

## Package Manager in Python



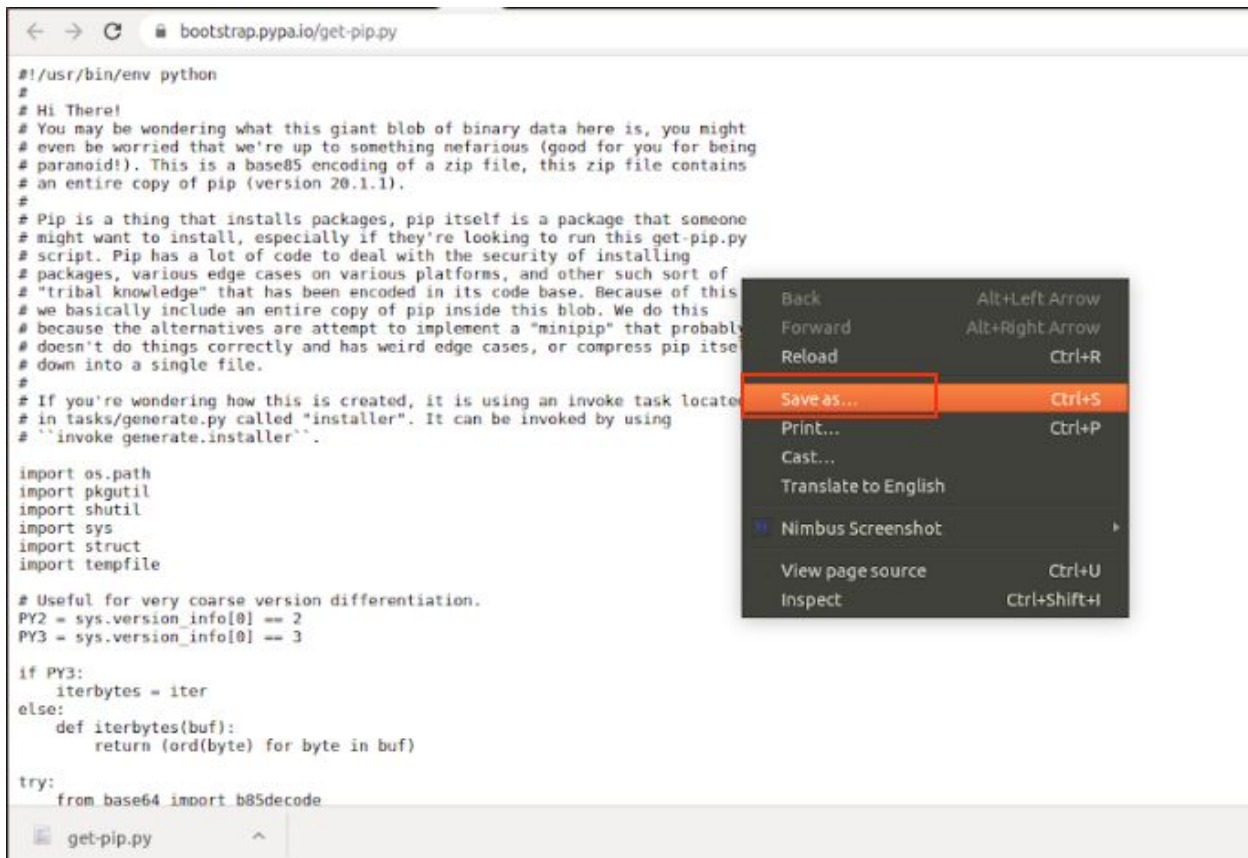
### **What we did:**

In last class we learned about os and shutil modules of python and to backup our files

In this class we learned to about python package manager and howdoi python package

### **How we did it:**

**We installed python package manager in our system**



```

#!/usr/bin/env python
#
# Hi There!
# You may be wondering what this giant blob of binary data here is, you might
# even be worried that we're up to something nefarious (good for you for being
# paranoid!). This is a base85 encoding of a zip file, this zip file contains
# an entire copy of pip (version 20.1.1).
#
# Pip is a thing that installs packages, pip itself is a package that someone
# might want to install, especially if they're looking to run this get-pip.py
# script. Pip has a lot of code to deal with the security of installing
# packages, various edge cases on various platforms, and other such sort of
# "tribal knowledge" that has been encoded in its code base. Because of this
# we basically include an entire copy of pip inside this blob. We do this
# because the alternatives are attempt to implement a "minipip" that probably
# doesn't do things correctly and has weird edge cases, or compress pip itself
# down into a single file.
#
# If you're wondering how this is created, it is using an invoke task located
# in tasks/generate.py called "installer". It can be invoked by using
# ``invoke generate.installer``.

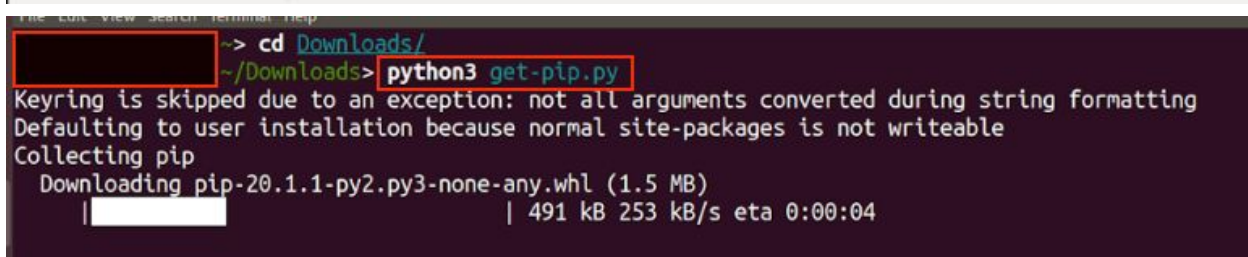
import os.path
import pkgutil
import shutil
import sys
import struct
import tempfile

# Useful for very coarse version differentiation.
PY2 = sys.version_info[0] == 2
PY3 = sys.version_info[0] == 3

if PY3:
    iterbytes = iter
else:
    def iterbytes(buf):
        return (ord(byte) for byte in buf)

try:
    from base64 import b85decode

```

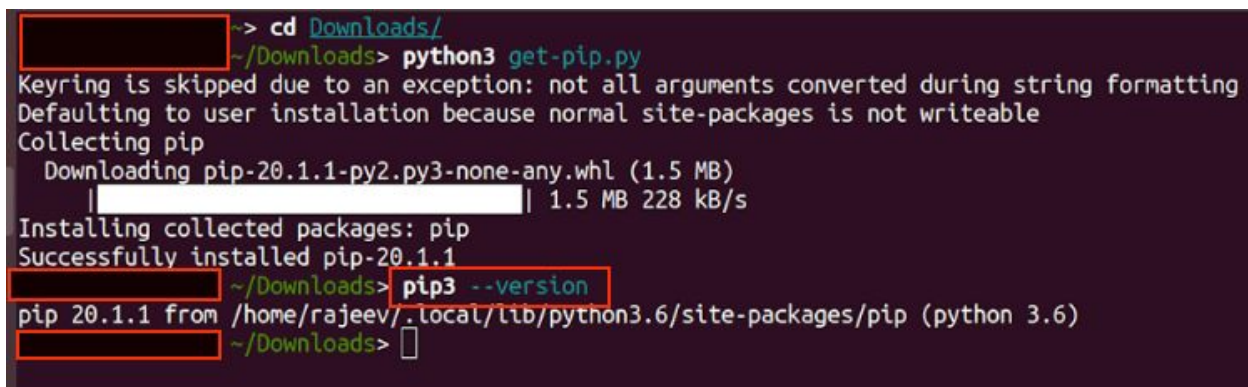


```

~/Downloads> cd Downloads/
~/Downloads> python3 get-pip.py
Keyring is skipped due to an exception: not all arguments converted during string formatting
Defaulting to user installation because normal site-packages is not writeable
Collecting pip
  Downloading pip-20.1.1-py2.py3-none-any.whl (1.5 MB)
    | 491 kB 253 kB/s eta 0:00:04

```

We checked if pip is installed on our system correctly by running command `pip3 --version`



```

~/Downloads> cd Downloads/
~/Downloads> python3 get-pip.py
Keyring is skipped due to an exception: not all arguments converted during string formatting
Defaulting to user installation because normal site-packages is not writeable
Collecting pip
  Downloading pip-20.1.1-py2.py3-none-any.whl (1.5 MB)
    | 1.5 MB 228 kB/s
Installing collected packages: pip
Successfully installed pip-20.1.1
~/Downloads> pip3 --version
pip 20.1.1 from /home/rajeev/.local/lib/python3.6/site-packages/pip (python 3.6)
~/Downloads>

```

We installed howdoi package to our system

```
~/Downloads> pip3 install howdoi
Keyring is skipped due to an exception: not all arguments converted during string formatting
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: howdoi in /home/rajeev/.local/lib/python3.6/site-packages (1.2.1)
Requirement already satisfied: appdirs in /home/rajeev/.local/lib/python3.6/site-packages (from howdoi) (1.4.4)
Requirement already satisfied: requests in /home/rajeev/.local/lib/python3.6/site-packages (from howdoi) (2.23.0)
Requirement already satisfied: cachelib in /home/rajeev/.local/lib/python3.6/site-packages (from howdoi) (0.1)
Requirement already satisfied: pyquery in /home/rajeev/.local/lib/python3.6/site-packages (from howdoi) (1.4.1)
Requirement already satisfied: pygments in /home/rajeev/.local/lib/python3.6/site-packages (from howdoi) (2.6.1)
Requirement already satisfied: idna<3,>=2.5 in /home/rajeev/.local/lib/python3.6/site-packages (from requests->howdoi) (2.9)
Requirement already satisfied: urllib3!=1.25.0,!<1.25.1,<1.26,>=1.21.1 in /home/rajeev/.local/lib/python3.6/site-packages (from requests->howdoi) (1.25.9)
Requirement already satisfied: certifi>=2017.4.17 in /home/rajeev/.local/lib/python3.6/site-packages (from requests->howdoi) (2020.4.5.1)
Requirement already satisfied: chardet<4,>=3.0.2 in /home/rajeev/.local/lib/python3.6/site-packages (from requests->howdoi) (3.0.4)
```

We saw that howdoi provides us the best results and saw its usage

```
~> howdoi write python function
class Line:
    def __init__(self, m, b):
        self.m = m
        self.b = b
    def __call__(self, x):
        return self.m * x + self.b
rajeev@atlantis ~>
```

We explored more using howdoi

```
~> howdoi declare variables in python
foo = 'bar' # the name 'foo' is now a name for the string 'bar'
foo = 2 * 3 # the name 'foo' stops being a name for the string 'bar',
# and starts being a name for the integer 6, resulting from the multiplication
rajeev@atlantis ~>
```

We used howdoi to see how to write class in python



```

~> howdoi write class in python

class Student(object):
    def __init__(self, name, age, gender, level, grades=None):
        self.name = name
        self.age = age
        self.gender = gender
        self.level = level
        self.grades = grades or {}

    def setGrade(self, course, grade):
        self.grades[course] = grade

    def getGrade(self, course):
        return self.grades[course]

    def getGPA(self):
        return sum(self.grades.values())/len(self.grades)

# Define some students
john = Student("John", 12, "male", 6, {"math":3.3})
jane = Student("Jane", 12, "female", 6, {"math":3.5})

# Now we can get to the grades easily
print(john.getGPA())
print(jane.getGPA())
rajeev@atlantis ~>

```

We defined class student

```

class Student(object):
    def __init__(self, name, age, gender, level, grades=None):
        self.name = name
        self.age = age
        self.gender = gender
        self.level = level
        self.grades = grades or {}

```

We also saw the usage of “self”

```

def setGrade(self, course, grade):
    self.grades[course] = grade

def getGrade(self, course):
    return self.grades[course]

def getGPA(self):
    return sum(self.grades.values())/len(self.grades)

```

We wrote code for our own class Car

```
1 class Car(object):  
2     """
```

Used the constructor to create object

```
def __init__(self, model, color, company, speed_limit):  
    self.color = color  
    self.company = company  
    self.speed_limit = speed_limit  
    self.model = model
```

Defined the methods in the class

```
6 def __init__(self, model, color, company, speed_limit):  
7     self.color = color  
8     self.company = company  
9     self.speed_limit = speed_limit  
10    self.model = model  
11  
12    def start(self):  
13        print("started")  
14  
15    def stop(self):  
16        print("stopped")  
17  
18    def accelerate(self):  
19        print("accelarating...")  
20        "accelarator functionality here"  
21  
22    def change_gear(self, gear_type):  
23        print("gear changed")  
24        " gear related functionality here"
```

We checked the output

```
>>> audi = Car("A6", "red", "audi", "80")  
>>> audi.color  
'red'  
>>> 
```

**What's next?**

In the next class, we will learn to store our files on cloud storage.

