COMP7630 – Web Intelligence and its Applications

Revision of Python

Outline

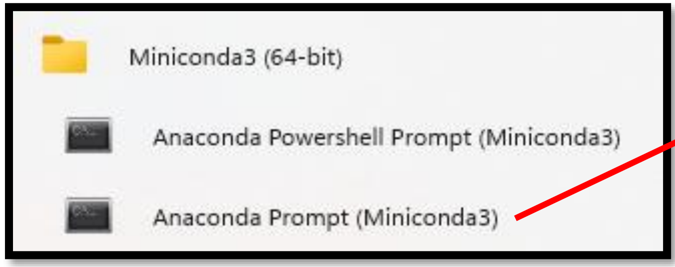
- A full Python environment
- Python 101

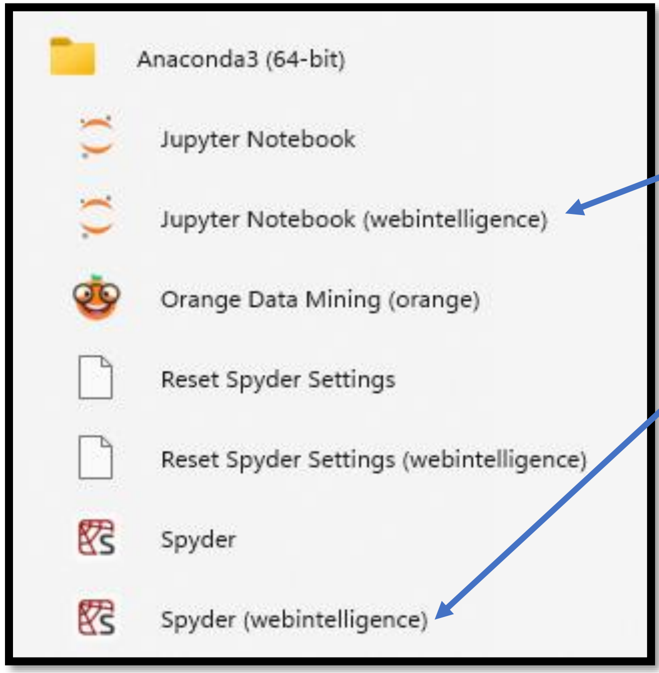
MiniConda

- MiniConda is one of the most popular Python distribution platform
- It is a lightweigth version of Anaconda
- Documentation and download links are available at

  https://docs.conda.io/projects/miniconda/en/latest/

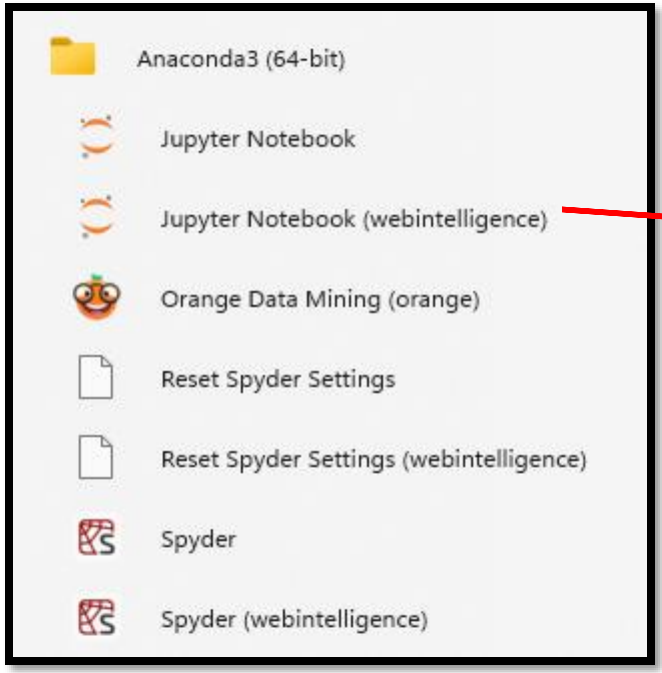
Once installed …


Miniconda3 (64-bit)
Anaconda Powershell Prompt (Miniconda3)
Anaconda Prompt (Miniconda3)


Anaconda3 (64-bit)
Jupyter Notebook
Jupyter Notebook (webintelligence)
Orange Data Mining (orange)
Reset Spyder Settings
Reset Spyder Settings (webintelligence)
Spyder
Spyder (webintelligence)

- Launch prompt inside the `base` environment.
- Environments are sand-boxed among them and ease the management of Python packages.
- I suggest to create the `webintelligence` environment:
  - `conda create -n webintelligence`
  - `conda activate webintelligence`
  - `conda install python==3.11.5`
- It may be useful to install also Jupyter and Spyder:
  - `conda install jupyter spyder`
- Please install `pip`, the Python packages' installer:
  - `conda install pip`
  - `conda config --set pip_interop_enabled True`
- Please install `ipython`, the Improved Python interactive shell
  - `pip install ipython`
- Please install some Python packages we will use later on
  - `pip install numpy matplotlib seaborn pandas scipy scikit-learn spacy nltk nevergrad mlxtend beautifulsoup4`
- If required, we will install additional Python packages by using
  - `pip install <package_name>`

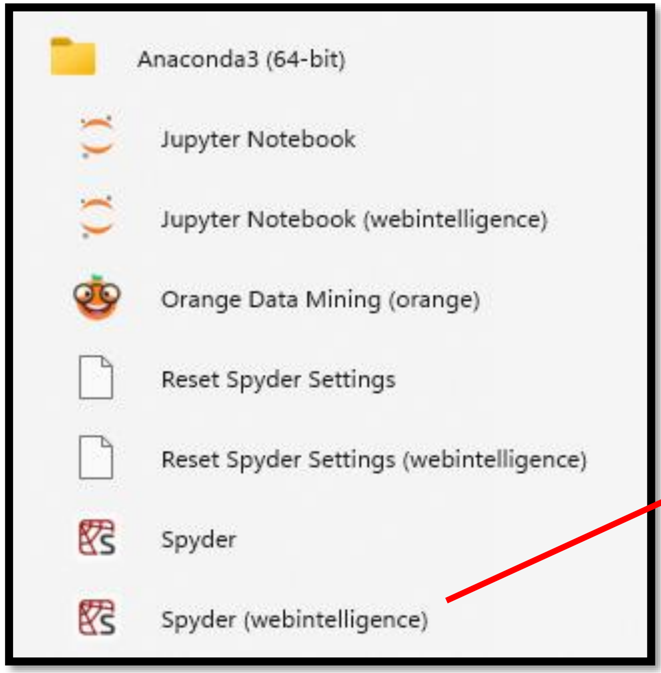
Once installed …


Anaconda3 (64-bit)
Jupyter Notebook
Jupyter Notebook (webintelligence)
Orange Data Mining (orange)
Reset Spyder Settings
Reset Spyder Settings (webintelligence)
Spyder
Spyder (webintelligence)

It launches a local web application, then it opens your browser pointing at that web application.

It is useful to launch and open Jupyter Notebook

Once installed …


Anaconda3 (64-bit)
Jupyter Notebook
Jupyter Notebook (webintelligence)
Orange Data Mining (orange)
Reset Spyder Settings
Reset Spyder Settings (webintelligence)
Spyder
Spyder (webintelligence)

This is a well known editor for Python. You are free to use it, but sometime I may prefer Notepad++ & a shell for running "ipython" or directly the "python" interpreter.

Notepad++ can be downloaded from https://notepad-plus-plus.org/downloads/

Hello World

- Create a file named `helloworld.py`
- Write the following Python statement inside `helloworld.py`
  - `print('Hello World')`

- Open the Miniconda shell (activate the "webintelligence" environment if you decided to use it) and use the following commands:
  - `cd <directory_where_you_saved_helloworld.py>`
  - `python helloworld.py`

- If you see the phrase "Hello World", your Miniconda/Python installation is working!
- You may also try to launch `ipython` and write `1+1`, you should see `2`

Outline

- A full Python environment
- Python 101

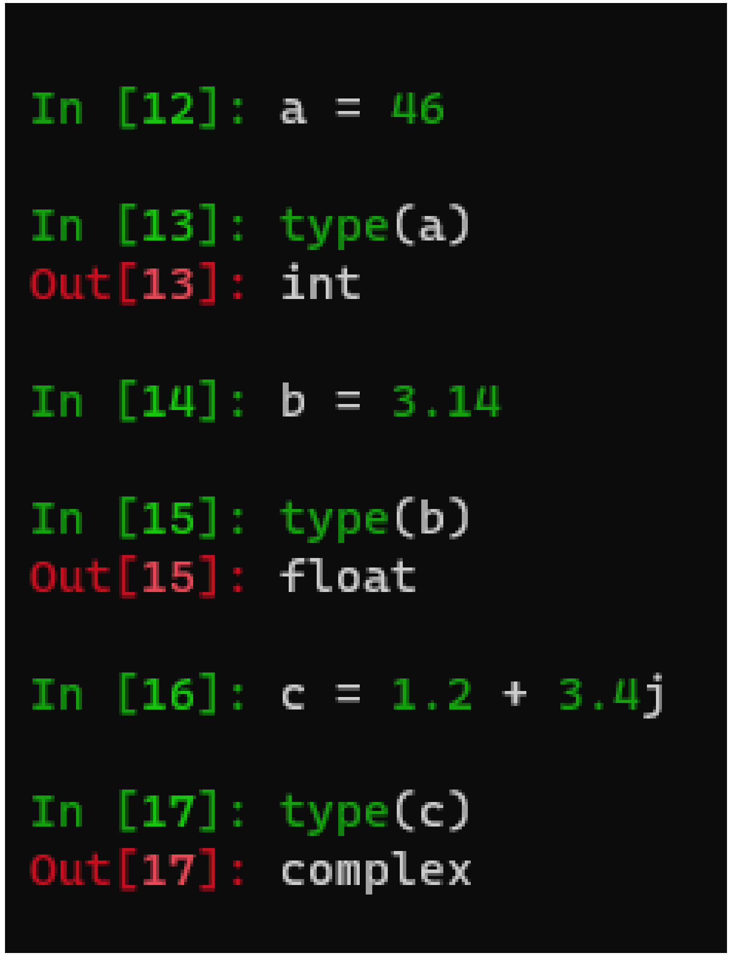
Basic variables and data structure

- Different from other programming languages, Python does not need to specify the data type while declaring the variable.
- It can automatically identify the data type by different declaring method.
- Python has five standard data types:
  - Numbers
  - String
  - List
  - Tuple
  - Dictionary

Numbers in Python

- 3 different numerical types:
  - `int` (signed integers)
  - `float` (floating point real values)
  - `complex` (complex numbers)

- The operator `type` gives you the type of any variable

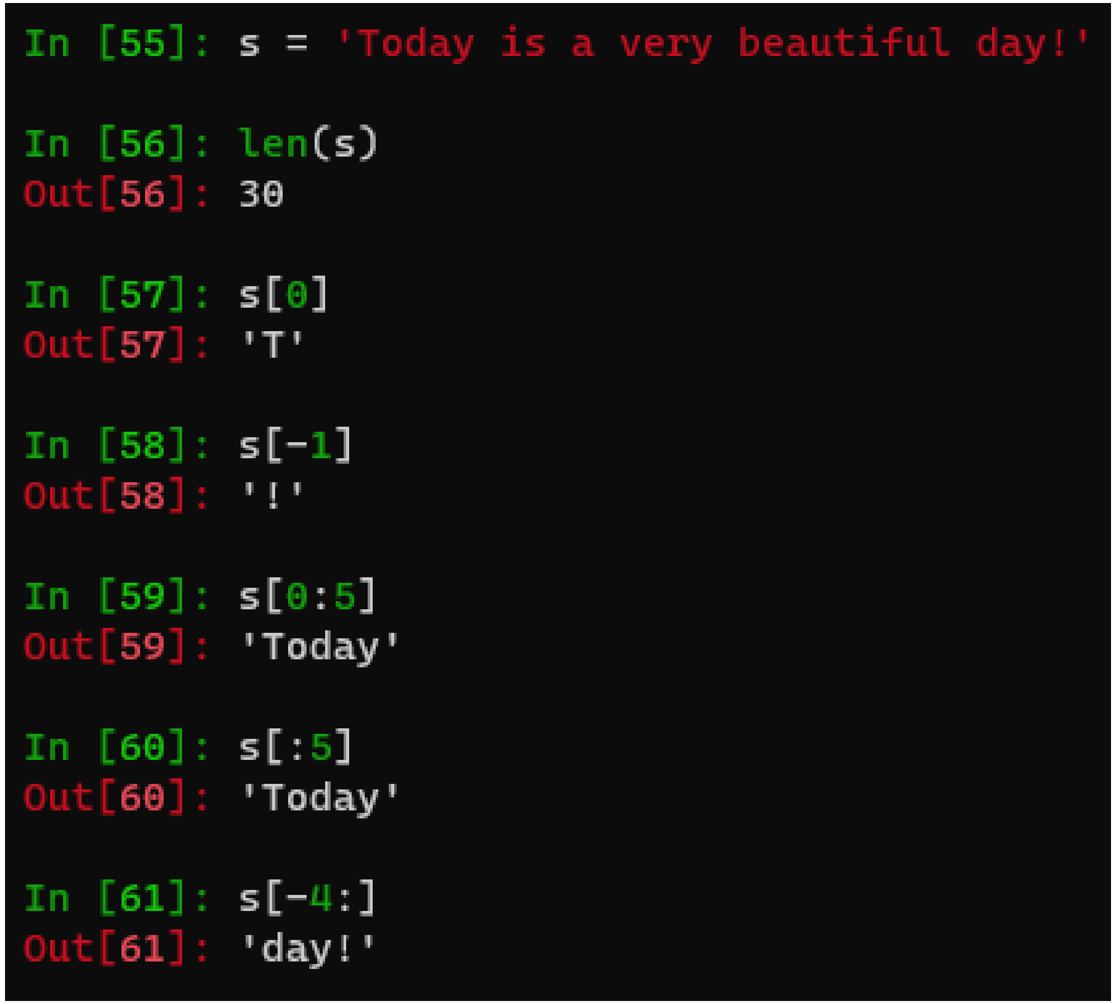


```
In [12]: a = 46

In [13]: type(a)
Out[13]: int

In [14]: b = 3.14

In [15]: type(b)
Out[15]: float

In [16]: c = 1.2 + 3.4j

In [17]: type(c)
Out[17]: complex
```

Strings in Python

- Strings are contiguous sequence of characters inside quotation marks (both single and double quotes are ok)

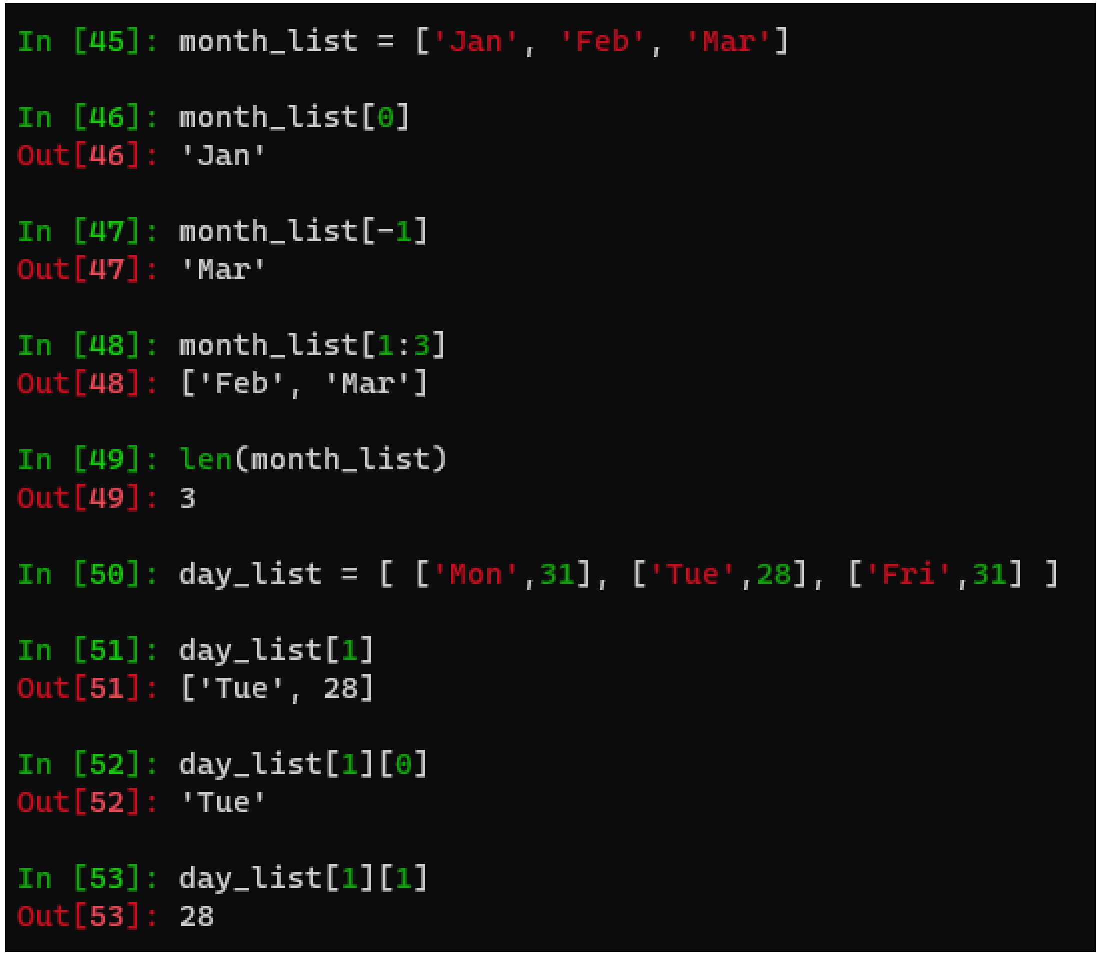
- You can get the subsequence of the strings by using slice operator ([ ] and [:]) with indexes, 0 from the beginning and -1 from the end of string



```
In [55]: s = 'Today is a very beautiful day!'

In [56]: len(s)
Out[56]: 30

In [57]: s[0]
Out[57]: 'T'

In [58]: s[-1]
Out[58]: '!'

In [59]: s[0:5]
Out[59]: 'Today'

In [60]: s[:5]
Out[60]: 'Today'

In [61]: s[-4:]
Out[61]: 'day!'
```

Lists in Python

- Python “list” is a variable that can store multiple items and types of data
- The data is stored sequentially in list “elements”
- Lists can have more than one index – to represent multiple dimensions

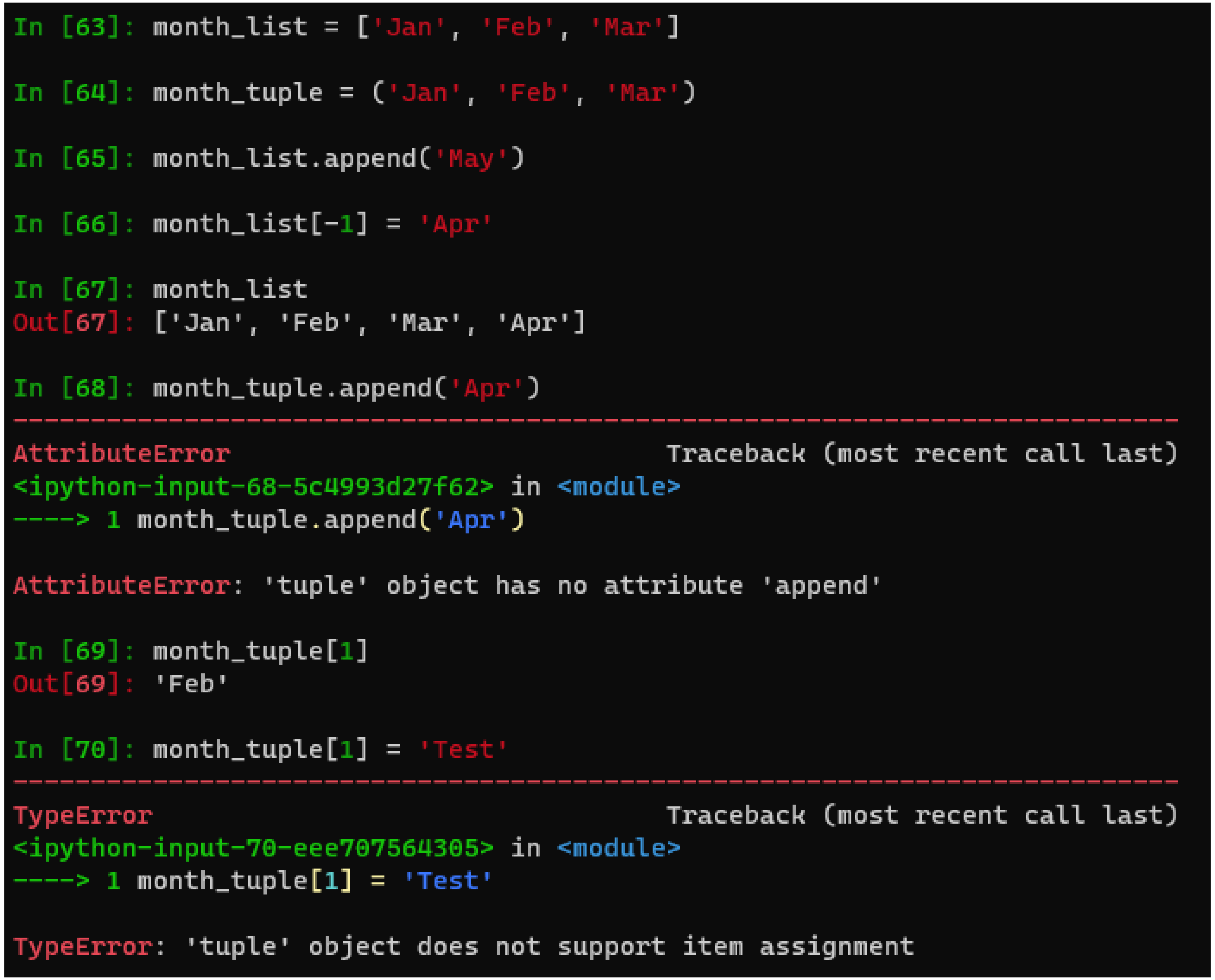


```
In [45]: month_list = ['Jan', 'Feb', 'Mar']

In [46]: month_list[0]
Out[46]: 'Jan'

In [47]: month_list[-1]
Out[47]: 'Mar'

In [48]: month_list[1:3]
Out[48]: ['Feb', 'Mar']

In [49]: len(month_list)
Out[49]: 3

In [50]: day_list = [ ['Mon',31], ['Tue',28], ['Fri',31] ]

In [51]: day_list[1]
Out[51]: ['Tue', 28]

In [52]: day_list[1][0]
Out[52]: 'Tue'

In [53]: day_list[1][1]
Out[53]: 28
```

Tuples in Python

The nature of tuple is similar to list, but the main differences between these two structures are:

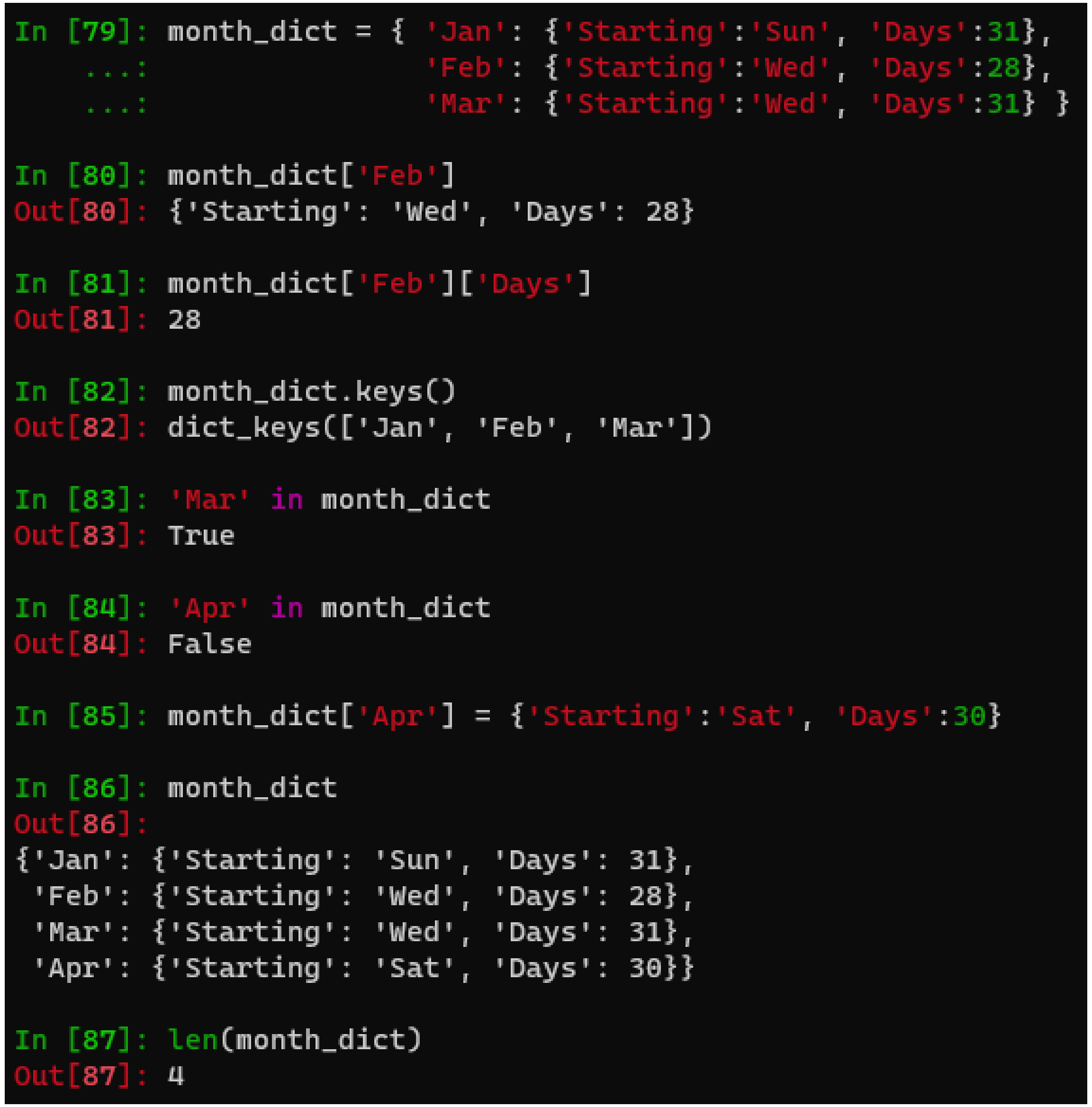
- list allowed modification for both size and elements, tuple cannot be updated
- list is enclosed by brackets [ ], tuple is enclosed by parentheses ( )



```
In [63]: month_list = ['Jan', 'Feb', 'Mar']

In [64]: month_tuple = ('Jan', 'Feb', 'Mar')

In [65]: month_list.append('May')

In [66]: month_list[-1] = 'Apr'

In [67]: month_list
Out[67]: ['Jan', 'Feb', 'Mar', 'Apr']

In [68]: month_tuple.append('Apr')
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-68-5c4993d27f62> in <module>
----> 1 month_tuple.append('Apr')

AttributeError: 'tuple' object has no attribute 'append'

In [69]: month_tuple[1]
Out[69]: 'Feb'

In [70]: month_tuple[1] = 'Test'
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-70-eee707564305> in <module>
----> 1 month_tuple[1] = 'Test'

TypeError: 'tuple' object does not support item assignment
```

Dictionaries in Python

- A Python's dictionary is a data container that can store multiple items of data as a list of `key:value` pairs.
- Unlike regular list container values, which are referenced by their index numbers, values stored in dictionaries are referenced by their associated key.
- The key must be unique within that dictionary and is typically a string name although numbers may be used.

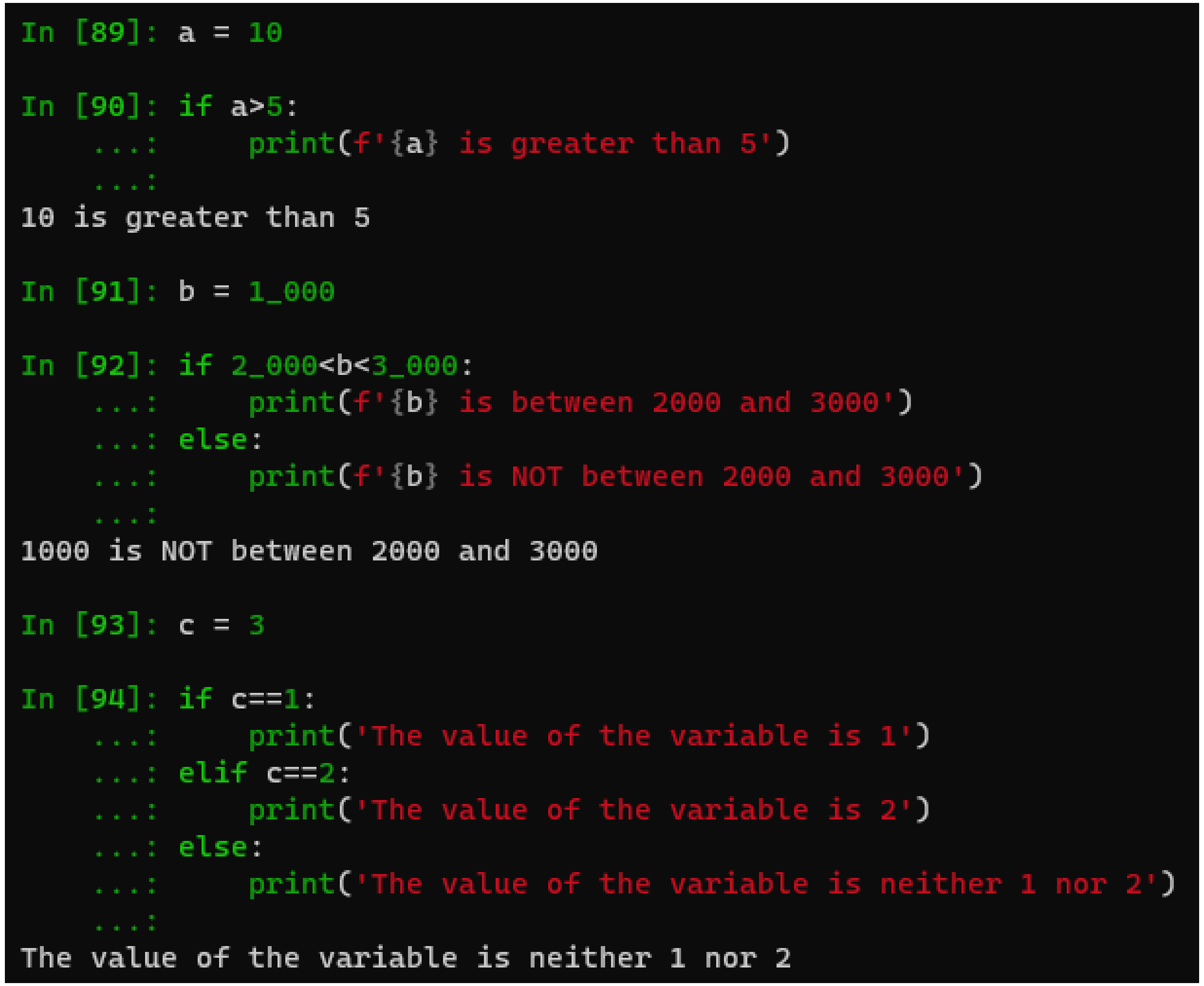


```
In [79]: month_dict = { 'Jan': {'Starting':'Sun', 'Days':31},
    ...:                'Feb': {'Starting':'Wed', 'Days':28},
    ...:                'Mar': {'Starting':'Wed', 'Days':31} }

In [80]: month_dict['Feb']
Out[80]: {'Starting': 'Wed', 'Days': 28}

In [81]: month_dict['Feb']['Days']
Out[81]: 28

In [82]: month_dict.keys()
Out[82]: dict_keys(['Jan', 'Feb', 'Mar'])

In [83]: 'Mar' in month_dict
Out[83]: True

In [84]: 'Apr' in month_dict
Out[84]: False

In [85]: month_dict['Apr'] = {'Starting':'Sat', 'Days':30}

In [86]: month_dict
Out[86]:
{'Jan': {'Starting': 'Sun', 'Days': 31},
 'Feb': {'Starting': 'Wed', 'Days': 28},
 'Mar': {'Starting': 'Wed', 'Days': 31},
 'Apr': {'Starting': 'Sat', 'Days': 30}}

In [87]: len(month_dict)
Out[87]: 4
```

If, else, elif

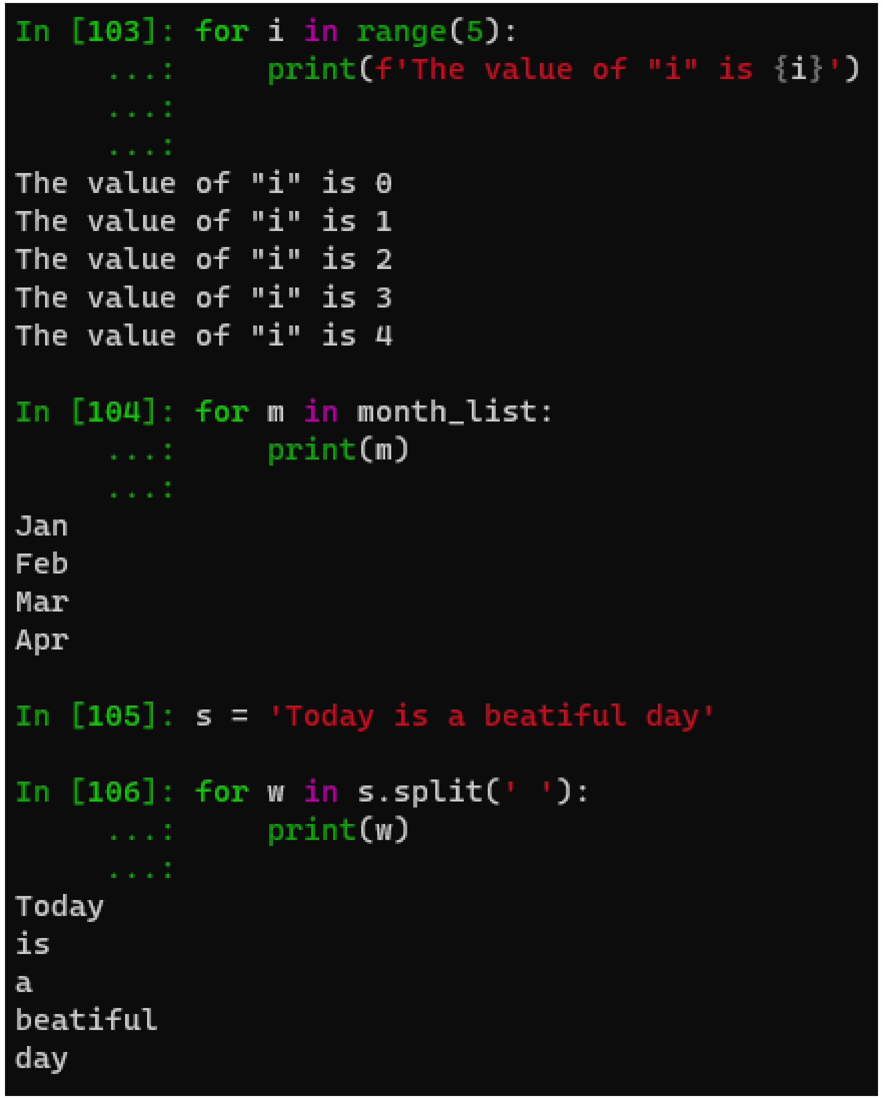
- In Python, braces are not used when defining classes, function, or flow control

- Instead, line indentation is used for denoting blocks of code

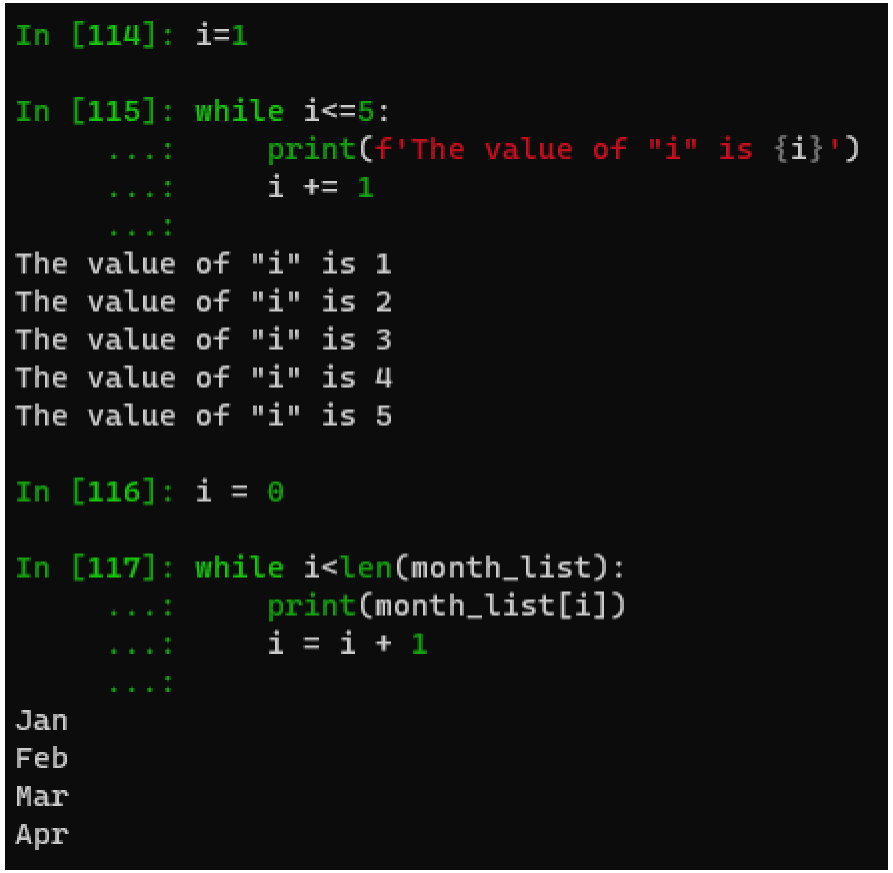


```
In [89]: a = 10

In [90]: if a>5:
    ...:     print(f'{a} is greater than 5')
    ...:
10 is greater than 5

In [91]: b = 1_000

In [92]: if 2_000<b<3_000:
    ...:     print(f'{b} is between 2000 and 3000')
    ...: else:
    ...:     print(f'{b} is NOT between 2000 and 3000')
    ...:
1000 is NOT between 2000 and 3000

In [93]: c = 3

In [94]: if c==1:
    ...:     print('The value of the variable is 1')
    ...: elif c==2:
    ...:     print('The value of the variable is 2')
    ...: else:
    ...:     print('The value of the variable is neither 1 nor 2')
    ...:
The value of the variable is neither 1 nor 2
```

For and while

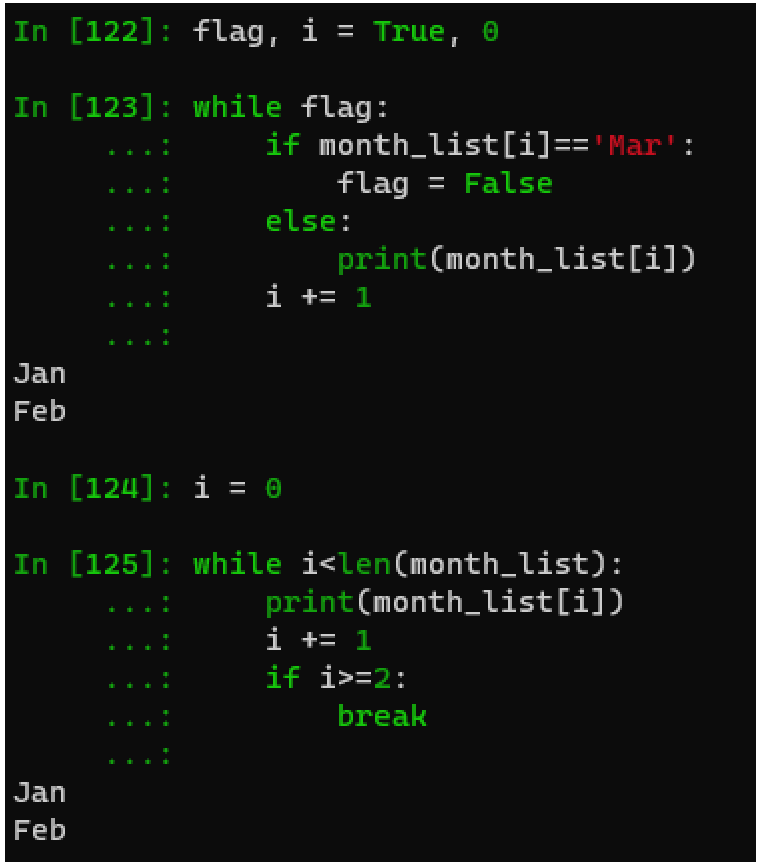


```
In [103]: for i in range(5):
     ...:     print(f'The value of "i" is {i}')
     ...:
     ...:
The value of "i" is 0
The value of "i" is 1
The value of "i" is 2
The value of "i" is 3
The value of "i" is 4

In [104]: for m in month_list:
     ...:     print(m)
     ...:
Jan
Feb
Mar
Apr

In [105]: s = 'Today is a beatiful day'

In [106]: for w in s.split(' '):
     ...:     print(w)
     ...:
Today
is
a
beatiful
day
```

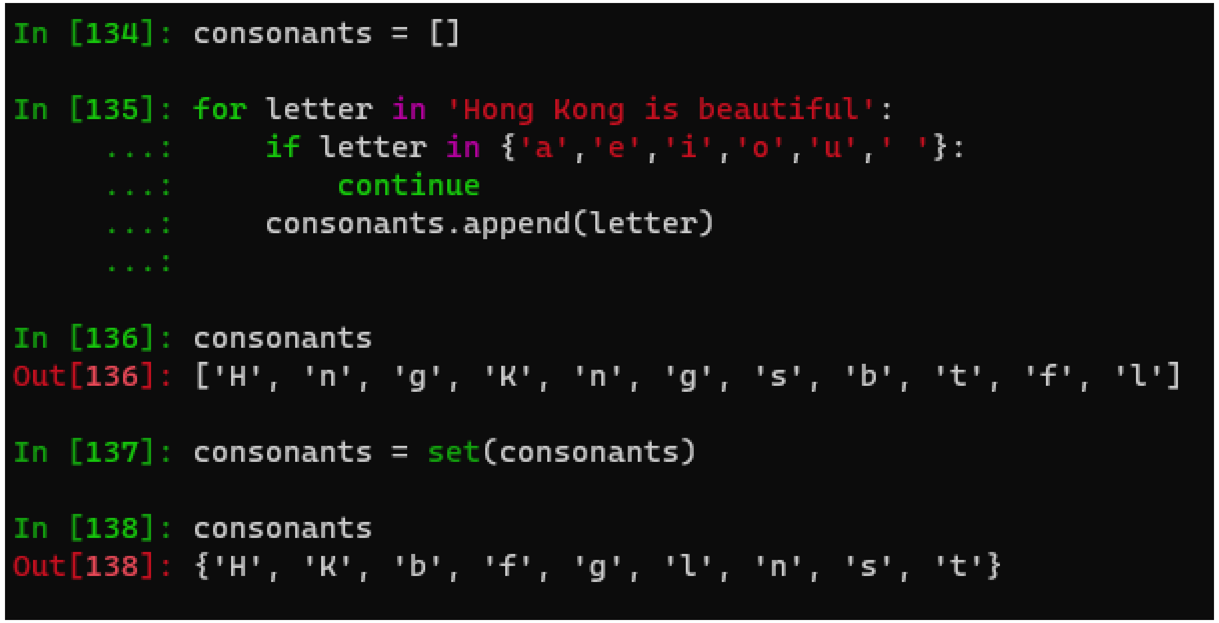


```
In [114]: i=1

In [115]: while i<=5:
     ...:     print(f'The value of "i" is {i}')
     ...:     i += 1
     ...:
The value of "i" is 1
The value of "i" is 2
The value of "i" is 3
The value of "i" is 4
The value of "i" is 5

In [116]: i = 0

In [117]: while i<len(month_list):
     ...:     print(month_list[i])
     ...:     i = i + 1
     ...:
Jan
Feb
Mar
Apr
```

Break and continue



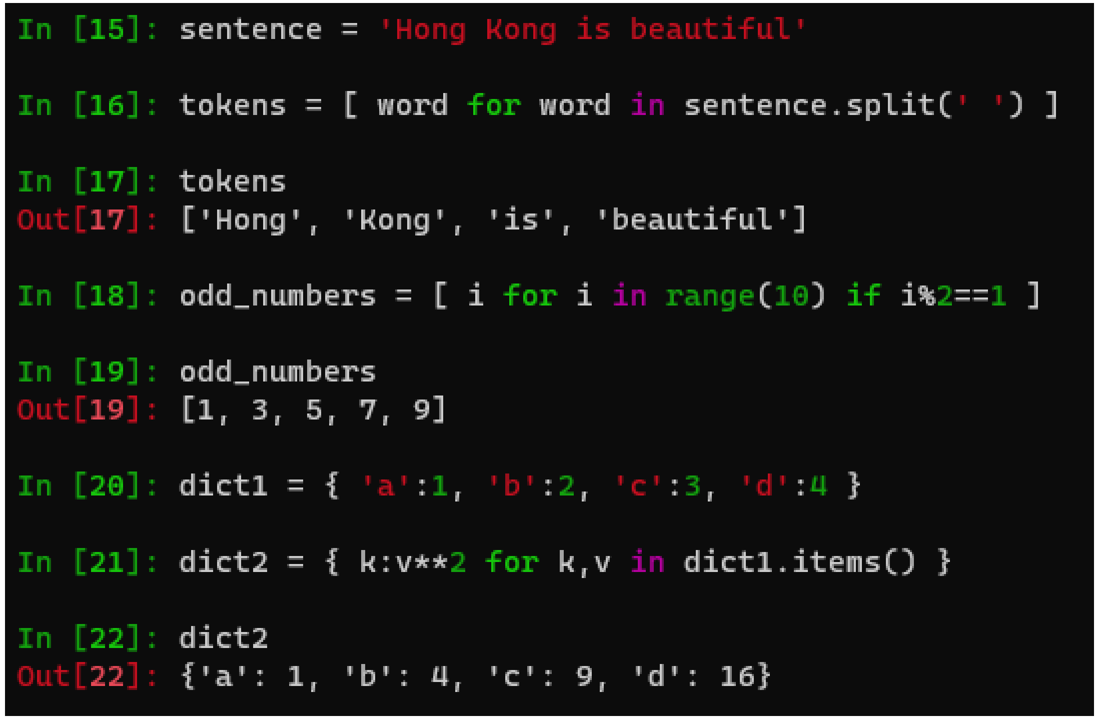
```
In [122]: flag, i = True, 0

In [123]: while flag:
     ...:     if month_list[i]=='Mar':
     ...:         flag = False
     ...:     else:
     ...:         print(month_list[i])
     ...:     i += 1
     ...:
Jan
Feb

In [124]: i = 0

In [125]: while i<len(month_list):
     ...:     print(month_list[i])
     ...:     i += 1
     ...:     if i>=2:
     ...:         break
     ...:
Jan
Feb
```



```
In [134]: consonants = []

In [135]: for letter in 'Hong Kong is beautiful':
     ...:     if letter in {'a','e','i','o','u',' '}:
     ...:         continue
     ...:     consonants.append(letter)
     ...:

In [136]: consonants
Out[136]: ['H', 'n', 'g', 'K', 'n', 'g', 's', 'b', 't', 'f', 'l']

In [137]: consonants = set(consonants)

In [138]: consonants
Out[138]: {'H', 'K', 'b', 'f', 'g', 'l', 'n', 's', 't'}
```

Comprehensive expressions

- Make possible to build lists/tuples/dictionaries by using an almost-mathematical notation



```
In [15]: sentence = 'Hong Kong is beautiful'

In [16]: tokens = [ word for word in sentence.split(' ') ]

In [17]: tokens
Out[17]: ['Hong', 'Kong', 'is', 'beautiful']

In [18]: odd_numbers = [ i for i in range(10) if i%2==1 ]

In [19]: odd_numbers
Out[19]: [1, 3, 5, 7, 9]

In [20]: dict1 = { 'a':1, 'b':2, 'c':3, 'd':4 }

In [21]: dict2 = { k:v**2 for k,v in dict1.items() }

In [22]: dict2
Out[22]: {'a': 1, 'b': 4, 'c': 9, 'd': 16}
```

What else?

- Defining functions
- Defining classes
- Using classes
- Importing modules
- …

Resources for further study:

- https://www.learnpython.org/
- https://www.w3schools.com/python/