Előadás -4

Programozás Pythonban

Instructor : Dr. AALWAHAB DHULFIQAR

Advisor : Dr. Tejfel Mate

Mit fogunk tanulni:

Tuple-ek

Szótárak

Kivételek

Adatfeldolgozás

Modulok, csomagok és PIP

Sorozattípusok és változtathatóság-> Sequence types and mutability

A szekvencia típus egy olyan adattípus a Pythonban, amely egynél több értéket képes tárolni (vagy egynél kevesebbet, mivel a szekvencia lehet üres), és ezeket az értékeket szekvenciálisan (innen a neve) elemenként lehet böngészni.

Mutabilitás - a Python bármely adatának olyan jellemzője, amely bemutatja, hogy a program végrehajtása során szabadon megváltoztatható. Kétféle Python-adat létezik: a változtatható és a megváltoztathatatlan (mutable and immutable).

Tuple

tuples inkább zárójelet ()használ
A tuple-ok szekvenciák.

A tuplik megváltoztathatatlanok

.

```
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

A szótár egy másik Python adatszerkezet. Nem szekvencia típus (de könnyen adaptálható a szekvencia feldolgozásához) és megváltoztatható.

A szótár kulcs-érték párok halmaza.
Megjegyzések:

1. Minden kulcsnak egyedinek kell lennie - nem lehet, hogy egynél több azonos értékű kulcs legyen
2. A kulcs bármilyen típusú objektum lehet: lehet szám (egész vagy float), vagy akár egy karakterlánc, de nem lehet lista;
3. A szótár nem lista - a lista számozott értékek halmazát tartalmazza, míg a szótár értékpárokat;
4. A len() függvény szótárak esetén is működik - a szótár kulcs-érték elemeinek számát adja vissza;
5. A szótár egyirányú eszköz - ha van egy angol-francia szótár, akkor az angol kifejezések francia megfelelőit keresheted, de fordítva nem.

```
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)
```

értékek módosítása és hozzáadása

```
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)
```

Tupelek és szótárak együtt is működhetnek

```
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)
```

Hibák - a programozó mindennapi kenyere

Hibák az adatokban vs. hibák a kódban

A programozási hibák kezelésének (legalább) két oldala van.

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”.

Amikor az adatok nem olyanok, mint kellene, hogy legyenek

```
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*

```
try:
        # Ez egy olyan hely
        # ahol tehetsz valamit.
        # anélkül, hogy engedélyt kérnél.
except:
        # Ez egy olyan hely, amelyet a
        # bocsánat kerhetsz.
```

Hogyan kezeljük az egynél több kivételt

```
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.')
```

Az alapvető kivétel és használata

```
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!')
```

Pár hasznos kivétel

ZeroDivisionError

Ez akkor jelenik meg, ha megpróbálod a Pythont olyan művelet végrehajtására rávenni, amely olyan osztást idéz elő, amelyben az osztó nulla.

ValueError

Általában ez a kivétel akkor lép fel, ha egy függvény (mint például az int() vagy a float()) megfelelő típusú argumentumot kap, de annak értéke elfogadhatatlan.

TypeError

Ez a kivétel akkor jelenik meg, amikor olyan adatot próbálsz alkalmazni, amelynek típusa nem fogadható el az aktuális kontextusban. Nézze meg a példát:

```
short_list = [1]
one_value = short_list[0.5]
```

AttributeError

Ez a kivétel - többek között - akkor jelentkezik, amikor olyan módszert próbálsz aktiválni, amely nem létezik a kezelt elemben. Például:

```
short_list = [1]
short_list.append(2)
short_list.depend(3)
```

SyntaxError

Ez a kivétel akkor lép fel, ha a programozás olyan kódsorhoz ér, amely sérti a Python nyelvtanát.).

Miért nem kerülheted el a kódod tesztelését

Ne dugja homokba a fejét - a hibák figyelmen kívül hagyása nem fogja eltüntetni azokat.

```
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

A programozó alapvető eszköze a bugok ellen - nem meglepő módon - a debugger, míg azt a folyamatot, amelynek során a hibákat eltávolítják a kódból, debuggingnak nevezzük.

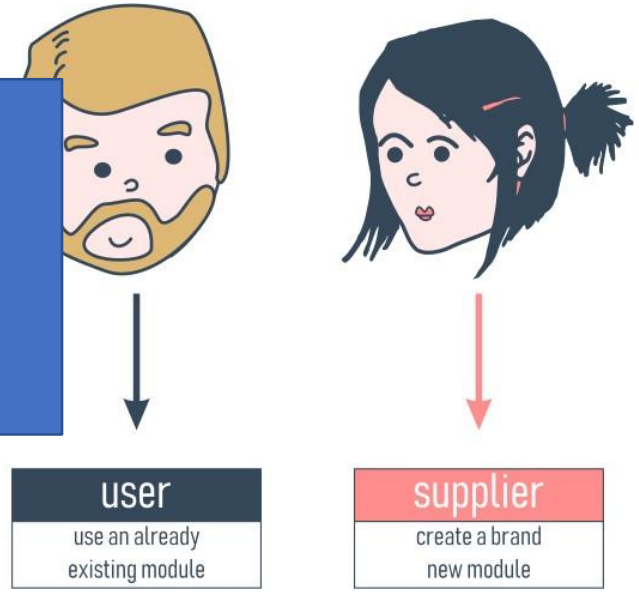
***print* debugging**

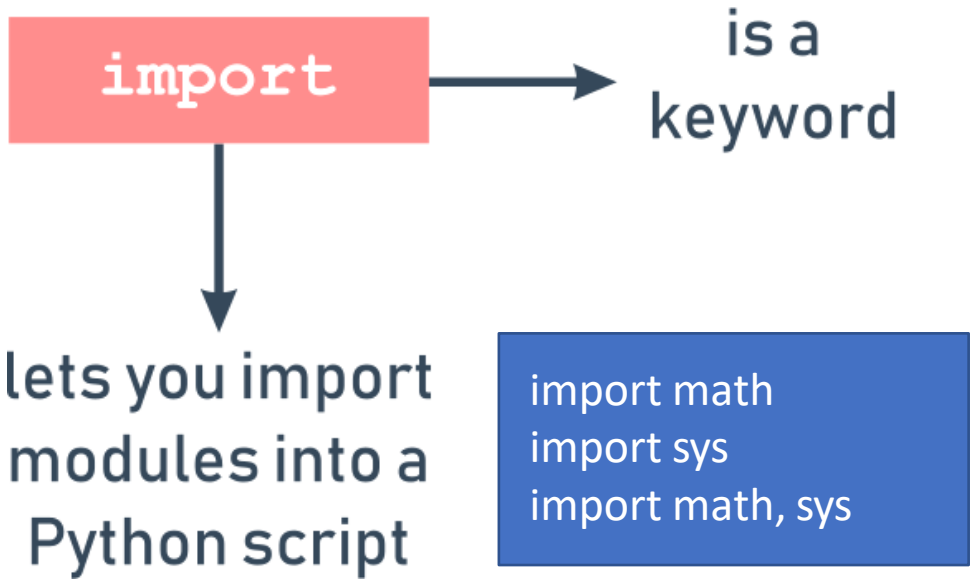
A Python egy unittest nevű dedikált modult tartalmaz.

What is a module?

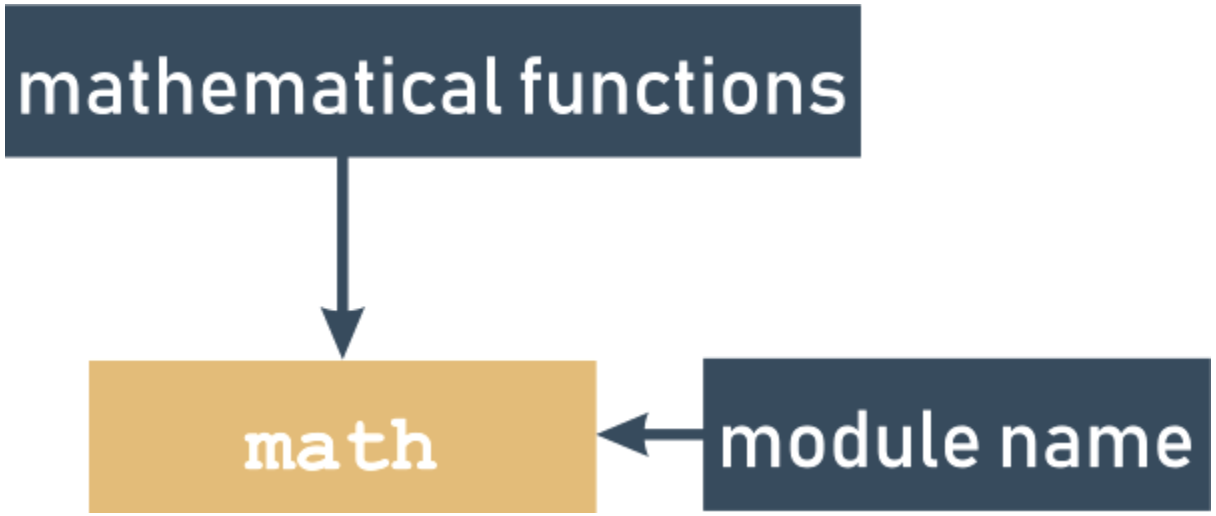
fájlként, amely Python definíciókat és utasításokat tartalmaz, amelyeket később importálhatunk és használhatunk, ha szükséges.

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

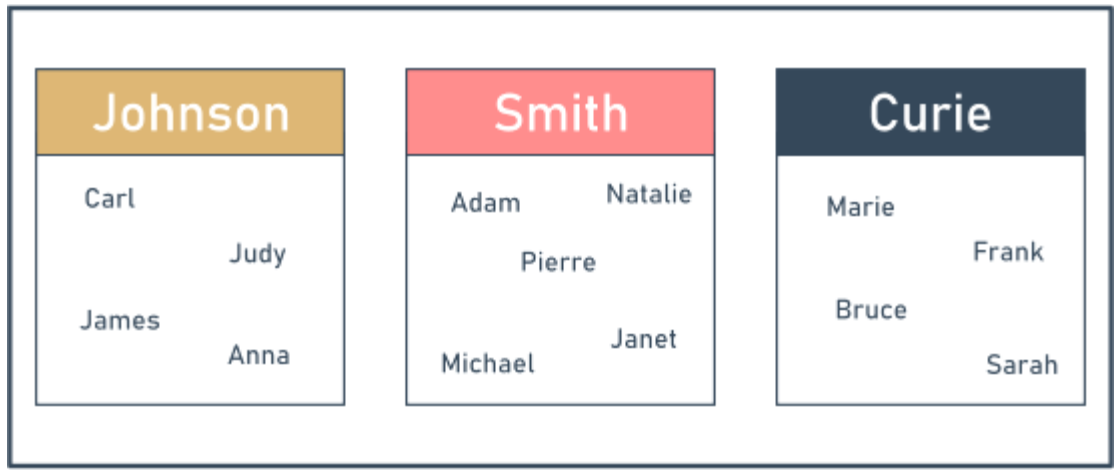





```
import math
import sys
import math, sys
```

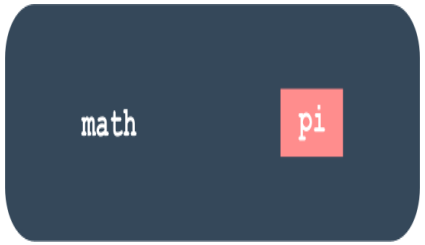


Namespace--- névtér



Modul importálása

```
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)
```

Dolgozás alap modulokkal

```
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))
```

Dolgozás alap modulokkal (folytatás)

```
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))
```

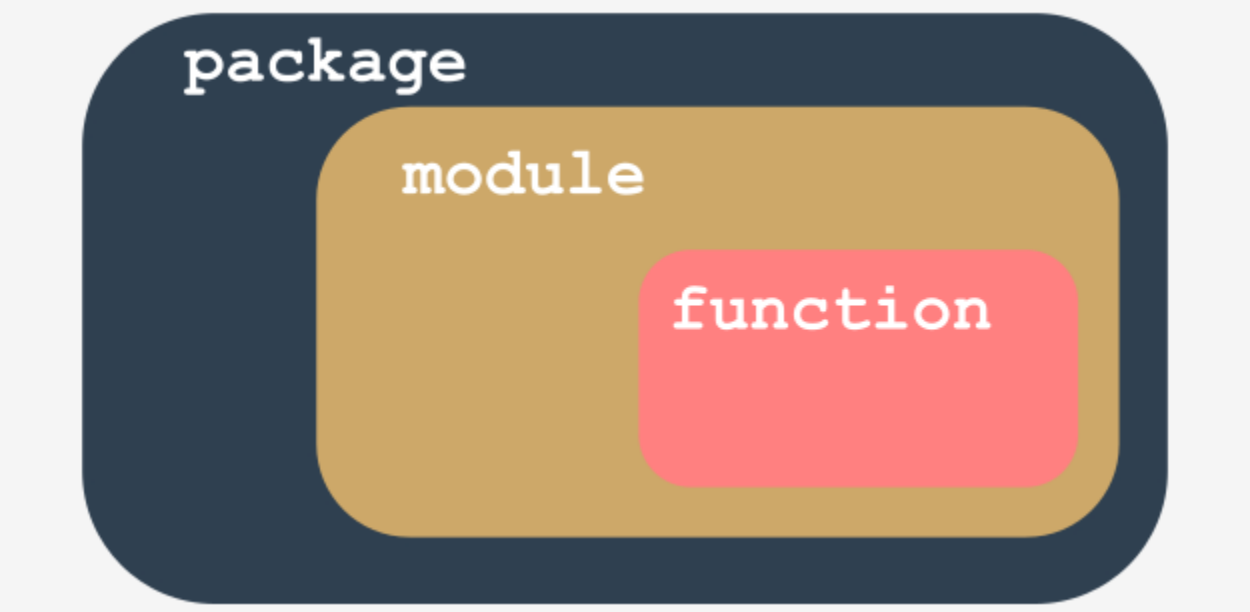
Hogyan lehet tudni, hogy hol vagy pythonból

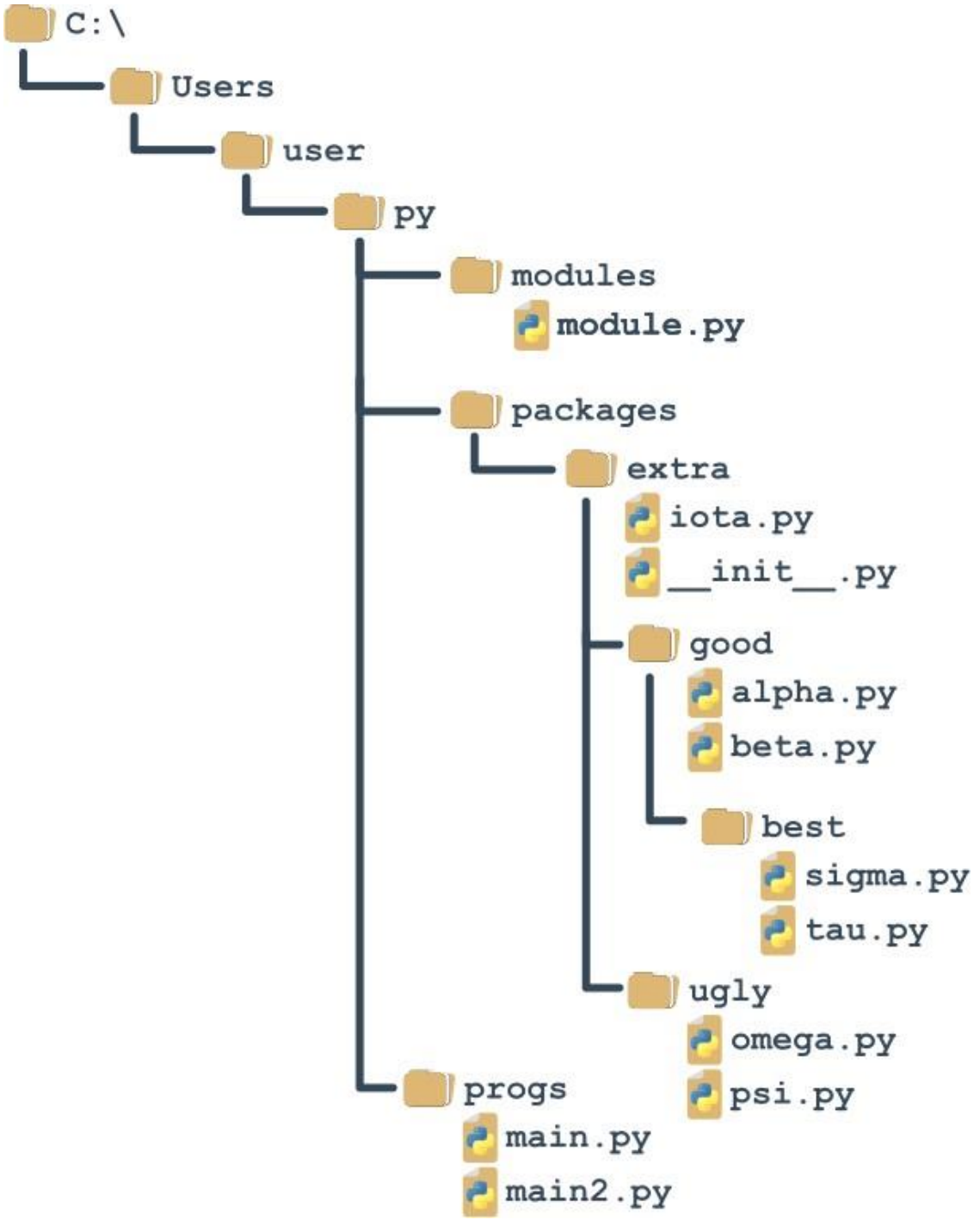
```
from platform import platform

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

Mi az a csomag

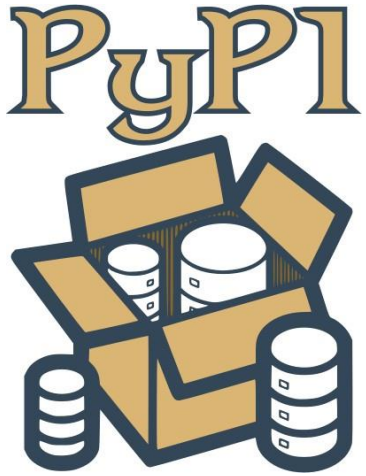




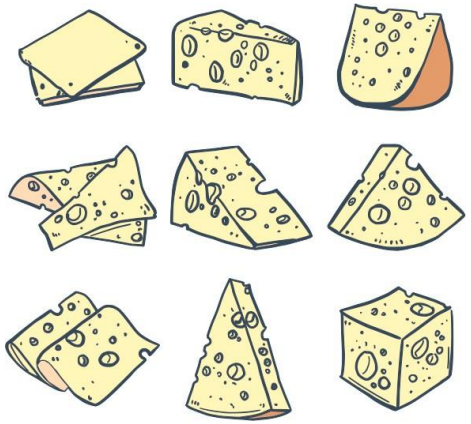
Python packaging ecosystem

Honlapjukat itt találjátok:
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

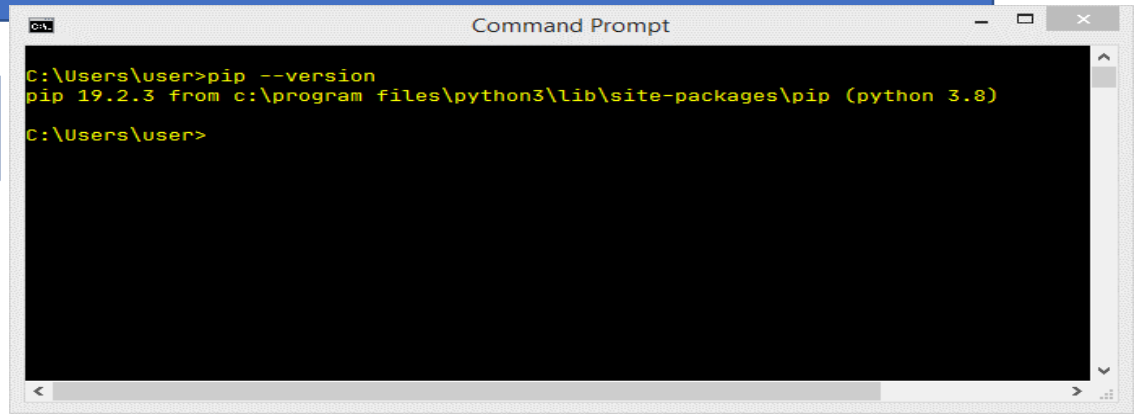


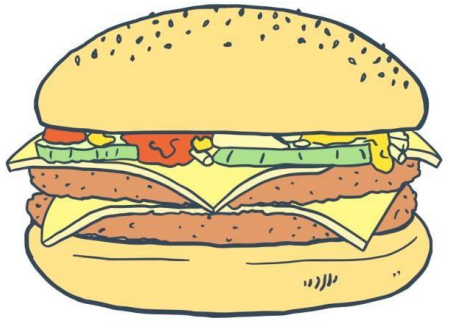
A PyPI teljesen ingyenes, és csak kiválasztasz egy kódot, és használhatod - nem találkozol sem pénztárossal, sem biztonsági őrrel. Persze ez nem mentesít az udvariasság és az őszinteség alól. Be kell tartanod az összes licencfeltételt, úgyhogy ne felejtsd el elolvasni őket.
.

pip

pip means “pip installs packages”

pip --version





Találkozunk a laborban ☺