BIT2008A/ITEC2000A: Multimedia Data Management
Fall 2022

Lab 5

1. Python Tricks

This tutorial aims to get you familiar with 11 Python features that make it easier to develop your programs. There are a couple of ways to run python. First, we can write our programs in (.py) files using code editors like Notepad or Visual Studio Code, and run them via the Python interpreter. If you have installed Anaconda, you can open Anaconda Navigator and use your favorite prompt. To run your
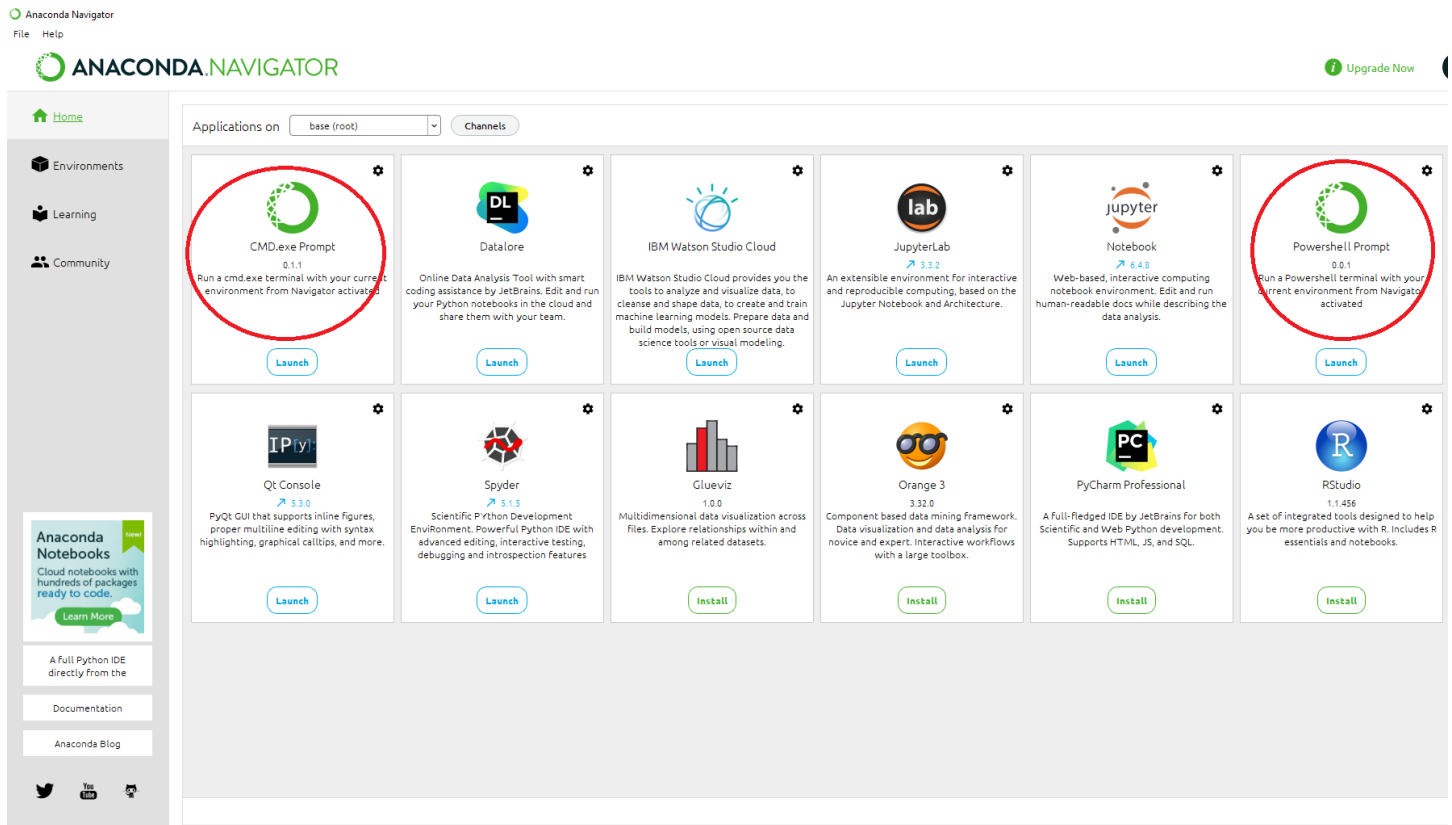

Figure 1. Python prompts available in PowerShell and CMD.

program, you need to locate the file and then write ***python [path to .py file]*** in the terminal. Secondly, There are interactive Python consoles to execute each line that you write in real-time. You can find the Qt Console shown in Fig. 1 or type ***ipython*** in your terminal if you have activated Anaconda there (Virtual Machine option). Finally, you can create a Jupyter Notebook to run your code, write instructions and comments, and present the results. Jupyter Notebooks enables you to create a mixture of text cells and code cells to report your Python projects more easily.

You can use google colab to create and run Jupyter Notebooks online. Jupyter Notebooks can be converted to other formats such as (.pdf) or (.html) easily. The rest of this instruction is directly copied from the Jupyter Notebook that you are provided with.

Lab5

October 11, 2022

1 11 Python Tricks that makes coding easier.

The text cells support Markdown as well as plain text.

1.1 01 Enumerate

If you really want to index a list on which you are executing a for loop, you have enumerate method to make it easier.

```python
values = ['a', 'b', 'c']


# typical python for loop
for v in values:
    print(v)
```

```python
# this is not the best way to index the list inside a for loop

i = 0
for _ in values:
    print(i, values[i])
    i += 1

# Nor this
for i in range(len(values)):
    print(i, values[i])
```

```python
# instead you can use enumerate()
for i, v in enumerate(values):
    print(i, v)
```

```python
# Now write a function using enumerate that takes two arrays with the same
 ↪length and creates
# a paired array as a result. [1, 2, 3], ['a', 'b', 'c'] -> [[1,'a'], [2,'b'],
 ↪[3,'c']]

def zipper(arr1, arr2):
    # send error if the arrays don't have the same length
    assert len(arr1) == len(arr2)
```

```
'''
    # create an empty array for the result
    res = []
    for ...:
        # use append to add elements to a list
        res.append(...)
    return res
'''
```

1.2 02 Else block

As mentioned in the lecture, we can use `else` blocks along with many flow control keywords in Python.

```
[ ]: # this method checks if a number is prime
     def prime(num):
         if num > 1:
             for i in range(2,num):
                 if (num % i) == 0:
                     return False
             else:
                 return True

     # now write a method that checks if there is any prime numbers in a list
     def list_prime(arr):
         '''
         for ...:
             ...
             if prime(a):
                 ...
         '''
```

1.3 03 List comprehensions

```
[ ]: # we can write more compact code using Python list comprehension
     # let's assume that we want to save the result of an elementwise operation on a␣
      ↪list

     x = [1, 2, 3, 4, 5]
     print(x)

     # now we want to store y = x^2 - 3
     y = [n**2 - 3 for n in x]
     print(y)
```

```
[ ]:  # now define a method that returns the cartesian product of two lists using␣
      ↪list comprehension
      def cart_prod(arr1, arr2):
          '''
          return [[...] for ... in arr1 for ... in arr2]
          '''

      print(cart_prod([1,2,3], ['a', 'b', 'c', 'd']))
```

1.4 04 Formatted Strings

There are multiple ways to inject your variables inside a python string but here is the most straight forward one. You should put an f character before starting your string, and put your variables inside {}.

```
[ ]:  fname, lname = 'Mohammad', 'Heydari'
      print(f'My first name is:{fname}, and my last name is:{lname}.')
```

```
[ ]:  # now define a function that takes a dictionary schedule and prints it properly.
      # here is a sample schedule
      schedule = {"laundry": 50, "study": 120, "dinner": 30}

      def print_schedule(dict1):
          '''
          for ... in dict1.items():
              print(f'...')
          '''
```

1.5 05 Itertools

Itertools is a library available in any Python installation providing you with functions for efficient looping. documentation

```
[ ]:  import itertools

      for a, b in itertools.product(['a', 'b', 'c'], range(1,5)):
          print(f'{{{a}, {b}}}')
```

```
[ ]:  # now write a method to extract the all of the possible DNA strings with length␣
      ↪n
      # product method has a 'repeat' argument that controls the length of the output
      # dna chains are shown using A,T,C,or G characters.
      # for instance AACTCGAG is a dna string with length 8.

      def dna_counter(n):
          ''' use this line in your implementation:
          itertools.product('ATCG', repeat=n)
```

```
    '''
```

1.6   06 one-liner if else

You can use mini conditional statement to initiate your variables or execute other functions.

```python
age = 45
if age>40:
    print('old')
else:
    print('young')
```

```python
age = 45
print('old' if age>40 else 'young')
```

```python
import random
# random.random() gives you a random value between [0, 1).
# use this to initiate your variable x with 0 if the random value is smaller␣
 ↪than 0.2
# or 1 otherwise.
'''
x = ... if ... else ...
print(x)
'''
```

1.7   07 Pathlib

Pathlibs enables us to find the current directory when executing our code. It does not depend on the OS or the libraries you have installed on your python distribution.

```python
import pathlib
print(f'currnet working directory: {pathlib.Path.cwd()}')
```

```python
# using pathlib you can find out if a path is pointing to a director or a file.
# write a code using pathlib.Path.is_dir to verify that
# your current working directory is actually a folder
```

1.8   08 Decorators (Wrapper Functions)

It is one of the most powerful tools of Python programming specifically in Django. Most of the time we just use predefined decorators but it is beneficial to know how to implement one. Decorators are functions that take another function as an input, consider it as a black box and add more functionality to it.

```python
# we want to write a decorator that makes its following function to run twice
def do_twice(func):
    def wrapper_func():
```

```
        func()
        func()
    return wrapper_func

# now this is the way to use a decorator:
@do_twice
def pr_name():
    print('Hey, Mohammad!')

# interpret the code above as this:
# pr_name = do_twice(pr_name)
# now we only have to call the function being wrapped once:
pr_name()
```

```
[ ]: # now let's write a decorator to report excution time
     import time

     # time.time() returns the current system time and you have to substract its
     # value in the beginning of a process from its value in the end of a process

     def time_takes(func):
         pass
     # *args and **kwargs passes every arguments from the original
     # function to the wrapped one
     '''
         def wrap(*args, **kwargs):
             ...
             result = func(*args, **kwargs)
             ...
             return result
         return wrap
     '''

     @time_takes
     def lazy():
         for _ in range(3):
             time.sleep(1)

     lazy()
```

1.9   09 Map

map takes a function and a list as its arguments. Then executes the function on each list element.

```
[ ]: # let's write a map to extract the length of each elements in a list of strings
     array = ['hello', 'world', 'bit2008']
     lengths = map(len, array)
```

```
print(list(lengths))
```

```
[ ]: # let's write another map to calculate the elementwise sum of two arrays
     arr1 = [1, 3, 12, -9, 4, -5, 8]
     arr2 = [3, -4, 5, -6, 7, -9, 8]

     '''
     def sum(...):
         return ...

     y = map(...)
     print(list(y))
     '''
```

1.10 10 Filter

similar to `map`, filter takes a function to pass some of the elements of its other argument which is a list and remove the rest.

```
[ ]: # lets write a filter map that removes frequencies below 100 Hz
     freqs = [10, 120, 125, 230, 80, 150, 70]
     passed = filter(lambda x: True if x>100 else False, freqs)
     print(list(passed))
```

1.11 11 Zip

Zip matches the corresponding elements of multiple lists together.

```
[ ]: id = (1,2,3,4,5,6)
     name = ('Alex', 'Joe', 'Mika')
     major = ('Music', 'Comp Sci', 'IT')
     res = zip(id, name, major)
     print(list(res))
```