Lecture -4

Programing in Python

Instructor : AALWAHAB DHULFIQAR

Advisor : Dr. Tejfel Mate

What you will learn:

- Sequence types and mutability
- Tuples
- Dictionaries
- Exceptions
- Data Processing
- Modules, Packages and PIP

Sequence types and mutability

sequence type is a type of data in Python which is able to store more than one value (or less than one, as a sequence may be empty), and these values can be sequentially (hence the name) browsed, element by element.

mutability - is a property of any of Python's data that describes its readiness to be freely changed during program execution. There are two kinds of Python data: **mutable** and **immutable**.

Tuple

tuples prefer to use parenthesis.
Tuples are sequence.
Tuples are immutable .

```
tuple_1 = (1, 2, 4, 8)
tuple_2 = 1., .5, .25, .125
empty_tuple = ()
one_element_tuple_1 = (1, )
one_element_tuple_2 = 1.,
```

```
my_tuple = (1, 10, 100, 1000)

my_tuple.append(10000)
del my_tuple[0]
my_tuple[1] = -10
```

```
my_tuple = (1, 10, 100, 1000)
print(my_tuple[0])
print(my_tuple[-1])
print(my_tuple[1:])
print(my_tuple[:-2])
for elem in my_tuple:
    print(elem)
```

```
my_tuple = (1, 10, 100, 1000)

print(my_tuple[0])
print(my_tuple[-1])
print(my_tuple[1:])
print(my_tuple[:-2])

for elem in my_tuple:
    print(elem)
```

Dictionary

The dictionary is another Python data structure. It's not a sequence type (but can be easily adapted to sequence processing) and it is mutable.

This means that a dictionary is a set of key-value pairs. Note:

1. Each key must be unique - it's not possible to have more than one key of the same value;
2. A key may be any immutable type of object: it can be a number (integer or float), or even a string, but not a list;
3. A dictionary is not a list - a list contains a set of numbered values, while a dictionary holds pairs of values;
4. The len() function works for dictionaries, too - it returns the numbers of key-value elements in the dictionary;
5. A dictionary is a one-way tool - if you have an English-French dictionary, you can look for French equivalents of English terms, but not vice versa.

```
dictionary = {"cat": "chat", "dog": "chien", "horse": "cheval"}
phone_numbers = {'boss': 5551234567, 'Suzy':
22657854310}
empty_dictionary = {}

print(dictionary)
print(phone_numbers)
print(empty_dictionary)
```

Output

```
{'dog': 'chien', 'horse': 'cheval', 'cat': 'chat'}
{'Suzy': 5557654321, 'boss': 5551234567}
{}
```

How to use a dictionary

```
dictionary = {"cat": "chat", "dog": "chien", "horse": "cheval"}
words = ['cat', 'lion', 'horse']

for word in words:
    if word in dictionary:
        print(word, "->", dictionary[word])
    else:
        print(word, "is not in dictionary")
```

the `keys()`

```
dictionary = {"cat": "chat", "dog": "chien", "horse": "cheval"}

for key in dictionary.keys():
    print(key, "->", dictionary[key])
```

The `items()` and `values()` methods

```
dictionary = {"cat": "chat", "dog": "chien", "horse": "cheval"}

for english, french in dictionary.items():
    print(english, "->", french)
```

modifying and adding values

```
dictionary = {"cat": "chat", "dog": "chien", "horse": "cheval"}

dictionary['cat'] = 'minou'  # Modify
dictionary['swan'] = 'cygne' # Add
print(dictionary)
```

Removing a key

```
dictionary = {"cat": "chat", "dog": "chien", "horse": "cheval"}

del dictionary['dog']
print(dictionary)
```

Tuples and dictionaries can work together

```
school_class = {}
while True:
    name = input("Enter the student's name: ")
    if name == '':
        break

    score = int(input("Enter the student's score (0-10): "))
    if score not in range(0, 11):
                    break

    if name in school_class:
        school_class[name] += (score,)
    else:
        school_class[name] = (score,)

for name in sorted(school_class.keys()):
    adding = 0
    counter = 0
    for score in school_class[name]:
        adding += score
        counter += 1
    print(name, ":", adding / counter)
```

Errors – the developer's daily bread

Errors in data vs. errors in code

Dealing with programming errors has (at least) two sides.

when you get into trouble because your – apparently correct – code is fed with bad data. For example, you expect the code will input an integer value, but your careless user enters some random letters instead.

When programming errors reveals itself when undesirable code behavior is caused by mistakes you made when you were writing your program. This kind of error is commonly called a “bug”.

When data is not what it should be

```
value = int(input('Enter a natural number: '))
print('The reciprocal of', value, 'is', 1/value)
```

```
try:
    value = int(input('Enter a natural number: '))
    print('The reciprocal of', value, 'is', 1/value)
except:
    print('I do not know what to do.')
```

The *try-except* branch

```
try:
        # It's a place where
        # you can do something
        # without asking for permission.
except:
        # It's a spot dedicated to
        # solemnly begging for forgiveness.
```

How to deal with more than one exception

```
try:
    value = int(input('Enter a natural number: '))
    print('The reciprocal of', value, 'is', 1/value)
except ValueError:
    print('I do not know what to do.')
except ZeroDivisionError:
    print('Division by zero is not allowed in our Universe.')
```

The default exception and how to use it

```
try:
    value = int(input('Enter a natural number: '))
    print('The reciprocal of', value, 'is', 1/value)
except ValueError:
    print('I do not know what to do.')
except ZeroDivisionError:
    print('Division by zero is not allowed in our Universe.')
except:
    print('Something strange has happened here... Sorry!')
```

Some useful exceptions

ZeroDivisionError

This appears when you try to force Python to perform any operation which provokes division in which the divider is zero

ValueError
In general, this exception is raised when a function (like int() or float()) receives an argument of a proper type, but its value is unacceptable.

TypeError
This exception shows up when you try to apply a data whose type cannot be accepted in the current context. Look at the example:
short_list = [1]
one_value = short_list[0.5]

AttributeError
This exception arrives – among other occasions – when you try to activate a method which doesn't exist in an item you're dealing with. For example:
short_list = [1]
short_list.append(2)
short_list.depend(3)

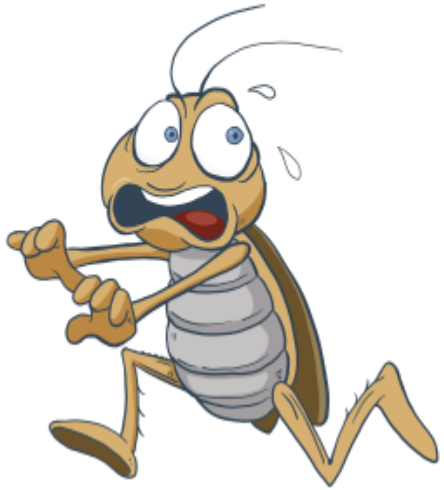
SyntaxError
This exception is raised when the control reaches a line of code which violates Python's grammar.)

Why you can’t avoid testing your code

Don't bury your head in the sand – ignoring errors won't make them disappear.

```
temperature = float(input('Enter current temperature:'))

if temperature > 0:
    print("Above zero")
elif temperature < 0:
    prin("Below zero")
else:
    print("Zero")
```

You already know that your code contains a bug or bugs (the latter is more likely). How do you locate them and how do you fix your code?



Bug vs. debug

The basic measure a developer can use against bugs is – unsurprisingly – a debugger, while the process during which bugs are removed from the code is called debugging.

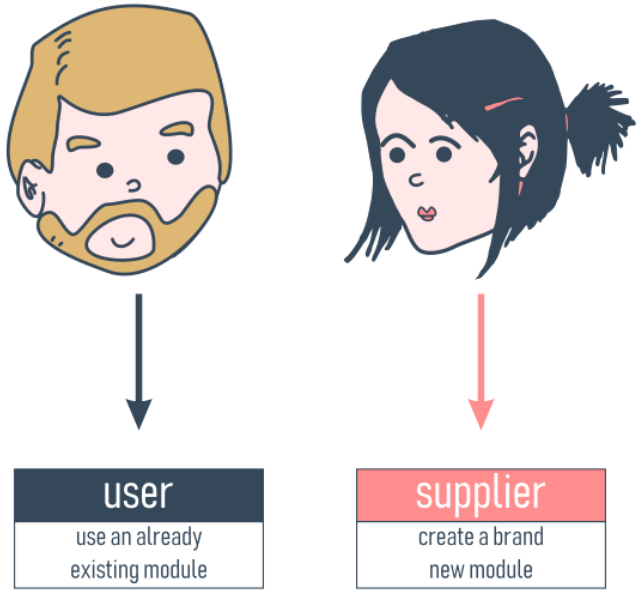
***print* debugging**

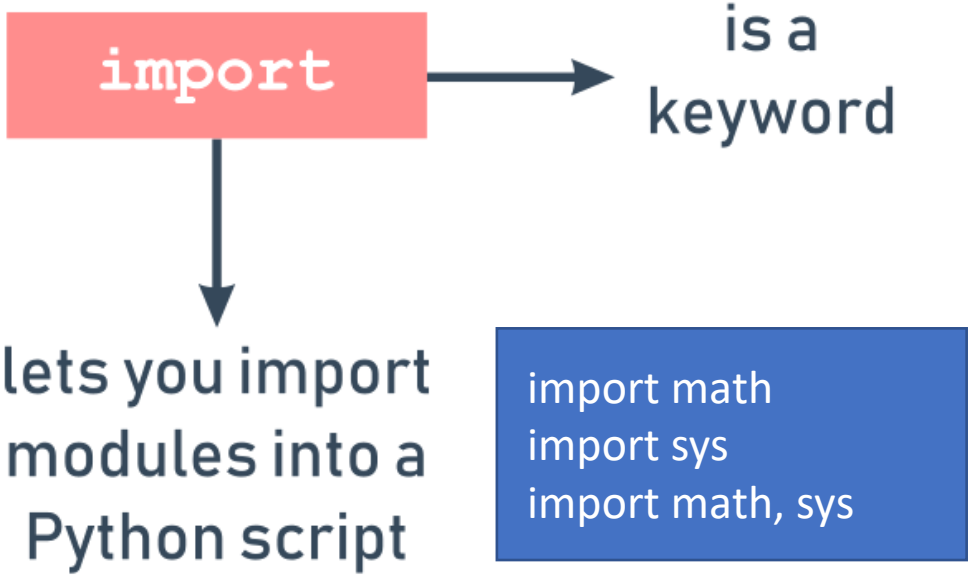
Python provides a dedicated module named **unittest**

What is a module?

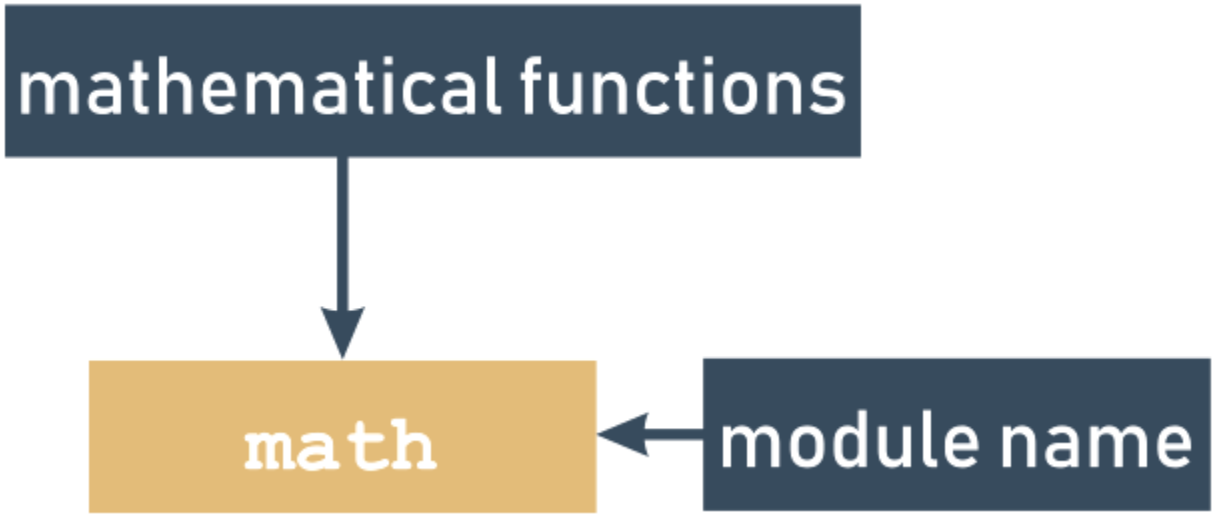
it as a file containing Python definitions and statements, which can be later imported and used when necessary.

https://docs.python.org/3/library/index.html

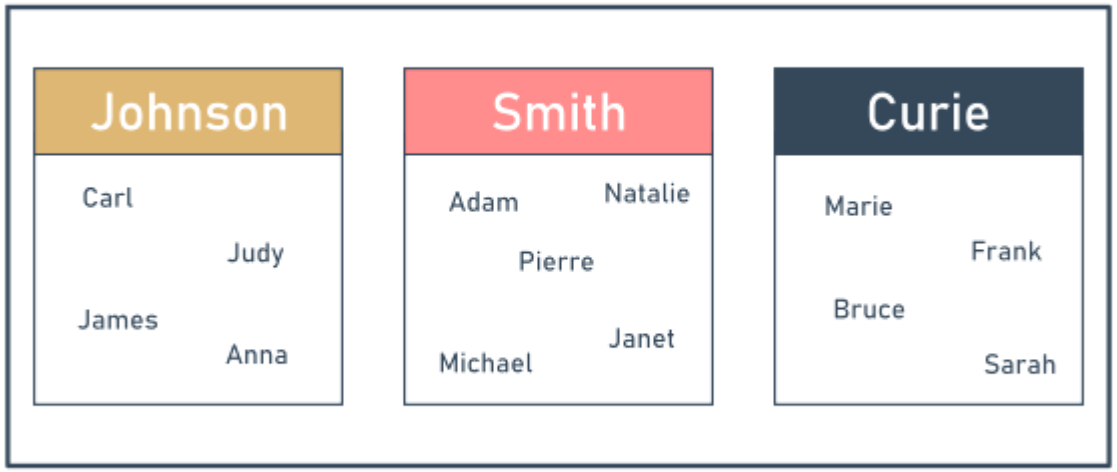





```
import math
import sys
import math, sys
```

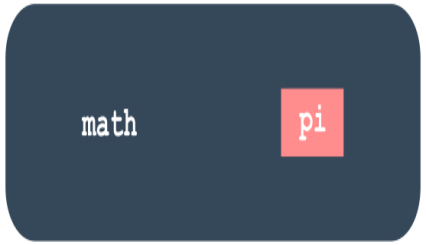


namespace



Importing a module

```
import math
print(math.sin(math.pi/2))
```




```
from module import *
```

```
import math


def sin(x):
    if 2 * x == pi:
        return 0.99999999
    else:
        return None

pi = 3.14

print(sin(pi/2))
print(math.sin(math.pi/2))
```

```
from math import sin, pi

print(sin(pi / 2))

pi = 3.14


def sin(x):
    if 2 * x == pi:
        return 0.99999999
    else:
        return None


print(sin(pi / 2))
```

```
from module import name as alias


from math import pi as PI, sin as sine

print(sine(PI/2)
```

Working with standard modules

dir(module)

```
import math

for name in dir(math):
    print(name, end="\t")
```

```
from math import pi, radians, degrees, sin,
cos, tan, asin

ad = 90
ar = radians(ad)
ad = degrees(ar)

print(ad == 90.)
print(ar == pi / 2.)
print(sin(ar) / cos(ar) == tan(ar))
print(asin(sin(ar)) == ar)
```

```
from math import e, exp, log

print(pow(e, 1) == exp(log(e)))
print(pow(2, 2) == exp(2 * log(2)))
print(log(e, e) == exp(0))
```

```
from math import ceil, floor, trunc

x = 1.4
y = 2.6

print(floor(x), floor(y))
print(floor(-x), floor(-y))
print(ceil(x), ceil(y))
print(ceil(-x), ceil(-y))
print(trunc(x), trunc(y))
print(trunc(-x), trunc(-y))
```

Working with standard modules (cont.)

```
from random import random, seed

seed(0)

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

The seed function

The seed() function is able to directly set the generator's seed.
We'll show you two of its variants:

seed() - sets the seed with the current time;
seed(int_value) - sets the seed with the integer value int_value

```
from random import randrange, randint

print(randrange(1), end=' ')
print(randrange(0, 1), end=' ')
print(randrange(0, 1, 1), end=' ')
print(randint(0, 1))
```

random?

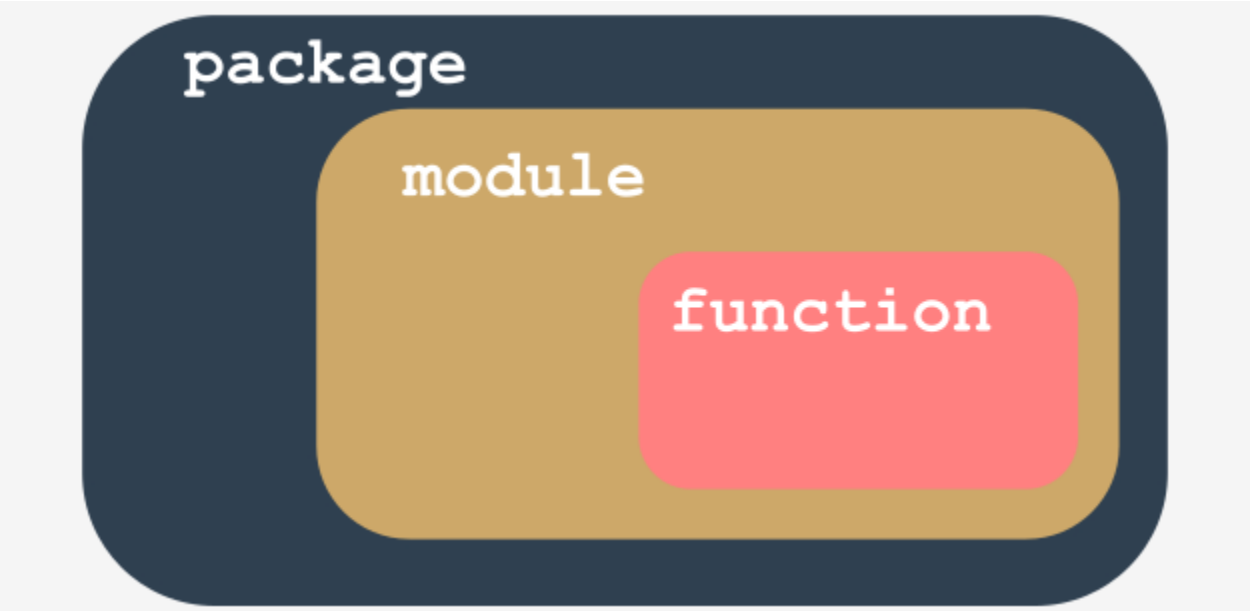
How to know where you are

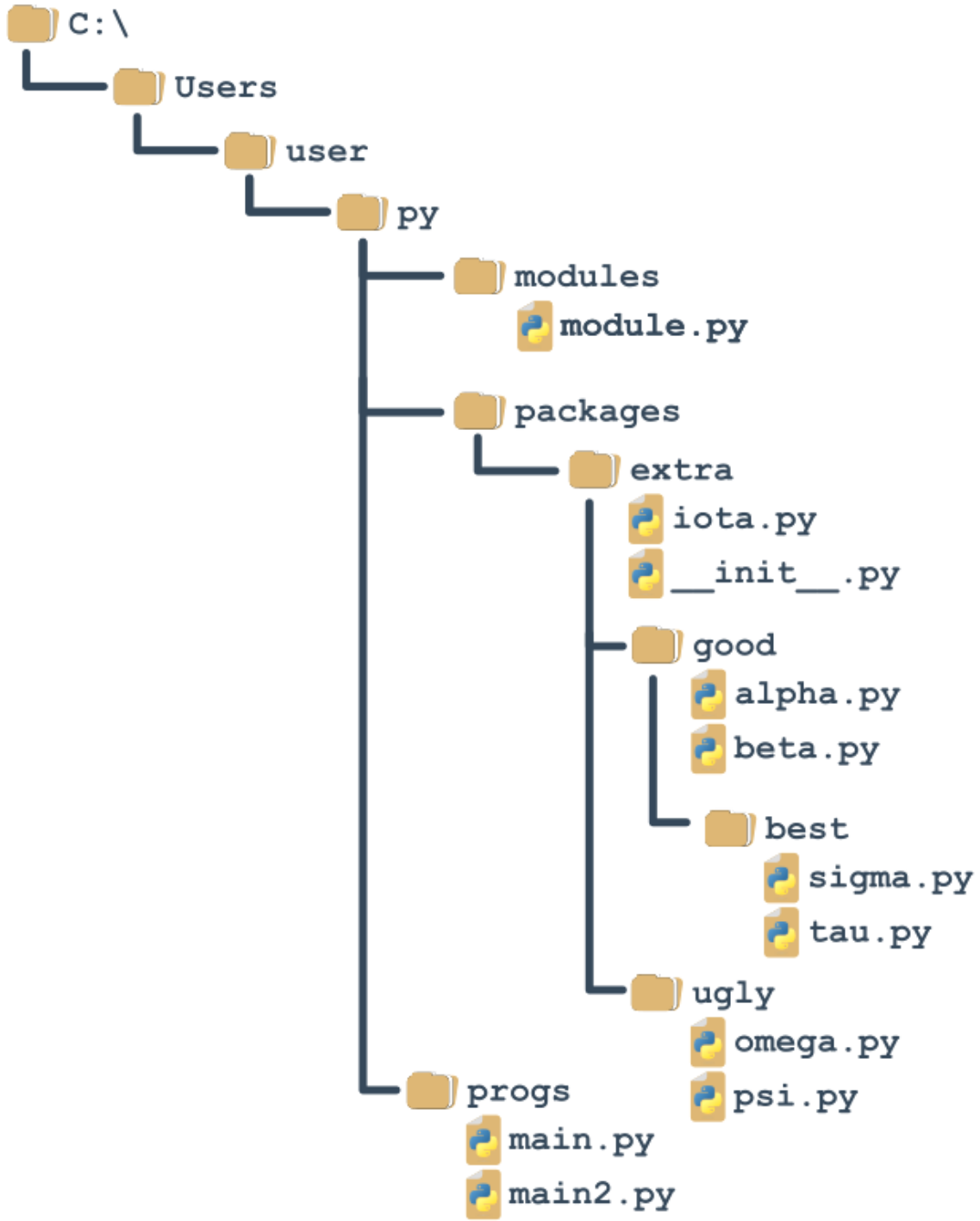
```
from platform import platform

print(platform())
print(platform(1))
print(platform(0, 1))
```

```
from platform import machine
from platform import processor
from platform import system
from platform import version
from platform import python_implementation,
python_version_tuple

print(python_implementation())

for atr in python_version_tuple():
    print(atr)
print(version())
print(system())
print(processor())
print(machine())
```

https://docs.python.org/3/py-modindex.html#cap-m

What is a package





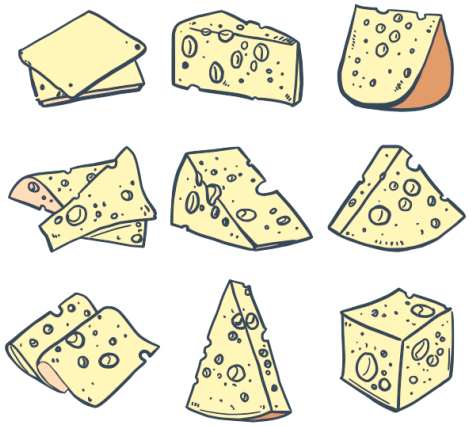
Python packaging ecosystem

You can find their website here:
https://wiki.python.org/psf/PackagingWG.

The PyPI website (administered by PWG) is located at the address:
https://pypi.org/.

The PyPI repo: the Cheese Shop

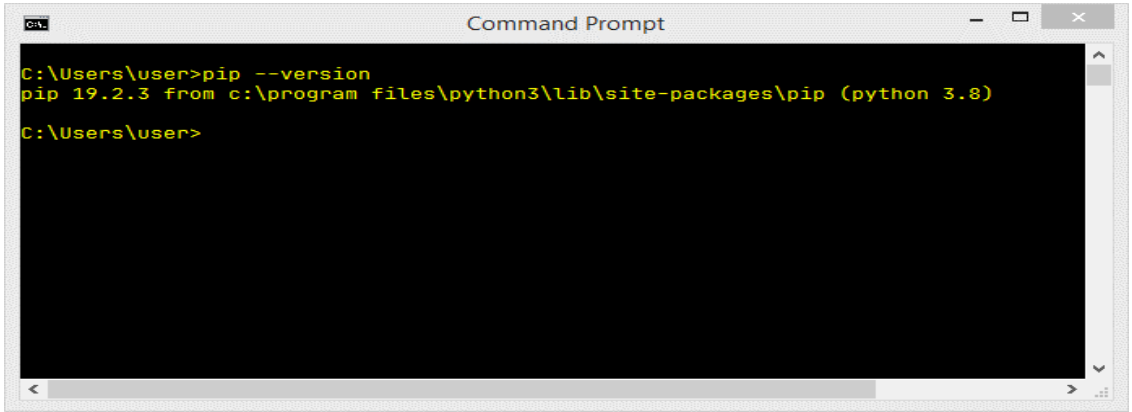
PyPI is completely free, and you can just pick a code and use it – you'll encounter neither cashier nor security guard. Of course, it doesn't absolve you from being polite and honest. You have to obey all the licensing terms, so don't forget to read them.

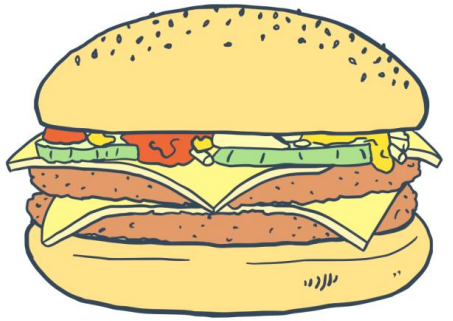


pip

pip means "pip installs packages"

pip --version





See you in the Lab ☺