", 4.44, "5", None, True, None)
- **Set:** unordered and mutable collections of unique elements
  - {1, 2, 4.44, "hi", None, "5"}



```
list_a = [1, 2, 1, "hi", 4.44, "5", None, True, None]
tuple_a = (1, 2, 1, "hi", 4.44, "5", None, True, None)
set_a = {1, 2, 1, "hi", 4.44, "5", None, True, None}
```

list\_a[0] =  
list\_a[:4] =  
list\_a[4:] =  
list\_a[-1] =

**Task 5.** What happens if you convert a list, ["True", True, False, 1, 0] to a set using `set()`?

# List

an iterable is an object capable of returning its members one at a time.

List is a mutable and ordered sequence class.

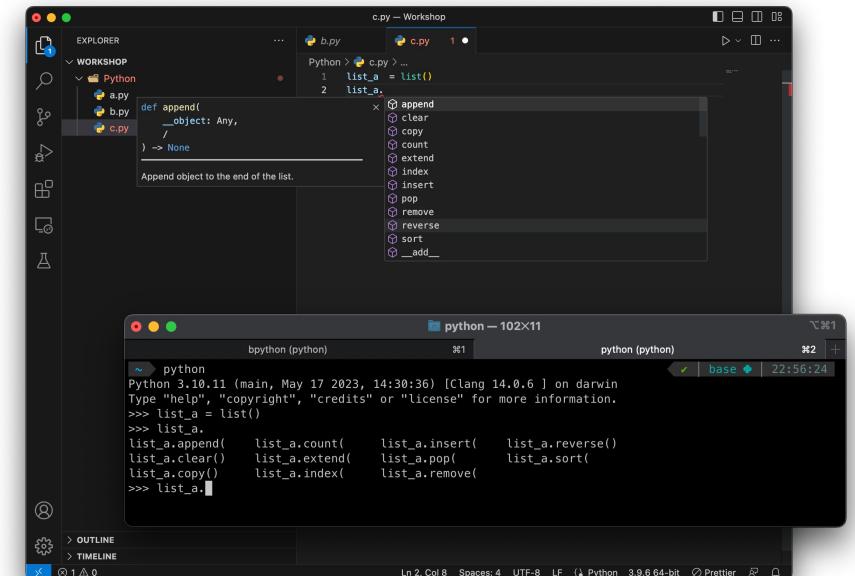
Mutable and immutable classes have some common functions.

- `append(self, object)`: Append object to the end of the list.
- `extend(self, iterable)`: Extend list by appending elements from the iterable.
- `insert(self, index, object)`: Insert object before index
- `clear(self)`: Remove all items from list.
- `index(self, value, start, stop)`: Return first index of value or ValueError if the value is not present.
- `remove(self, value)`: Remove first occurrence of value or raise ValueError if the value is not present.
- `sort(self, \*, key=None, reverse=False)`: Sort the list in ascending order

```
● ● ●

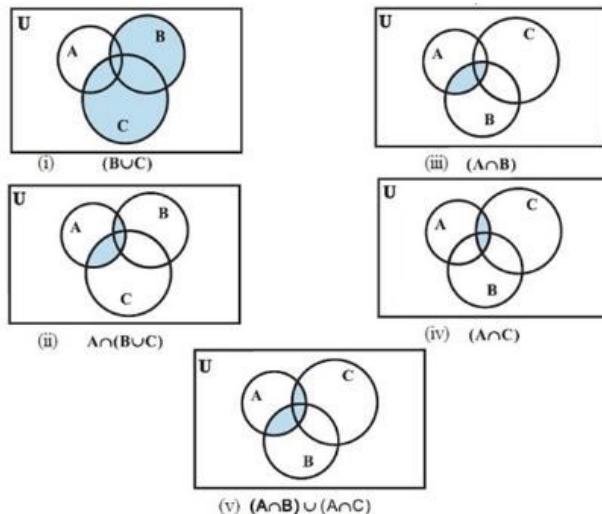
list_a = [1, 2, 1, "hi", 4.44, "5", None, True, None]
list_b = [None, True, None, 1, 2, 1, "hi", 4.44, "5"]
```

Task 6. What's the difference between append and extend?



# Set

Set is an unordered collection of distinct hashable objects.



The **hash()** function generates a hash value based on the contents of an object.

**Task 9.** What is the has value of “A” and “a”?

<https://guidancecorner.com/set-theory/>

**Hash value:** `hash(object)` returns an integer that is unique to the object.  
 Same hash values → same objects.  
 This can be used for data lookup, data integrity checks and object uniqueness testing.



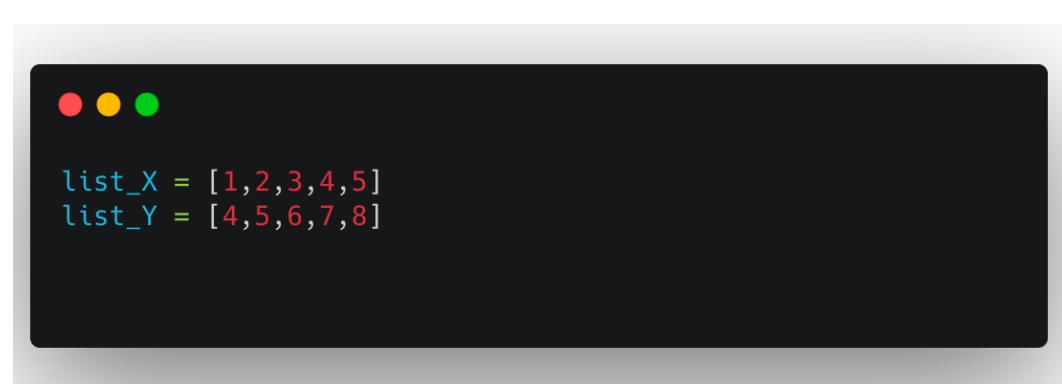
```
list_a = [1, 2, 1, "hi", 4.44, "5", None, True, None]

hash(1) # 1
hash(2) # 2
hash(1) # 1
hash("hi") # 2340380648754954203
hash(4.44) # 1014570924054026244
hash("5") # 7363345458445120069
hash(None) # -9223372036583882765
hash(True) # 1
hash(None) # -9223372036583882765
```

# Set

**Task 9A.** What are the differences and common elements between of “list\_X” and “list\_Y”?

- `set_A.union(set_B) = set_A | set_B`
- `set_A.difference(set_B) = set_A – set_B`
- `set_A.intersection(set_B) = set_A & set_B`
- `set_A.issubset(set_B) = set_A <= set_B`
- `set_B.issuperset(set_A) = set_A <= set_A`
- `set_A.symmetric_difference(set_B) = set_A ^ set_B`



```
● ● ●  
list_X = [1,2,3,4,5]  
list_Y = [4,5,6,7,8]
```

# Set

**Task 9A.** What are the differences and common elements between of “list\_X” and “list\_Y”?

- `set_A.union(set_B) = set_A | set_B`
- `set_A.difference(set_B) = set_A – set_B`
- `set_A.intersection(set_B) = set_A & set_B`
- `set_A.issubset(set_B) = set_A <= set_B`
- `set_B.issuperset(set_A) = set_A <= set_A`
- `set_A.symmetric_difference(set_B) = set_A ^ set_B`

```
● ● ●

list_X = [1,2,3,4,5]
list_Y = [4,5,6,7,8]
```

```
● ● ●

X_n_Y = set(list_X).intersection(set(list_Y))
X_Y = set(list_X).difference(set(list_Y))
Y_X = set(list_Y).difference(set(list_X))

#X_n_Y = set(list_X) & set(list_Y)
#X_Y = set(list_X) - set(list_Y)
#Y_X = set(list_Y) - set(list_X)

print(X_n_Y)
print(X_Y, Y_X)
```

# Dictionary

A mapping object maps hashable values to arbitrary objects.

keys                    values

```
● ● ●
a = dict(one=1, two=2, three=3)

b = {'one': 1, 'two': 2, 'three': 3}

c = dict(zip(['one', 'two', 'three'], [1, 2,
3]))
d = dict([('two', 2), ('one', 1), ('three', 3)])

e = dict({'three': 3, 'one': 1, 'two': 2})

f = dict({'one': 1, 'three': 3}, two=2)

a == b == c == d == e == f
```

key                    value  
**{'ID-1': 'y.myung', 'ID-2': 'e.park', ...}**

**{'ID-1': {'name': 'Eric', 'age': 24}, ...}**

**dict.keys(), dict.values() and dict.items()** return the iterable of keys, values and items (keys and values), respectively.

```
● ● ●
a = dict(one=1,two=2,three=3)

a['one'] # 1
a.get('one') # 1
```

**Task 10.** What happens if we add same keys?

**Task 11.** What is difference between using slicing (`[]`) and `get()` method?

# Compound statements

An essential part of the Python programming, which control the execution of other statements in some way.

[**if**, while, **for**, try, with, match, funcdef, classdef, **async\_with**, **async\_for**, **async\_func**]

The `if` statement runs **<statement>** if the **<expr>** satisfies a condition.

Operation	Meaning
<	strictly less than
<=	less than or equal
>	strictly greater than
>=	greater than or equal
==	equal
!=	not equal
is	object identity (id)
is not	negate object identity (id)

```
● ● ●
x = -0.5
if 0 < x < 101:
    print(f"{x} is positive, but less than 101.")
elif x == 0:
    print(f"{x} is zero.")
elif x < 0:
    print(f"{x} is negative.")
else:
    print(f"{x} is greater than or equal to 101.")
```

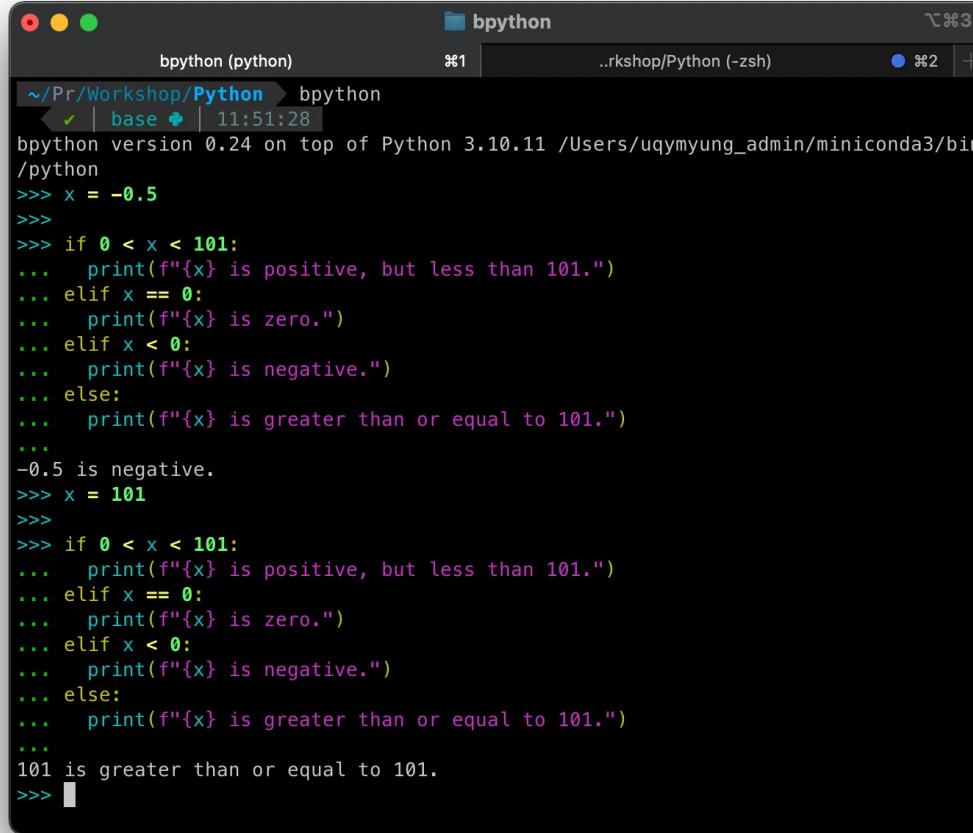
The **id()** function returns a unique identifier based on the memory address of an object.

```
● ● ●
if <expr>:
    <statement>
else:
    <statement>

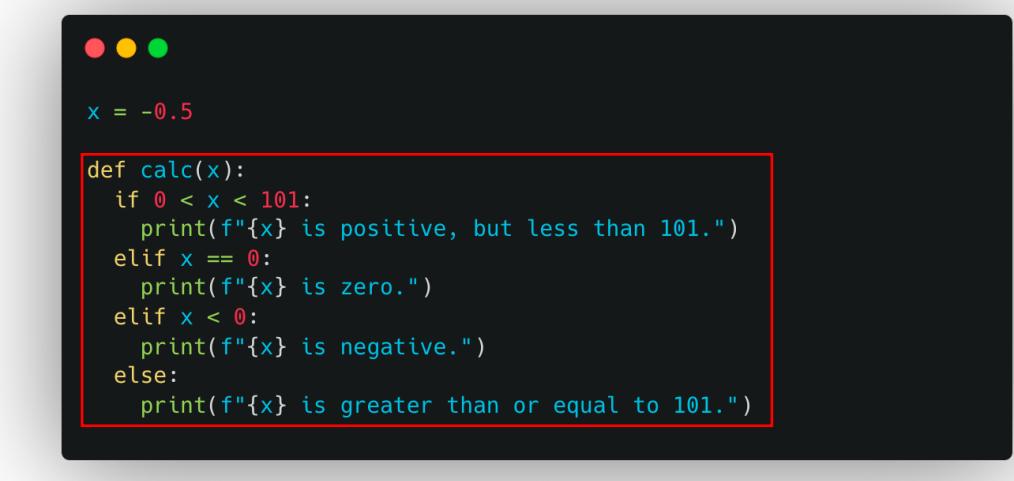
if <expr>:
    <statement>
elif <expr>:
    <statement>
else:
    <statement>

if <expr>:
    <statement>
elif <expr>:
    <statement>
else:
    <statement>
```

# IF and Function

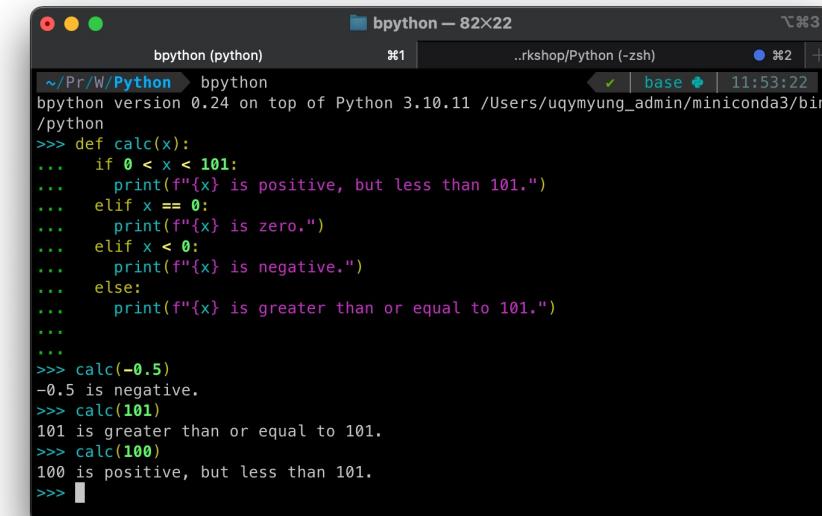


```
bpython (python)
~/.Pr/Workshop/Python > bpython
✓ | base ✘ | 11:51:28
bpython version 0.24 on top of Python 3.10.11 /Users/uqymyung_admin/miniconda3/bin/python
>>> x = -0.5
>>>
>>> if 0 < x < 101:
...     print(f"{x} is positive, but less than 101.")
... elif x == 0:
...     print(f"{x} is zero.")
... elif x < 0:
...     print(f"{x} is negative.")
... else:
...     print(f"{x} is greater than or equal to 101.")
...
-0.5 is negative.
>>> x = 101
>>>
>>> if 0 < x < 101:
...     print(f"{x} is positive, but less than 101.")
... elif x == 0:
...     print(f"{x} is zero.")
... elif x < 0:
...     print(f"{x} is negative.")
... else:
...     print(f"{x} is greater than or equal to 101.")
...
101 is greater than or equal to 101.
>>> 
```



```
x = -0.5

def calc(x):
    if 0 < x < 101:
        print(f"{x} is positive, but less than 101.")
    elif x == 0:
        print(f"{x} is zero.")
    elif x < 0:
        print(f"{x} is negative.")
    else:
        print(f"{x} is greater than or equal to 101.")
```



```
bpython (python)
~/.Pr/W/Python > bpython
✓ | base ✘ | 11:53:22
bpython version 0.24 on top of Python 3.10.11 /Users/uqymyung_admin/miniconda3/bin/python
>>> def calc(x):
...     if 0 < x < 101:
...         print(f"{x} is positive, but less than 101.")
...     elif x == 0:
...         print(f"{x} is zero.")
...     elif x < 0:
...         print(f"{x} is negative.")
...     else:
...         print(f"{x} is greater than or equal to 101.")
...
...
>>> calc(-0.5)
-0.5 is negative.
>>> calc(101)
101 is greater than or equal to 101.
>>> calc(100)
100 is positive, but less than 101.
>>> 
```

# Function definitions

A function is an executable statement.

The function definition does not execute the function body; this gets executed only when the function is called.

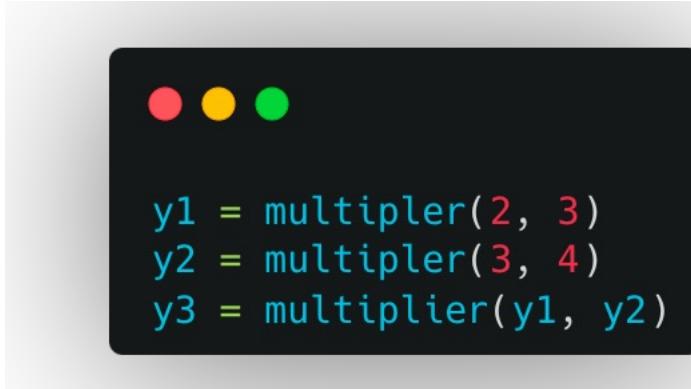
name            parameter values

```
def multiplier(x1, x2):
    y = x1 * x2
    print(y)
    return y
```

4 spaces or tab

## Indentation Levels:

Indentation is used to denote nested blocks of code, such as loops, conditionals, and function definitions. Each nested block should be indented by one-level deeper than its parent block.



```
y1 = multiplier(2, 3)
y2 = multiplier(3, 4)
y3 = multiplier(y1, y2)
```

# Function to Class

A class is an executable statement,

```
● ● ●
def calculator(x1, x2):
    def multiplier(x1, x2):
        return x1 * x2
    def summation(x1, x2):
        return x1+x2
    def subtraction(x1, x2):
        return x1-x2

    return multiplier(x1,x2), summation(x1,x2), subtraction(x1,x2)
```

**function**

a **class** is a template for creating objects, while a **function** is a block of reusable code that performs a specific task

```
● ● ●
class calculator:
    def __init__(self):
        print("You initiated a calculator")
        self.x1 = 0
        self.x2 = 0
        self.history = list()

    def set_values(self, x1, x2):
        self.x1 = x1
        self.x2 = x2
        self.history.append([x1,x2])

    def multiplier(self):
        return self.x1 * self.x2
    def summation(self):
        return self.x1+self.x2
    def subtraction(self):
        return self.x1-self.x2

    def get_all_values(self):
        return self.multiplier(), self.summation(),
        self.subtraction()
```

**constructor**

**method**

# FOR statement

The `for` statement is used to iterate over the elements of a sequence or other iterable object.

A `break` statement executed in the first suite terminates the loop.



```
● ● ●
for i in range(3): # [0,1,2]
    print(i)
    break
```

**Task 13.** Given the list below, how can we retrieve a new list that contains only words starting with the letter “A”?

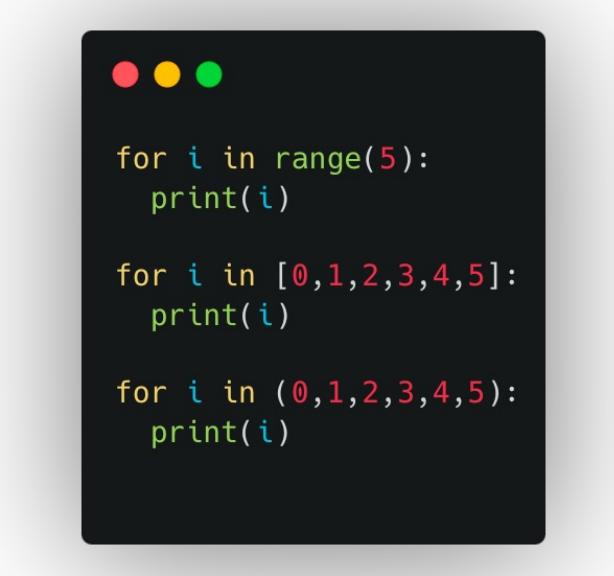


```
● ● ●
list_of_queries = ['ATC', 'ATT', 'TAC', 'ACT', 'AAA', 'ATA',
'CTT', 'TTT', 'CAC']

# Hint 1: str.startswith('A')

# Hint 2: IF statement

# Hint 3: For statement
```



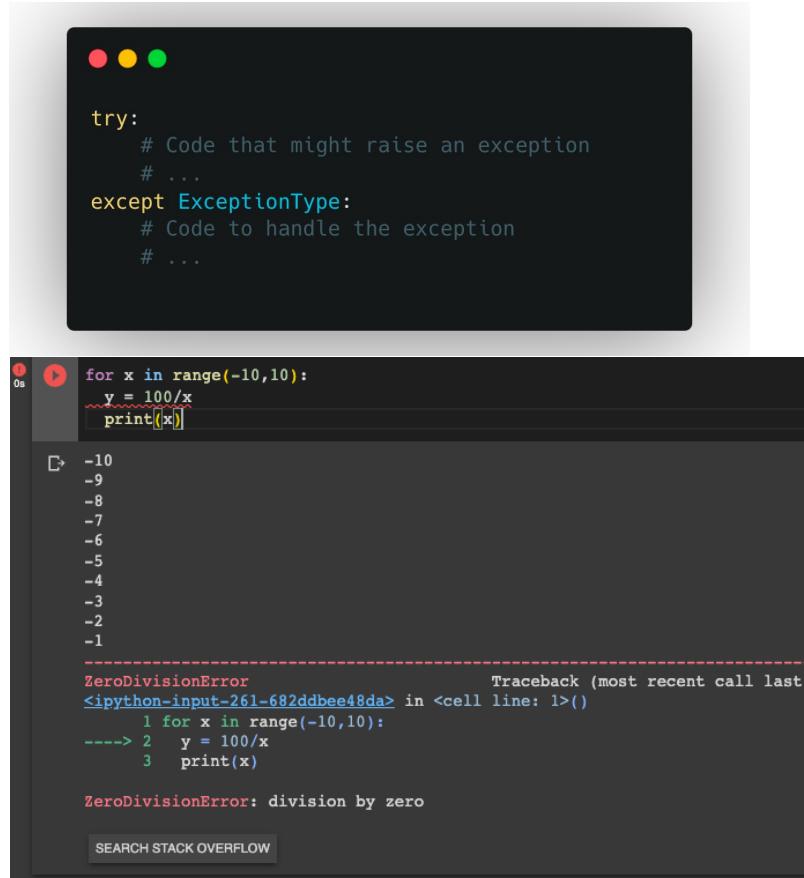
```
● ● ●
for i in range(5):
    print(i)

for i in [0,1,2,3,4,5]:
    print(i)

for i in (0,1,2,3,4,5):
    print(i)
```

# TRY / EXCEPT statement

Try and Except are used to implement exception handling, which allows you to manage and respond to errors or exceptional conditions that may occur during the execution of a program.



```

try:
    # Code that might raise an exception
    # ...
except ExceptionType:
    # Code to handle the exception
    # ...

```

for x in range(-10,10):
y = 100/x
print(x)

-10  
-9  
-8  
-7  
-6  
-5  
-4  
-3  
-2  
-1

ZeroDivisionError Traceback (most recent call last)
<ipython-input-261-682ddbee48da> in <cell line: 1>()
 1 for x in range(-10,10):
----> 2 y = 100/x
 3 print(x)

ZeroDivisionError: division by zero

[SEARCH STACK OVERFLOW](#)



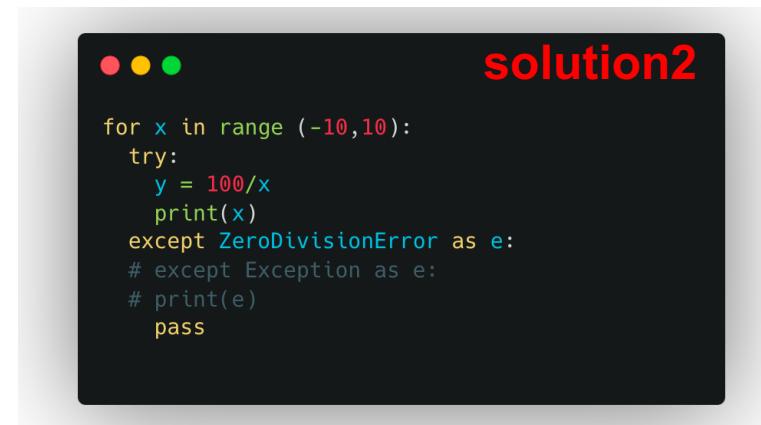
**solution1**

```

try:
    for x in range (-10,10):
        y = 100/x
        print(x)
except ZeroDivisionError as e:
# except Exception as e:
#     print(e)
    pass

```

**Check**  
**CoLab**



**solution2**

```

for x in range (-10,10):
    try:
        y = 100/x
        print(x)
    except ZeroDivisionError as e:
# except Exception as e:
#     print(e)
        pass

```

# TIME FOR A BREAK

**Please be ready for using Google CoLab and ChatGPT!**

[https://github.com/YoochanMyung/python\\_workshop](https://github.com/YoochanMyung/python_workshop)

<https://openai.com/blog/chatgpt>

# Numpy array

[Check  
CoLab](#)

0	2	3
1	5	10

list\_a

8	2	1
25	1	1

list\_b

```
● ● ●
list_a = [[0,2,3],[1,5,10]] # 2D list
list_b = [[8,2,1],[25,1,1]] # 2D list
print(list_a+list_b) # append list_a, list_b

array_a = np.array(list_a) # 2 x 3 matrix
array_b = np.array(list_b) # 2 x 3 matrix
```

**Q. How to select “5” using slicing?**

**Q. How to append arrays?**

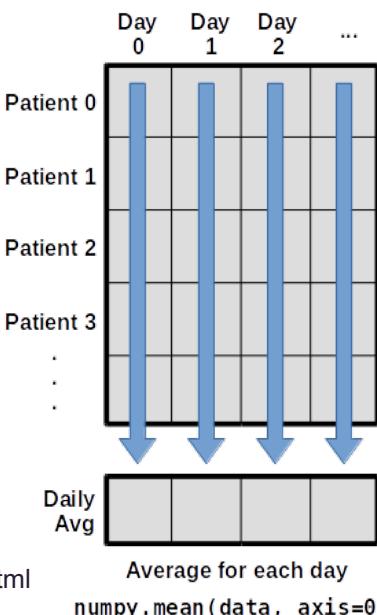
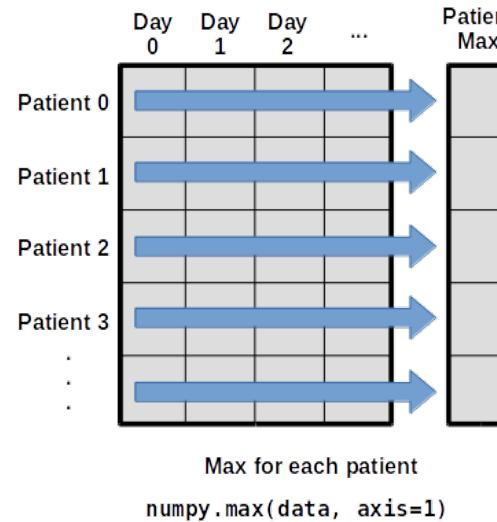
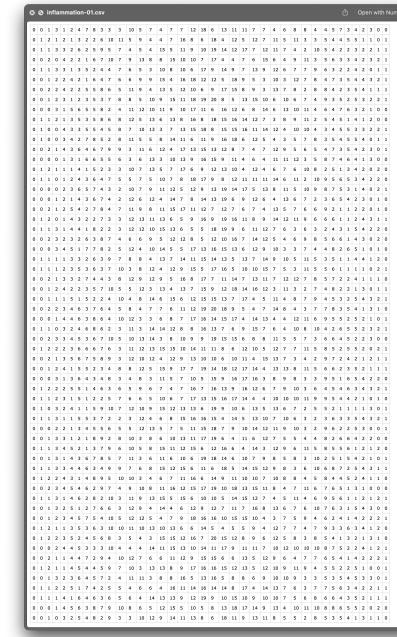
0	2	3
1	5	10
8	2	1
25	1	1

# Analysing Patient Data (CSV)

## Requirements:

- a clinical trial inflammation data (csv); 60 arthritis patients for 40 days
- a library called NumPy, Numerical Python, for slicing and analysing data

**Task 14.** How to get “MIN/MAX/MEAN/STD” per patient or per day across all days or all patients?

A screenshot of a CSV file titled "inflammation-01.csv" showing a grid of numerical data. The columns represent different patients and the rows represent different days. The data consists of integers ranging from 0 to 100.

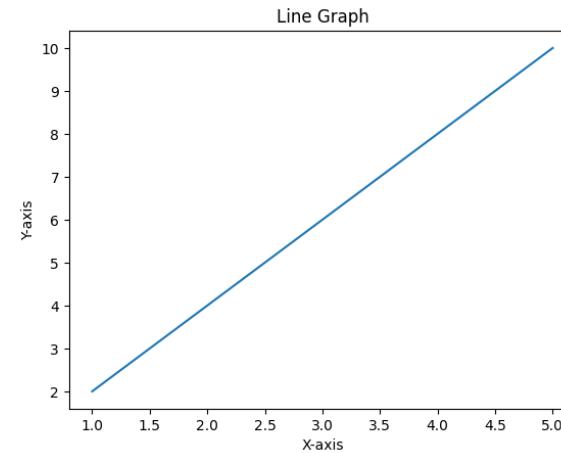
**numpy.max()**  
**numpy.min()**  
**numpy.mean()**  
**numpy.std()**

# Data Visualisation - Matplotlib

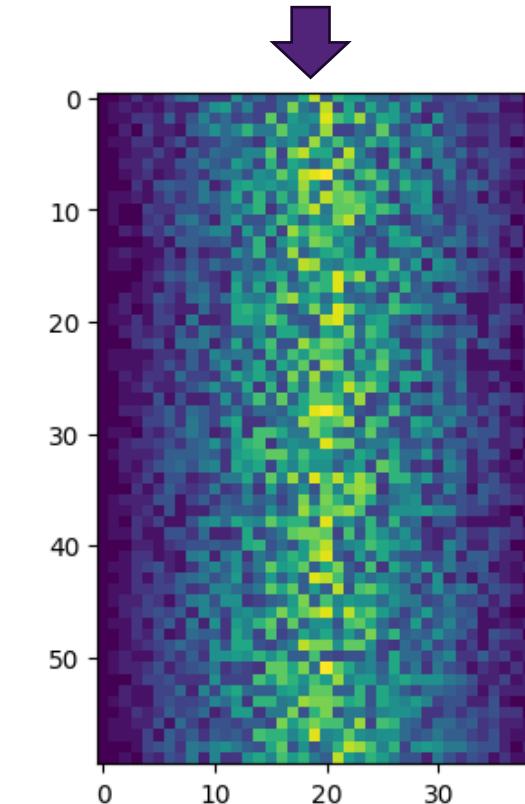
a popular Python library used for creating data visualisations.

- line plots, scatter plots, bar plots, histograms, pie charts and more.

```
● ● ●  
  
import matplotlib.pyplot as plt  
  
x = [1, 2, 3, 4, 5]  
y = [2, 4, 6, 8, 10]  
  
plt.plot(x, y)  
plt.xlabel('X-axis')  
plt.ylabel('Y-axis')  
plt.title('Line Graph')  
plt.show()
```



```
● ● ●  
  
image = matplotlib.pyplot.imshow(loaded_data)
```



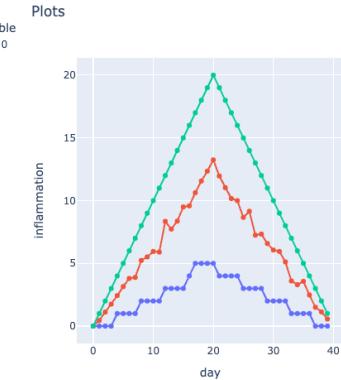
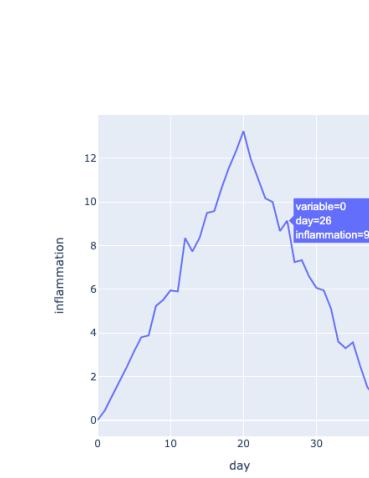
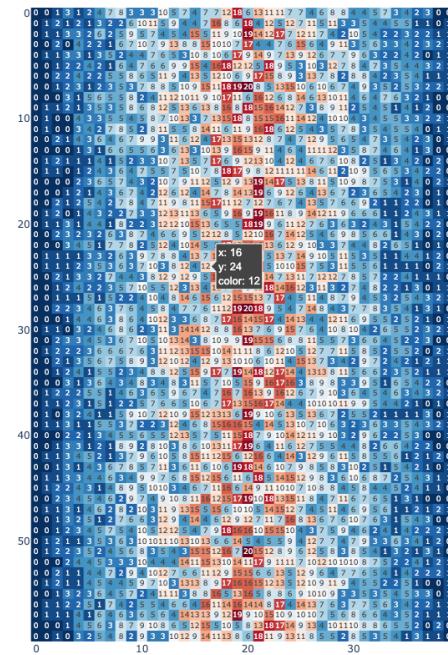
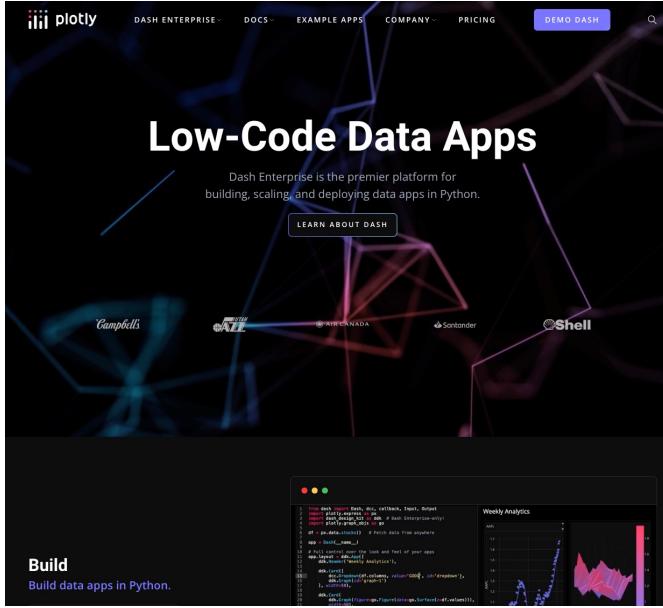
Check  
CoLab

# Data Visualisation - Plotly

a Python library that allows for interactive data visualisation and provide wide range of graphing capabilities.



Check  
CoLab



# ChatGPT

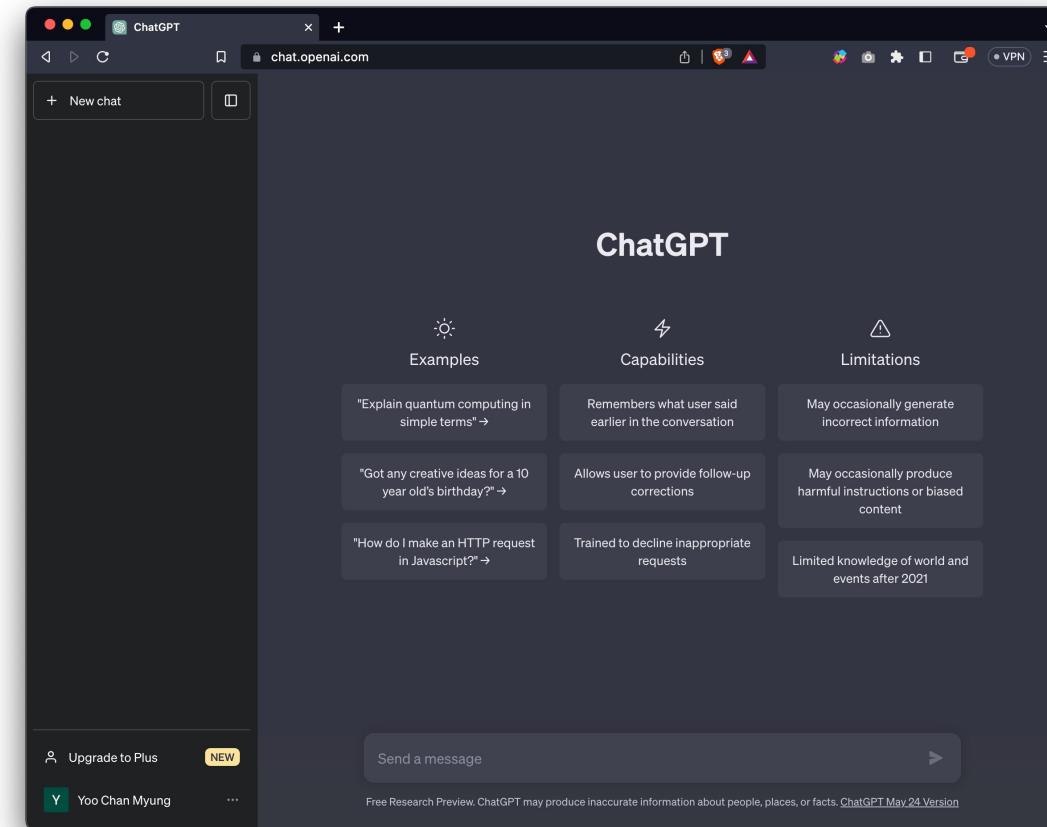
- ChatGPT is a language-based model designed for learning on sequential data.
- Due to the temperature parameter, ChatGPT offers flexibility and creativity, resulting in potentially different answers over time. (use “regenerate response”)

#1 Writing Code and Generating data

#2 Debugging Code

#3 Converting Code

You need an openAI account for using ChatGPT.



# ChatGPT

**Temperature give flexibility and creativity.**

## **What is ChatGPT? (1<sup>st</sup> try)**

ChatGPT is a language model developed by OpenAI. It stands for "Chat Generative Pre-trained Transformer." It is built upon the GPT (Generative Pre-trained Transformer) architecture, specifically GPT-3.5, which is designed to generate human-like text based on the given input. ChatGPT can understand and generate natural language responses, making it useful for a wide range of conversational tasks and applications. It can be used to answer questions, provide explanations, assist with writing, simulate dialogue, and more.

## **What is ChatGPT? (2<sup>nd</sup> try)**

ChatGPT is a language model developed by OpenAI. It is part of the GPT (Generative Pre-trained Transformer) family of models and is based on GPT-3.5 architecture. GPT-3.5 is a highly advanced and powerful language model that has been trained on a large corpus of text data from the internet. It can generate human-like text and respond to prompts in a wide range of topics and languages.

ChatGPT can understand and generate natural language, making it capable of engaging in conversations, answering questions, generating code, providing explanations, and much more. It has been designed to be versatile and useful in various applications, including customer service, content generation, language translation, and more. Users can interact with ChatGPT through API calls, integrating it into their applications, or by accessing it through platforms like ChatGPT website or other interfaces.

# ChatGPT: Writing Python code

**Query:** Create **Python** code to generate a **heatmap** visualization using **Plotly**, representing data from **60 patients**, which includes their **blood pressure, temperature, height, weight, and heart rate**.

```

You
Create Python code to generate a heatmap visualization using Plotly, representing data from 60 patients, which includes their blood pressure, temperature, height, weight, and heart rate.

ChatGPT
To create a heatmap visualization using Plotly for data from 60 patients, including their blood pressure, temperature, height, weight, and heart rate, you can follow these steps. First, make sure you have Plotly installed. You can install it using `pip` if you haven't already:

bash
Copy code
pip install plotly

Now, you can create the Python code to generate the heatmap:

python
Copy code
import plotly.express as px
import pandas as pd
import numpy as np

# Generate synthetic patient data
np.random.seed(42)
n_patients = 60

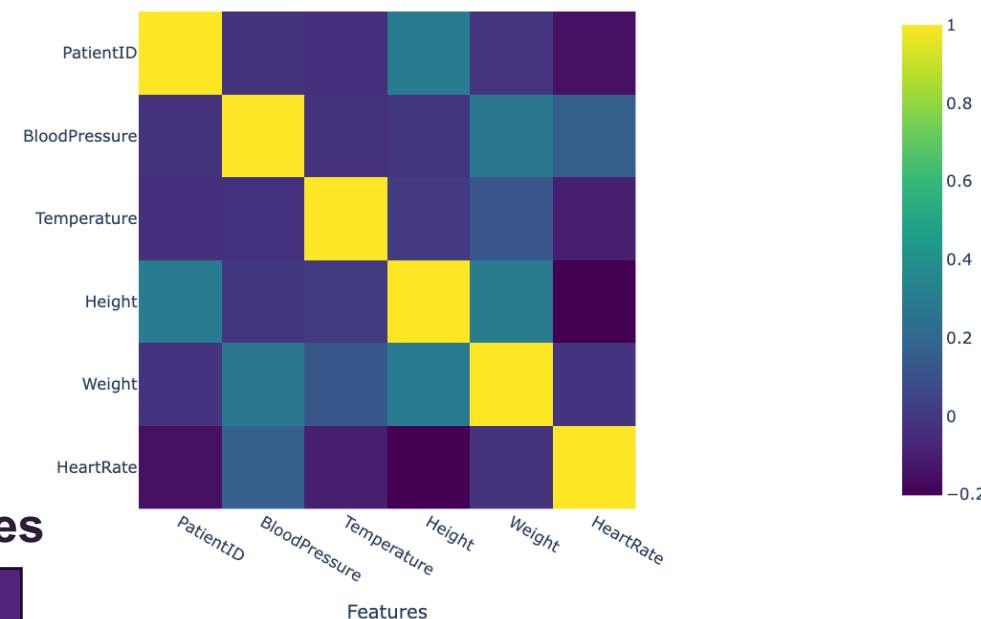
data = {
    'PatientID': range(1, n_patient + 1),
    'BloodPressure': np.random.randint(90, 160, n_patients),
    'Temperature': np.random.uniform(35.0, 37.5, n_patients),
    'Height': np.random.uniform(150, 200, n_patients),
    'Weight': np.random.uniform(50, 100, n_patients),
    'HeartRate': np.random.uniform(60, 100, n_patients)
}

```

brief explanation of the code

details about the code

Correlation Heatmap of Patient Data



features  
patient

# ChatGPT: Writing Python code

**Query:** Not correlation heatmap, but normal heatmap

You  
Not correlation heatmap, but normal heatmap

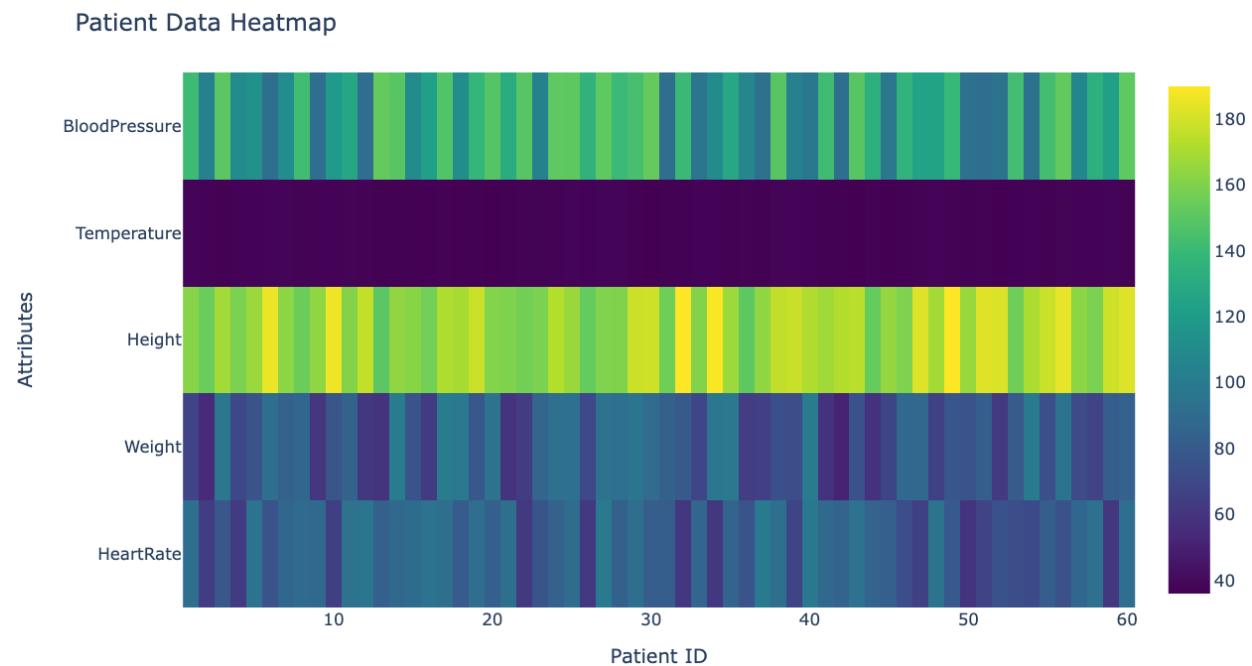
ChatGPT  
I see, you want to create a normal heatmap without correlation values. In that case, you can create a heatmap to visualize the values of various patient attributes directly without computing correlations. Here's the updated code:

```
python
import plotly.express as px
import pandas as pd
import numpy as np

# Generate synthetic patient data
np.random.seed(42)
n_patients = 60

data = {
    'PatientID': range(1, n_patients + 1),
    'BloodPressure': np.random.randint(90, 160, n_patients),
    'Temperature': np.random.uniform(36.0, 37.5, n_patients),
    'Height': np.random.uniform(150, 190, n_patients),
    'Weight': np.random.uniform(50, 100, n_patients),
    'HeartRate': np.random.randint(60, 100, n_patients),
}

df = pd.DataFrame(data)
```



# ChatGPT: Writing Python code

## Asking for Machine Learning examples

Y Can you write Python code to run machine learning on a dataset of 500 patients using the XGBoost classifier?

Certainly! I can provide you with an example of how to use the XGBoost classifier for machine learning on a dataset of 500 patients. Here's an example code snippet that demonstrates the steps involved:

```
python
import pandas as pd
import xgboost as xgb
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Load the dataset (assuming it's in a CSV file)
dataset = pd.read_csv('patient_data.csv')

# Split the dataset into features and target variable
X = dataset.drop('target', axis=1)
y = dataset['target']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create an XGBoost classifier
model = xgb.XGBClassifier()

# Train the model
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Calculate accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
```

Y Write Python code for generating patient data and running machine learning with XGBoost on a dataset of 500 patients.

Certainly! Here's the Python code for generating patient data and running machine learning with XGBoost on a dataset of 500 patients:

```
python
import pandas as pd
import numpy as np
import xgboost as xgb
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Set the number of patients and attributes
num_patients = 500
num_attributes = 5

# Generate random patient data
np.random.seed(0)
patient_data = np.random.randint(18, 80, size=(num_patients, num_attributes))

# Create column names for attributes
attribute_names = ['Age', 'Weight', 'Height', 'Temperature', 'Blood Pressure']

# Create a DataFrame from the patient data
df = pd.DataFrame(patient_data, columns=attribute_names)
df['Target'] = np.random.randint(0, 2, size=num_patients) # Adding a random target column

# Split the dataset into features and target variable
X = df.drop(['Target'], axis=1)
y = df['Target']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create an XGBoost classifier
model = xgb.XGBClassifier()

# Train the model
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Calculate accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
```

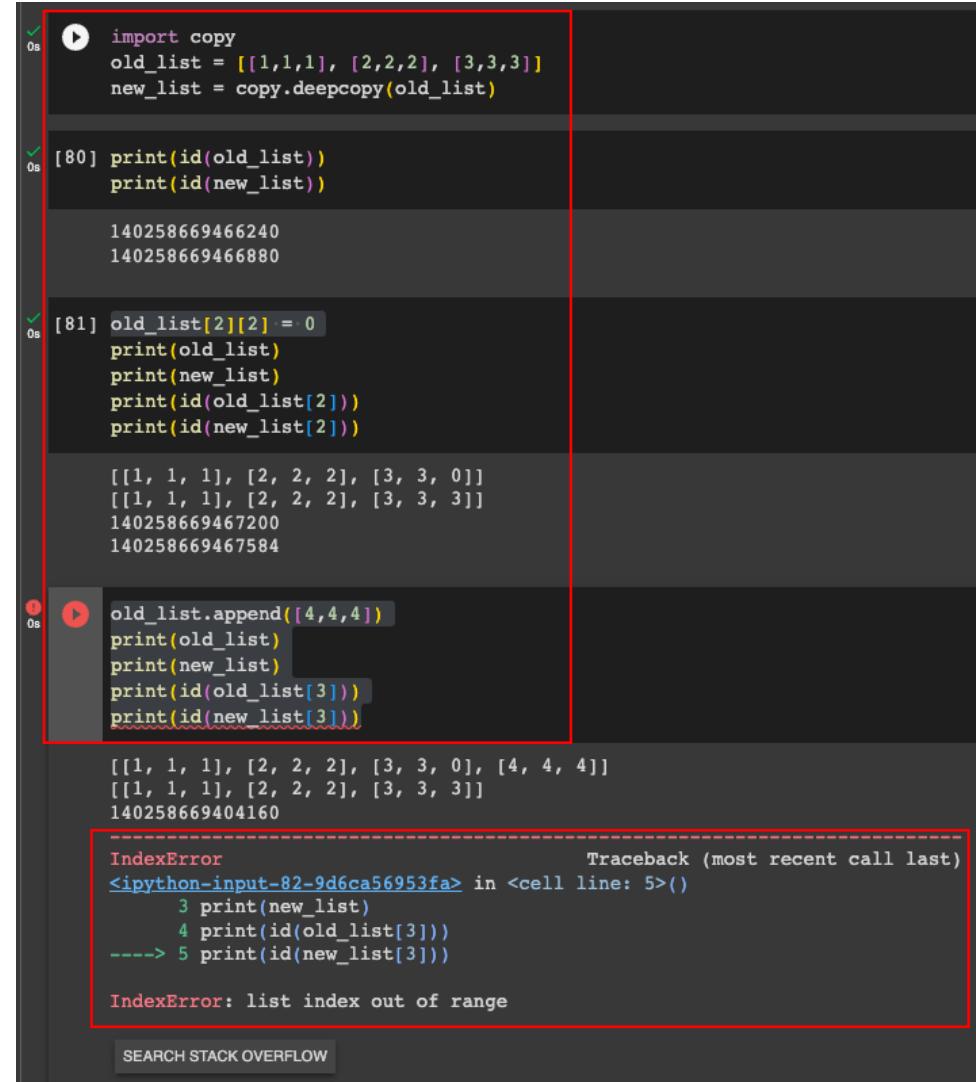
# ChatGPT : Debugging

Y

Can you debug the following code?

```
import copy
old_list = [[1,1,1], [2,2,2], [3,3,3]]
new_list = copy.deepcopy(old_list)

old_list[2][2] = 0
old_list.append([4,4,4])
print(old_list)
print(new_list)
print(id(old_list[3]))
print(id(new_list[3]))
```

```
0s ⏪ import copy
old_list = [[1,1,1], [2,2,2], [3,3,3]]
new_list = copy.deepcopy(old_list)

0s [80] print(id(old_list))
print(id(new_list))

140258669466240
140258669466880

0s [81] old_list[2][2] = 0
print(old_list)
print(new_list)
print(id(old_list[2]))
print(id(new_list[2]))

[[1, 1, 1], [2, 2, 2], [3, 3, 0]]
[[1, 1, 1], [2, 2, 2], [3, 3, 3]]
140258669467200
140258669467584

0s ⏴ old_list.append([4,4,4])
print(old_list)
print(new_list)
print(id(old_list[3]))
print(id(new_list[3]))

[[1, 1, 1], [2, 2, 2], [3, 3, 0], [4, 4, 4]]
[[1, 1, 1], [2, 2, 2], [3, 3, 3]]
140258669404160

IndexError Traceback (most recent call last)
<ipython-input-82-9d6ca56953fa> in <cell line: 5>()
      3 print(new_list)
      4 print(id(old_list[3]))
----> 5 print(id(new_list[3]))

IndexError: list index out of range
```

SEARCH STACK OVERFLOW

# ChatGPT : Debugging

Q: Can you debug the following code? (1<sup>st</sup> try)

The code you provided seems to be correct, and there is no apparent error in it. However, let's go through it and understand the behavior and output:

```
python
import copy

old_list = [[1, 1, 1], [2, 2, 2], [3, 3, 3]]
new_list = copy.deepcopy(old_list)

old_list[2][2] = 0
old_list.append([4, 4, 4])
print(old_list)
print(new_list)
print(id(old_list[3]))
print(id(new_list[3]))
```

This code demonstrates the usage of `copy.deepcopy()` to create a deep copy of `old\_list` into `new\_list`. Here's the breakdown of what the code does and the expected output:

1. `old\_list` is a list of lists containing three sublists.
2. `new\_list` is created using `copy.deepcopy(old\_list)`, which creates a separate copy of `old\_list` with independent nested lists.
3. The value at `old\_list[2][2]` is modified to 0.
4. A new list `[4, 4, 4]` is appended to `old\_list`.
5. `old\_list` and `new\_list` are printed.
6. The `id()` function is used to print the memory addresses of the fourth element in both lists.

The expected output of the code should be as follows:

```
lua
[[1, 1, 1], [2, 2, 2], [3, 3, 0], [4, 4, 4]]
[[1, 1, 1], [2, 2, 2], [3, 3, 3]]
<id_of_old_list_element>
<id_of_new_list_element>
```

Note: The `<id\_of\_old\_list\_element>` and `<id\_of\_new\_list\_element>` values will be specific memory addresses and will differ from system to system.

If you are experiencing any specific issues or unexpected behavior, please let me know, and I'll be happy to assist you further.

# ChatGPT : Debugging

Q: Can you debug the following code? (2<sup>nd</sup> try)

The code you provided appears to be correct and does not require any debugging. It performs a deep copy of the `old\_list` using `copy.deepcopy()`, modifies the `old\_list` by changing the value at `old\_list[2][2]` to 0, and appends a new sublist `[4, 4, 4]` to `old\_list`. After that, it prints both `old\_list` and `new\_list`, followed by printing the IDs of the fourth elements (`old\_list[3]` and `new\_list[3]`) using the `id()` function.

The output of the code should be as follows:

```
lua
[[1, 1, 1], [2, 2, 2], [3, 3, 0], [4, 4, 4]]
[[1, 1, 1], [2, 2, 2], [3, 3, 3]]
[4, 4, 4]
[4, 4, 4]
```

The IDs of `old\_list[3]` and `new\_list[3]` will be the same because `copy.deepcopy()` creates a completely independent copy of the original list.

If you encounter any specific error or unexpected behavior, please provide more details, and I'll be happy to assist you further.

# ChatGPT : Debugging

Y Are you sure?

 Apologies for the confusion. I made a mistake in my previous response. The code you provided does have an error.

Here's the corrected version of the code:

```
python
import copy

old_list = [[1, 1, 1], [2, 2, 2], [3, 3, 3]]
new_list = copy.deepcopy(old_list)

old_list[2][2] = 0
old_list.append([4, 4, 4])

print(old_list)
print(new_list)
print(id(old_list[3]))
print(id(new_list[2]))
```

In the last line of the code, the index for `new\_list` should be `2` instead of `3`.

The corrected code will give the following output:

```
lua
[[1, 1, 1], [2, 2, 2], [3, 3, 0], [4, 4, 4]]
[[1, 1, 1], [2, 2, 2], [3, 3, 3]]
140351722261888
140351722262016
```

Please let me know if you have any further questions or concerns.

# Thank you

Yoochan Myung

Postdoctoral Research Fellow

y.myung@uq.edu.au



CRICOS 00025B • TEQSA PRV12080