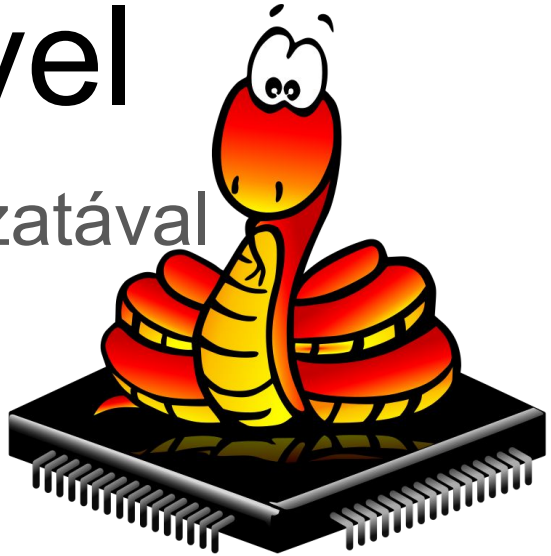
Ismerkedés a Python programnyelvvel

és annak micropython változatával



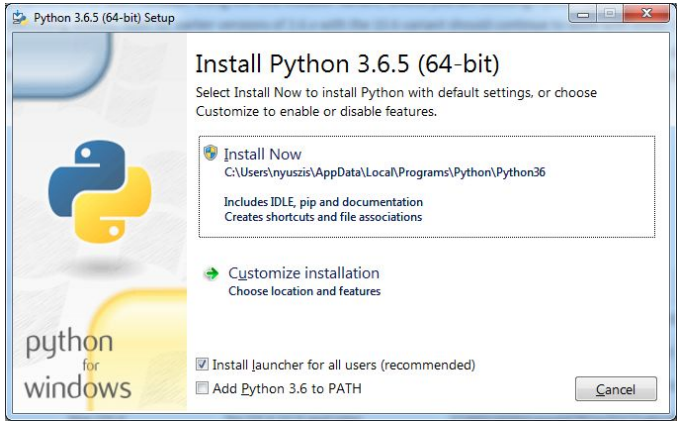
A Python programozási nyelv története

- Az alapötlet 1980-ban született, 1989 decemberében kezdte el fejleszteni Guido van Rossum a CWI-n holland Matematikai és Informatikai központban
- A nevét a Monty Python Flying Circus BBC műsorról kapta
- 1991-ben jelent meg a 0.9.0-ás verzió
- 1994-ben 1.0 verzió, lambda, map, reduce
  CP4A (Computer Programming for Everybody)
  1.6.1 GPL licence
- 2000 októberben 2.0-ás verzió, gc, unicode
- 2008 decemberében a python 3.0, nem kompatibilis a korábbival

A Python főbb jellemzői

- A cél az volt, hogy könnyen olvasható kódot készíthetünk benne
  a kód behúzásai jelölik a kódblokkot
- script nyelv, előfordított kódot futtat (megvalósításfüggő)
- hordozható, windows, linux, macos, létezik java kódot generáló jpython
- Dinamikus változó és típuskezelés, az alaptípusok érték szerint adódnak át,
  így összehasonlíthatóak, a többi típusnál referenciát tárolnak (vitatható)
- hivatkozás számláló mechanizmus, egyfajta szemétgyűjtő algoritmus
- objektum orientált nyelv, támogatja a többszörös öröklést
- bővíthető, könnyen illeszthető meglévő C könytárakhoz
- csomagkezelővel rendelkezik (pip install)
- ingyenes

Python beszerzése, futtatása PC-n



Letölthető, oprendszer és géptípus választás után

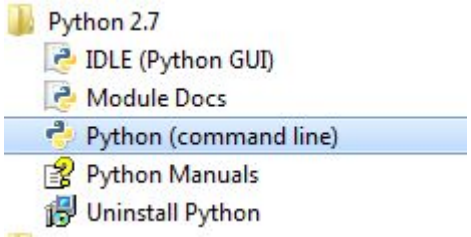
https://www.python.org/downloads/windows/

érdemes MSI installer választani windows alatt

Linux alatt a disztribúció csomagkezelőjével

Telepítése:

MSI esetén a tovább (next) gomb nyomogatása



Futtatás:

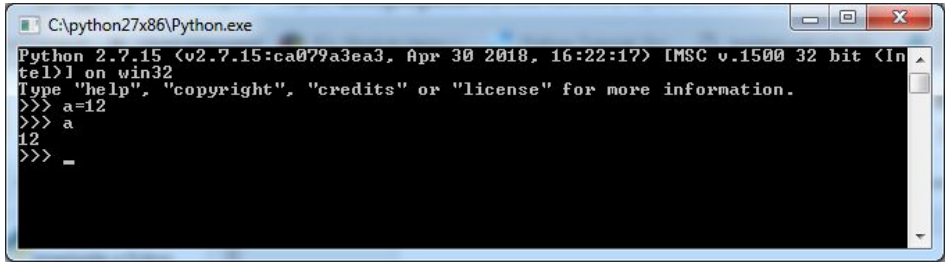
cmd ablakban python szó beírásával,

vagy Start menü Python (command line) kiválasztása

Első python program futtatása

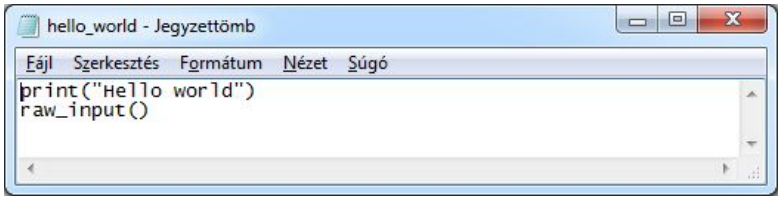
Interaktív REPL (read-eval-print loop)

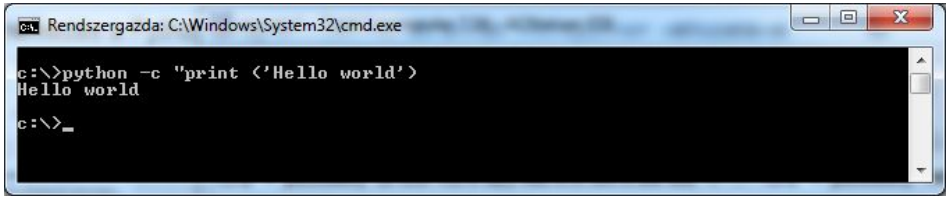
A változó értékét a változó beírásával adja vissza



Script mód

A megírt script futtatása cmd ablakban





Parancssori paraméterek, pl. -c

Nyelv foglalt szavai és blokkok létrehozása

Foglalt szavak listája

```
and       del       from      not       while
as        elif      global    or        with
assert    else      if        pass      yield
break     except    import    print
class     exec      in        raise
continue  finally   is        return
def       for       lambda    try
```

Blokkok létrehozása

A blokkok minden esetben egy utasítással és egy kettősponttal kezdődnek, ez a fejléce

A blokk belső sorai ugyanannyi behúzással kell kezdődjön, a tabulátor nem helyettesíthető szóközökkel. Érdemes a szövegszerkesztőt úgy beállítani, hogy a begépelt tabulátorokat x szóközre cserélje automatikusan.

```
fejsor:
   blokk első sora
   ...
   blokk utolsó sora
```

Változók elnevezése adattípusai, tárolása

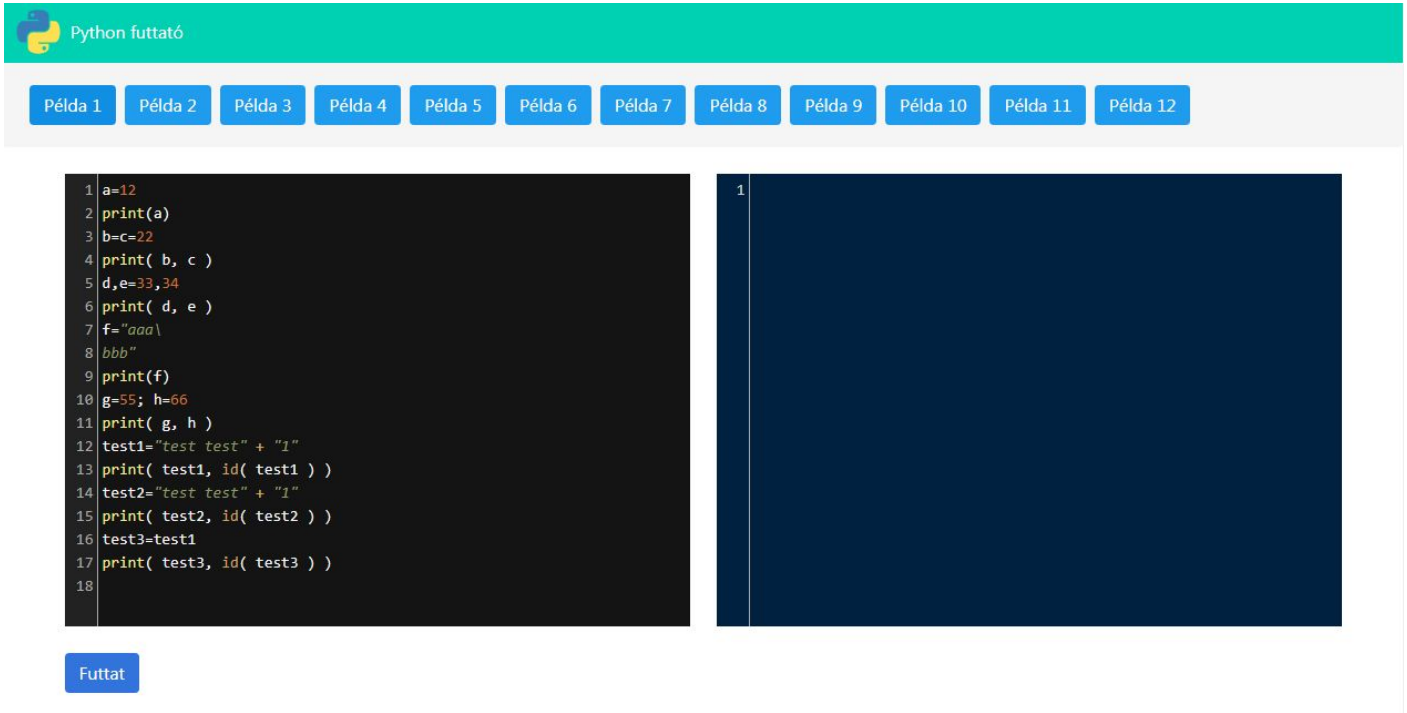
Minden python értéknek van egy adattípusa. Minden adattípus egy osztály és a változók ezeknek az osztályoknak egy-egy példányai, amik a memóriában egy id-val azonosított területen foglalnak helyet.

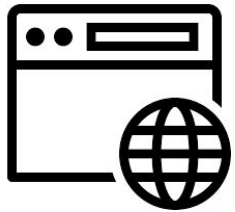
```
1 , tipus :  <class 'int'>
1.0 , tipus :  <class 'float'>
1 , id-je :  1647143968
1.0 , id-je :  33755376
1.0 == 1 Igaz
```

A változók elnevezése a foglalt szavakon kívüli szavakkal lehetséges, kis és nagy betű különböző-nek számít és betűvel vagy _ kell kezdődjön, ezen felül tartalmazhat és kezdődhet is _ (aláhúzás) karakterrel. Létezik ajánlás a python nyelvhez PEP8 (Python Javító Javaslatok) amely a változók elnevezéséhez is ad javaslatot.

Python példák a böngészőben

a **pyserv.py** program elindítása után írjuk a böngésző címsorába: http://localhost:8000





Értékadások, kifejezések több sorban, változók tárolásának módja

```
1  a=12
2  print( a )
3  b=c=22
4  print( b, c )
5  d,e=33,44
6  print( d, e )
7  f="aaa\
8  bbb"
9  print(f)
10 g=55; h=66
11 print( g, h )
12 test1="test test" + "1"
13 print( test1, id(test1) )
14 test2="test test" + "1"
15 print( test2, id(test2) )
16 test3=test1
17 print( test3, id(test3) )
```

```
1  12
2  22 22
3  33 44
4  aaabbb
5  55 66
6  test test1 49839920
7  test test1 49873136
8  test test1 49839920
9
```

Egyszerű adattípusok, literálok

```
1  print(type(1),1)
2  print(type(0b011),0b011)
3  print(type(0xa),0x0a)
4  print(type(0o15),0o15)
5  print(type(1.0),1.0)
6  print(type(-1+0j),-1+0j)
7  print(type(True),True)
8  print(type(1==1),1==1)
9  print(type("string"),"string")
10 print(type(u"string"),u"string")
11 print(type([1,2]),[1,2])
12 print(type((1,2)),(1,2))
13 print(type({1,2}),{1,2})
14 print(type({"a":"a","b":"b"}),{"a":"a","b":"b"})
15 class a:
16   pass
17 print(type(a),a)
18 print(type(a()),a(),id(a()))
19
```

```
1  <class 'int'> 1
2  <class 'int'> 3
3  <class 'int'> 10
4  <class 'int'> 13
5  <class 'float'> 1.0
6  <class 'complex'> (-1+0j)
7  <class 'bool'> True
8  <class 'bool'> True
9  <class 'str'> string
10 <class 'str'> string
11 <class 'list'> [1, 2]
12 <class 'tuple'> (1, 2)
13 <class 'set'> {1, 2}
14 <class 'dict'> {'a': 'a', 'b': 'b'}
15 <class 'type'> <class '__main__.a'>
16 <class '__main__.a'> <__main__.a object at
   0x0000000003015C88> 50421096
17
```

Tuple (véges rendezett lista) és set (halmaz)

```
1  print("-- tuple --")
2  print( ("egy",) )
3  print( ("egy","kettő") )
4  print( ("egy","kettő")[0] )
5  print( len( ("egy","kettő") ) )
6  print("-- set --")
7  print( {"egy"} )
8  print( {"egy","kettő"} )
9  print( "egy" in  {"egy","kettő"} )
10 print( {1,2,3} | {3,4,5})
11 print( {1,2,3} - {3,4,5})
12 print( {1,2,3} & {3,4,5})
13 print( {1,2,3} ^  {3,4,5})
14 print("-- halmazértelmezés --")
15 print( {x for x in {1,2,3} if x not in {1,2}} )
16 print("-- halmazértelmezés, kimenete lista --")
17 print( [x for x in {1,2,3} if x not in {1,2}] )
```

```
1  -- tuple --
2  ('egy',)
3  ('egy', 'kettő')
4  egy
5  2
6  -- set --
7  {'egy'}
8  {'egy', 'kettő'}
9  True
10 {1, 2, 3, 4, 5}
11 {1, 2}
12 {3}
13 {1, 2, 4, 5}
14 -- halmazértelmezés --
15 {3}
16 -- halmazértelmezés, kimenete lista --
17 [3]
18
```

List lista, sorozat vagy tömb

```
1  print("-- list --")
2  print( ["egy","kettő"] )
3  print( ["egy","kettő"][0] )
4  a=["egy","kettő"] ; print( len( a ) )
5  a.append("három") ; print( a )
6  a.pop() ; print( a )
7  a.remove("egy") ; print( a )
8  a.insert(0,"egy") ; print( a )
9  a.extend(["három","négy"]) ; print( a )
10 a.reverse() ; print( a )
11 print(a [1:3] )
12 print(a[2:] , a[:2] , a[-1])
13 del( a[0] ) ; print(a)
14 print("-- listaértelmezés --")
15 print( [ x**2 for x in [1,2,3] ] )
16 print("-- listává alakítás --")
17 print( list(range(4)) )
18 print( list( (1,2,3) ) )
19 print( [0] * 3 )
```

```
1  -- list --
2  ['egy', 'kettő']
3  egy
4  2
5  ['egy', 'kettő', 'három']
6  ['egy', 'kettő']
7  ['kettő']
8  ['egy', 'kettő']
9  ['egy', 'kettő', 'három', 'négy']
10 ['négy', 'három', 'kettő', 'egy']
11 ['három', 'kettő']
12 ['kettő', 'egy'] ['négy', 'három'] egy
13 ['három', 'kettő', 'egy']
14 -- listaértelmezés --
15 [1, 4, 9]
16 -- listává alakítás --
17 [0, 1, 2, 3]
18 [1, 2, 3]
19 [0, 0, 0]
```

Dictionary, szótár vagy asszociatív tömb

```
print("-- dict --")
print( {'a': 1 , 'b': 2} )
a={'a': 1 , 'b': 2} ; print( a['a'] )
print( a.keys(), a.values() )
a['c']=3 ; print( a )
del a['c']; print( a )
a.update({'d' : 4}) ; print( a )
print( *a )
c={**a,**{'e':5}} ; print( c )
print( dict([('a',1) , ('b',2) , ('c',3) ]) )
print( list(enumerate(a)) )
print( 'd' in a , 'z' in a )
print( a.get('d') , a.get('z') )
print( a, len( a ) )
print("-- szótárértelmezés --")
print( {k: v*2 for k,v in a.items()} )
```

```
-- dict --
{'a': 1, 'b': 2}
1
dict_keys(['a', 'b']) dict_values([1, 2])
{'a': 1, 'b': 2, 'c': 3}
{'a': 1, 'b': 2}
{'a': 1, 'b': 2, 'd': 4}
a b d
{'a': 1, 'b': 2, 'd': 4, 'e': 5}
{'a': 1, 'b': 2, 'c': 3}
[(0, 'a'), (1, 'b'), (2, 'd')]
True False
4 None
{'a': 1, 'b': 2, 'd': 4} 3
-- szótárértelmezés --
{'a': 2, 'b': 4, 'd': 8}
```

String, karaterlánc

```python
print("-- string --")
print("szia")
print('szia')
print('"szia"' , "'szia'" , "\"szia\"")
print("""szia
python""")
print("szia" + " python")
print("szia" " python")
print("szia\tpython")
print("szia"[-1])
print("szia"[1:])
print("szia"[5:])
print( len("szia") )
print(list("szia") )
print(set("szeptember") )
print("szia python".split(" "))
print("szia python".replace("s","z"))
print("szia python".upper())

```

```
-- string --
szia
szia
"szia" 'szia' "szia"
szia
python
szia python
szia python
szia    python
a
zia

4
['s', 'z', 'i', 'a']
{'p', 's', 'e', 'z', 't', 'm', 'r', 'b'}
['szia', 'python']
zzia python
SZIA PYTHON

```

Utasítások, If feltételes elágazás és a for ciklus

```
1  print("-- if --") ; x=2
2  if x>0:
3    print( "pozitív" )
4  elif x<0:
5    print( "negatív" )
6  else:
7    print( "nulla" )
8  print( ("nagyobb" if x>1 else "kisebb")+ " mint "\
9  "egy")
10 print("-- for --")
11 for i in [1,2,3]:
12   print( i )
13 a = {"a" : 1 , "b" : 2 , "c" : 3}
14 for k,v in a.items():
15   print( k , ":" , v )
16 for i in range(1,4): print(i)
17 for i in range(1,4):
18   if i>2: break
19   print( i )
```

```
1  -- if --
2  pozitív
3  nagyobb mint egy
4  -- for --
5  1
6  2
7  3
8  a : 1
9  b : 2
10 c : 3
11 1
12 2
13 3
14 1
15 2
16
```

Utasítások, While ciklus, függvény definiálás, paraméterátadás

```
1  print("-- while --")
2  a=1
3  while a<=3:
4    print(a)
5    a+=1
6  while a>0:
7    a-=1
8    if a>2: continue
9    print( a )
10 print("-- def --")
11 def fugv():
12   return 1
13 print( fugv() )
14 def fugv(*arg):
15   return arg
16 print( fugv(1,2,3) )
17 def fugv(**arg):
18   return arg
19 print( fugv(a=1,b=2) )
```

```
1  -- while --
2  1
3  2
4  3
5  2
6  1
7  0
8  -- def --
9  1
10 (1, 2, 3)
11 {'a': 1, 'b': 2}
12
```

Példa 8

Modulok használata, kivételkezelés

```
1  print("-- modul --")
2  import math
3  print( math.floor(1.6) )
4  from math import floor
5  print( floor(2.8) )
6  from math import *
7  print( ceil(2.8) )
8  import sys
9  print( sys.path )
10 print("-- exception --")
11 try:
12   print(1/0)
13   print("-- end --")
14 except ZeroDivisionError as e:
15   print( e )
16 else:
17   print("--other error--")
18 finally:
19   print("--finally--")
```

```
1  -- modul --
2  1
3  2
4  3
5  ['c:\\python36\\python36.zip', 'c:\\python36']
6  -- exception --
7  division by zero
8  --finally--
9
```

Példa 9

Osztályok létrehozása, öröklés

```
1  class osztaly:
2    def fuggveny1(self):
3      return "fuggveny1"
4  print( osztaly.fuggveny1 )
5  print( osztaly().fuggveny1 )
6  print( osztaly().fuggveny1() )
7  class osztaly:
8    def __init__(self, par1 ):
9      self.par1=par1
10   def fuggveny1(self):
11     return "fuggveny1" + self.par1
12
13 print( osztaly("xx").fuggveny1() )
14 print( osztaly("yy").fuggveny1() )
15 class osztaly2(osztaly):
16   def fuggveny2(self):
17     return "fuggveny2" + self.par1
18 print( osztaly2("zz").fuggveny1() )
19 print( osztaly2("zz").fuggveny2() )
```

```
1  <function osztaly.fuggveny1 at 0x0000000002F9B620>
2  <bound method osztaly.fuggveny1 of
   <__main__.osztaly object at 0x0000000002F92A90>>
3  fuggveny1
4  fuggveny1xx
5  fuggveny1yy
6  fuggveny1zz
7  fuggveny2zz
8
```

Példa 10

Láthatóság, aggregate generátor függvény

```python
a=11
def func():
  a=22
  return a
print( func() , a)
def func():
  global a
  a=22
  return a
print( func() , a)
print("-- aggregate --")
def func():
  print("--1--")
  yield 1
  print("--2--")
  yield 2
for i in func():
  print( i, "***" )

```

```
22 11
22 22
-- aggregate --
--1--
1 ***
--2--
2 ***

```

Példa 11

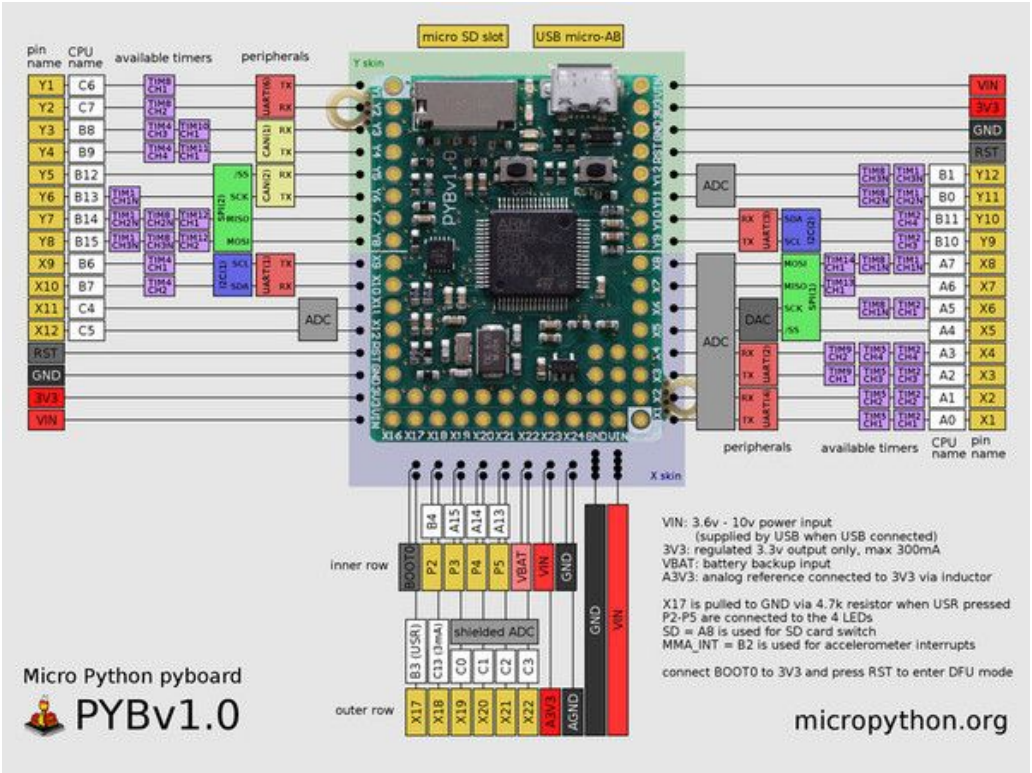
IO műveletek, karakterlánc formázások

```
1  import os
2  print( os.getcwd() )
3  with open("out.txt","w") as f:
4    f.write("szia")
5  print("van" if os.path.isfile("out.txt") else \
6  "nincs" )
7  f=open("out.txt","r")
8  print( f.read() )
9  f.close()
10 class osztaly: pass
11 print( dir(osztaly) )
12 def fakestr(self):
13   return "***osztaly***"
14 print("-- osztály jellemző felülírása --")
15 print( osztaly() )
16 osztaly.__str__=fakestr
17 print( osztaly() )
18 print("str: %s, int: %04d" % ("szia" , 12) )
19 print("str: {0}, int: {1:04d}".format("szia",12))
```

```
1  d:\munkak\előadások\micropython
2  van
3  szia
4  ['__class__', '__delattr__', '__dict__',
   '__dir__', '__doc__', '__eq__', '__format__',
   '__ge__', '__getattribute__', '__gt__',
   '__hash__', '__init__', '__init_subclass__',
   '__le__', '__lt__', '__module__', '__ne__',
   '__new__', '__reduce__', '__reduce_ex__',
   '__repr__', '__setattr__', '__sizeof__',
   '__str__', '__subclasshook__', '__weakref__']
5  -- osztály jellemző felülírása --
6  <__main__.osztaly object at 0x0000000002FA2E48>
7  ***osztaly***
8  str: szia, int: 0012
9  str: szia, int: 0012
10
```

Micropython és a PYboard

A micropython egy mikrovezérlőn futtatható python megvalósítás.
Létezik PyBoard ami a “hivatalos” hardver
STM32F405RG mikrovezérlővel rendelkezik
168MHz ARM M4 1Mb flash, 192Kb RAM
USB és SD kártya foglalat és 3 tengelyes gyorsulásmérő is helyet kapott a kártyán
Számítógépre csatlakoztatva felismeri meghajtóként és egy letölthető a meghajtóról ami a virtuális soros port használatához kell
Ezek után a virtuális soros porton kapcsolódni lehet a futtatóhoz
Érdemes putty programot használni



micropython pyb modul alapjai

a hardver eszközhöz a pyb modulon keresztül lehet hozzáférni (ESP esetén machine a neve)

a kártyán lévő led kezelése:

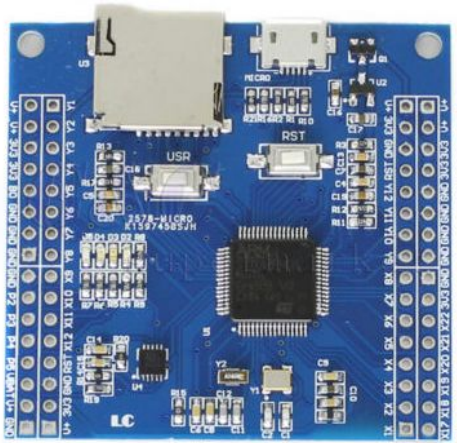
```
from pyb import LED

led = LED(1) # 1=red, 2=green, 3=yellow, 4=blue
led.toggle()
led.on()
led.off()

# LEDs 3 and 4 support PWM intensity (0-255)
LED(4).intensity()    # get intensity
LED(4).intensity(128) # set intensity to half
```

GPIO kezelése:

```
from pyb import Pin

p_out = Pin('X1', Pin.OUT_PP)
p_out.high()
p_out.low()

p_in = Pin('X2', Pin.IN, Pin.PULL_UP)
p_in.value() # get value, 0 or 1
```

Gyorsulásmérő lekérdezése:

```
from pyb import Accel

accel = Accel()
print(accel.x(), accel.y(), accel.z(), accel.tilt())
```



micropython pyb modul i2c példa

HTU21D modul, I2C hőmérséklet és páratartalom mérő module
Hőmérséklet 14 bites felbontás 0.01°C

```
>>> from pyb import I2C
>>> i2c=I2C(2, I2C.MASTER)
>>> i2c.scan()
[64]
```


```
from pyb import I2C

i2c = I2C(1, I2C.MASTER, baudrate=100000)
i2c.scan() # returns list of slave addresses
i2c.send('hello', 0x42) # send 5 bytes to slave with address 0x42
i2c.recv(5, 0x42) # receive 5 bytes from slave
i2c.mem_read(2, 0x42, 0x10) # read 2 bytes from slave 0x42, slave memory 0x10
i2c.mem_write('xy', 0x42, 0x10) # write 2 bytes to slave 0x42, slave memory 0x10
```