



```
def :  
    return x+5
```

```
def dotwrite(ast):
```

```
    nodename = getNodename()
```

```
    label=symbol.sym_name.get(int(ast[0]), ast[0])
```

```
    print '    %s [%s] (%s)' % (ast[0], label, ast[1])
```

```
    if isinstance(ast[1], str):
```

```
        if ast[1].strip():
```

```
            print '= %s';' % ast[1]
```

```
        else:
```

```
            print ''
```

```
    else:
```

```
        print '";'
```

```
        children = []
```

```
        for n, child in enumerate(ast[1:]):
```

```
            children.append(dotwrite(child))
```

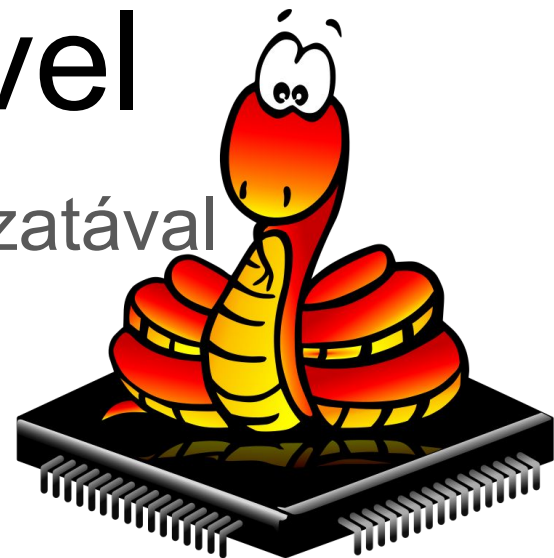
```
        print ', %s -> {' % nodename
```

```
        for name, child in zip(children):
```

```
            print '%s' % name,
```

# Ismerkedés a Python programnyelvvvel

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 szemétygyűjtő algoritmus
- objektum orientált nyelv, támogatja a többszörös öröklést
- bővíthető, könnyen illeszthető meglévő C könyvtárakhoz
- csomagkezelővel rendelkezik (pip install)
- ingyenes



# Python beszerzése, futtatása PC-n

Letölthető, oprendszer és géptípus választás után

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

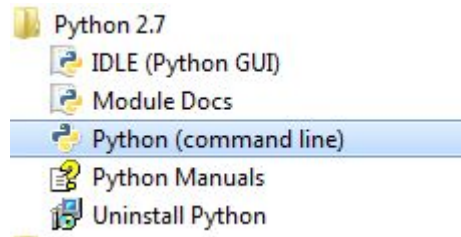
érdeemes MSI installer választani windows alatt  
Linux alatt a disztribúció csomagkezelőjével

Telepítése:

MSI esetén a tovább (next) gomb nyomogatása

Futtatás:

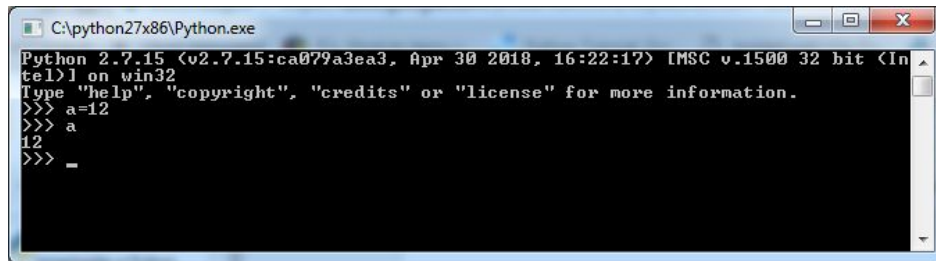
cmd ablakban python szó beírásával,  
vagy Start menü Python (command line) kiválasztása



# Első python program futtatása

## Interaktív REPL (read-eval-print loop)

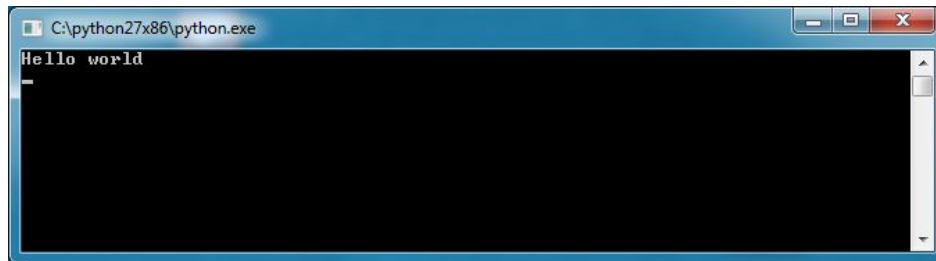
A változó értékét a változó beírásával adja vissza



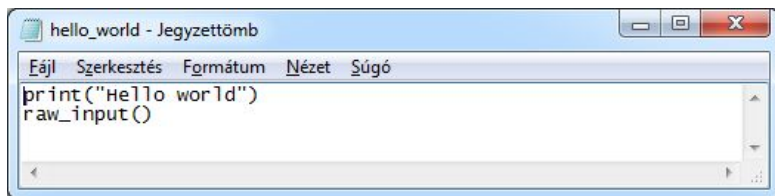
```
C:\python27x86\Python.exe
Python 2.7.15 (v2.7.15:ca079a3ea3, Apr 30 2018, 16:22:17) [MSC v.1500 32 bit <In
tel>] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> a=12
>>> a
12
>>> -
```

## Script mód

A megírt script futtatása cmd ablakban

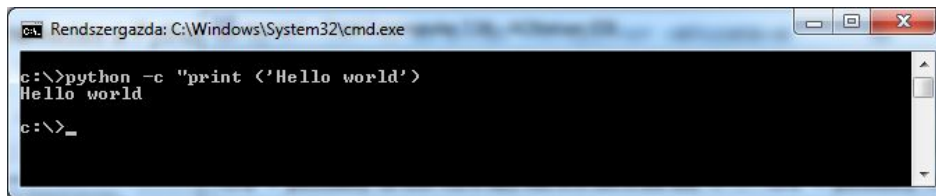


```
C:\python27x86\python.exe
Hello world
-
```



```
hello_world - Jegyzetömb
Fájl Szerkesztés Formátum Nézet Súgó
print("Hello world")
raw_input()
```

Parancssori paraméterek, pl. -c



```
C:\Rendszergazda: C:\Windows\System32\cmd.exe
c:\>python -c 'print (<'Hello world<')
Hello world
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 kezdődjen, a tabulátor nem helyettesíthető szóközzel.

Érdemes a szövegszerkesztőt úgy beállítani, hogy a begépett tabulátorokat x szóközre cserélje automatikusan.

fejlsor:

```
    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.

A változók elnevezése a foglalt szavakon kívüli szavakkal lehetséges, kis és nagy betű különbözőnek 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.

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

# Python példák a böngészőben

a `pyserv.py` program elindítása után írjuk a böngésző címsorába: <http://localhost:8000>

Python futtató

Példa 1 Példa 2 Példa 3 Példa 4 Példa 5 Példa 6 Példa 7 Példa 8 Példa 9 Példa 10 Példa 11 Példa 12

```
1 a=12
2 print(a)
3 b=c=22
4 print( b, c )
5 d,e=33,34
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 ) )
18
```

Futtat





## 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),0xa)
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
0x000000003015C88> 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

```
1 print("-- dict --")
2 print( {'a': 1 , 'b': 2} )
3 a={'a': 1 , 'b': 2} ; print( a['a'] )
4 print( a.keys(), a.values() )
5 a['c']=3 ; print( a )
6 del a['c']; print( a )
7 a.update({'d' : 4}) ; print( a )
8 print( *a )
9 c={**a,**{'e':5}} ; print( c )
10 print( dict([( 'a',1) , ('b',2) , ('c',3) ]) )
11 print( list(enumerate(a)) )
12 print( 'd' in a , 'z' in a )
13 print( a.get('d') , a.get('z') )
14 print( a, len( a ) )
15 print("-- szótárértelmezés --")
16 print( {k: v*2 for k,v in a.items()} )
```

```
1 -- dict --
2 {'a': 1, 'b': 2}
3 1
4 dict_keys(['a', 'b']) dict_values([1, 2])
5 {'a': 1, 'b': 2, 'c': 3}
6 {'a': 1, 'b': 2}
7 {'a': 1, 'b': 2, 'd': 4}
8 a b d
9 {'a': 1, 'b': 2, 'd': 4, 'e': 5}
10 {'a': 1, 'b': 2, 'c': 3}
11 [(0, 'a'), (1, 'b'), (2, 'd')]
12 True False
13 4 None
14 {'a': 1, 'b': 2, 'd': 4} 3
15 -- szótárértelmezés --
16 {'a': 2, 'b': 4, 'd': 8}
17
```

## String, karakterlánc

```
1 print("-- string --")
2 print("szia")
3 print('szia')
4 print("""szia""", "'szia'", "\"szia\"")
5 print("""szia
6 python""")
7 print("szia" + " python")
8 print("szia" " python")
9 print("szia\tpython")
10 print("szia"[-1])
11 print("szia"[1:])
12 print("szia"[5:])
13 print( len("szia") )
14 print(list("szia") )
15 print(set("szeptember") )
16 print("szia python".split(" "))
17 print("szia python".replace("s","z"))
18 print("szia python".upper())
19
```

```
1 -- string --
2 szia
3 szia
4 "szia" 'szia' "szia"
5 szia
6 python
7 szia python
8 szia python
9 szia    python
10 a
11 zia
12
13 4
14 ['s', 'z', 'i', 'a']
15 {'p', 's', 'e', 'z', 't', 'm', 'r', 'b'}
16 ['szia', 'python']
17 zzia python
18 SZIA PYTHON
19
```

## 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
```



## 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
```

## 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
3 <__main__.osztaly object at 0x0000000002F92A90>>
4 fuggveny1
5 fuggveny1xx
6 fuggveny1yy
7 fuggveny1zz
8 fuggveny2zz
```

## Láthatóság, aggregate generátor függvény

```
1 a=11
2 def func():
3     a=22
4     return a
5 print( func() , a)
6 def func():
7     global a
8     a=22
9     return a
10 print( func() , a)
11 print("-- aggregate --")
12 def func():
13     print("--1--")
14     yield 1
15     print("--2--")
16     yield 2
17 for i in func():
18     print( i, "***" )
19
```

```
1 22 11
2 22 22
3 -- aggregate --
4 --1--
5 1 ***
6 --2--
7 2 ***
8
```

## 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))
```

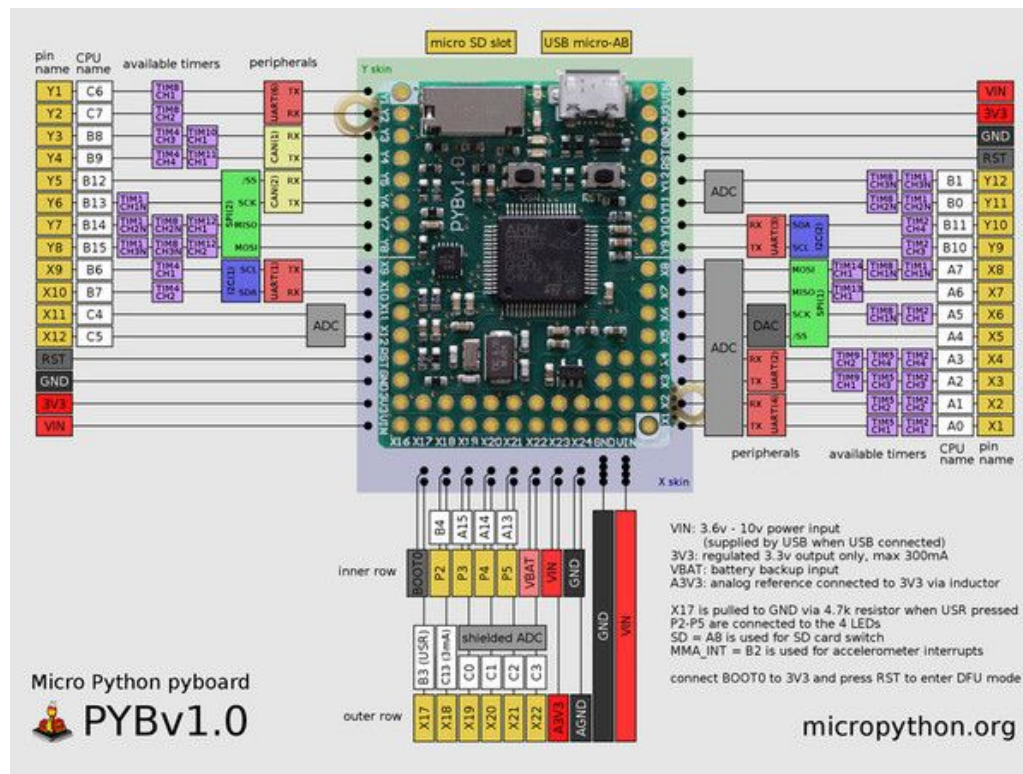
```
1 d:\munkak\előadások\micropython
2 van
3 szia
4 ['__class__', '__delattr__', '__dict__',
5  '__dir__', '__doc__', '__eq__', '__format__',
6  '__ge__', '__getattr__', '__gt__',
7  '__hash__', '__init__', '__init_subclass__',
8  '__le__', '__lt__', '__module__', '__ne__',
9  '__new__', '__reduce__', '__reduce_ex__',
10 '__repr__', '__setattr__', '__sizeof__',
11 '__str__', '__subclasshook__', '__weakref__']
12 -- osztály jellemző felülírása --
13 <_main_.osztaly object at 0x000000002FA2E48>
14 ***osztaly***
15 str: szia, int: 0012
16 str: szia, int: 0012
17
```

# Micropython és a PYboard

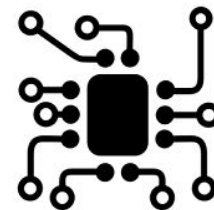
A micropython egy mikrovezérlőn futtatható python megvalósítás.

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 USB meghajtóként és egy driver letölthető a kártyán

Ezek után a virtuális soros porton kapcsolódni lehet a futtatóhoz (REPL).  
Érdeemes putty programot használni



# micropython pyb modul alapjai, GPIO



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

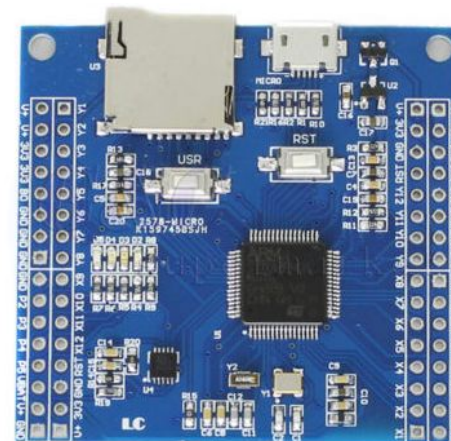
p_out = Pin('X1', Pin.OUT_PP)
p_out.high()
p_out.low()

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

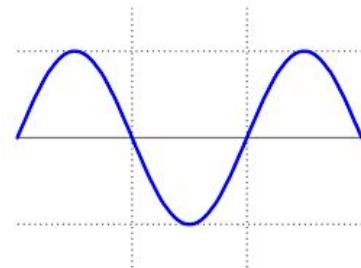
Gyorsulásmérő lekérdezése:

```
from pyb import Accel

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



# micropython pyb modul alapjai, DAC és ADC



DAC kezelése:

```
from pyb import DAC

dac = DAC(1)           # create DAC 1 on pin X5
dac.write(128)        # write a value to the DAC (makes X5 1.65V)

dac = DAC(1, bits=12) # use 12 bit resolution
dac.write(4095)       # output maximum value, 3.3V
```

Szinusz hullám generálása:

```
import math
from pyb import DAC

# create a buffer containing a sine-wave
buf = bytearray(100)
for i in range(len(buf)):
    buf[i] = 128 + int(127 * math.sin(2 * math.pi * i / len(buf)))

# output the sine-wave at 400Hz
dac = DAC(1)
dac.write_timed(buf, 400 * len(buf), mode=DAC.CIRCULAR)
```

ADC:

```
adc = pyb.ADC(pin)           # create an analog object from a pin
val = adc.read()             # read an analog value

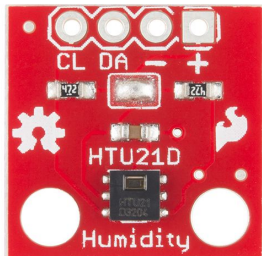
adc = pyb.ADCAll(resolution)  # create an ADCAll object
adc = pyb.ADCAll(resolution, mask) # create an ADCAll object for selected analog channels
val = adc.read_channel(channel) # read the given channel
val = adc.read_core_temp()    # read MCU temperature
val = adc.read_core_vbat()    # read MCU VBAT
```

## 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]
```



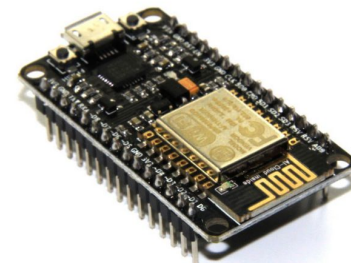
```
from pyb import I2C
```

```
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
```

```
COM18 - PuTTY
MicroPython v1.8.7-478-gbfb48c1 on 2017-03-24; PYBv1.1 with STM32F405RG
Type "help()" for more information.
>>> from pyb import I2C
>>> import time
>>>
paste mode; Ctrl-C to cancel, Ctrl-D to finish
=== def temp():
===     i2c = I2C(2, I2C.MASTER)
===     i2c.send(0xE3,0x40)
===     time.sleep_ms(50)
===     b=i2c.recv(3,0x40)
===     x=b[0]<<8 | b[1] & 0b11111100
===     t=x * 175.72 / 65536 - 46.85
===     return t
===
>>> temp()
23.57099
>>>
```

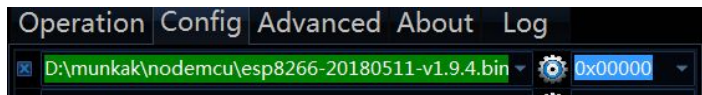


# Micropython telepítése ESP8266 -ra



Legegyszerűbben az ESP8266Flasher.exe -vel lehet feltölteni az ESP8266 Dev board-ra, a micropython bin állományt.

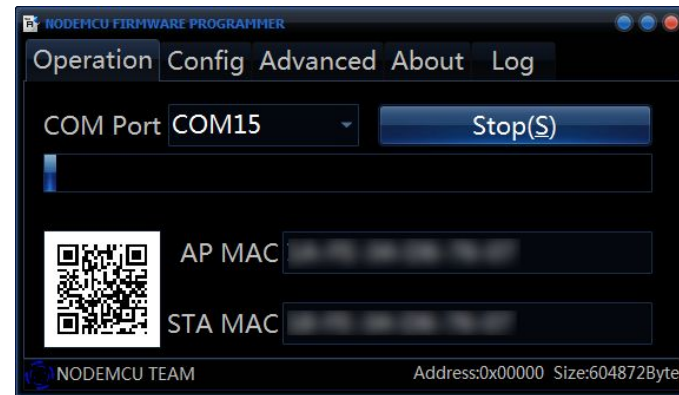
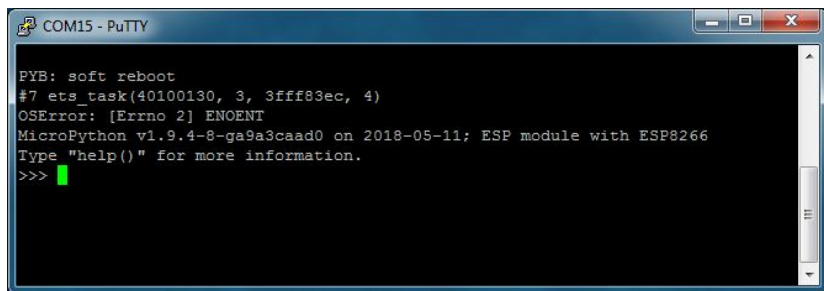
COM port kiválasztása után a Config menüben kell kitallózni a letöltött bin állományt



, majd az Operation menüben a Flash gombra kattintva elindul a feltöltés és a Flash gomb Stop-ra változik amíg tart.

Bin állomány letölthető a <http://micropython.org/download> , Firmware for ESP8266 boards szekcióból

A sikeres feltöltés és reset után terminálban megnyitva a COM portot 115200 -as baud rate-el, elérjük a REPL-t



## ESP8266 spesifikus python modulok, network

```
1 import network
2
3 # Client mode
4 sta_if=network.WLAN(network.STA_IF)
5 print(sta_if.active()) #1
6 sta_if.connect("ssid","password")
7 print( sta_if.isconnected() ) #2
8
9 # AP mode
10 ap_if = network.WLAN(network.AP_IF)
11 print(ap_if.active(True)) #3
12 print(ap_if.ifconfig()) #4
13
```

```
1 True
2 True
3 True
4 ('192.168.4.1', '255.255.255.0', '192.168.4.1',
5  '213.46.246.53')
```

## ESP8266 specifikus python modulok, DHT

```
1 import machine
2 import dht
3 d = dht.DHT11(machine.Pin(14))
4 d.measure()
5 print( d.temperature() ) #1
6
7 import time
8 import urequests
9 while True:
10     d.measure()
11     urequests.post(\
12         "http://192.168.0.13:8000/temp"\
13         ,data=str(d.temperature()))
14     time.sleep(30)
15
```

```
1 22
2
```

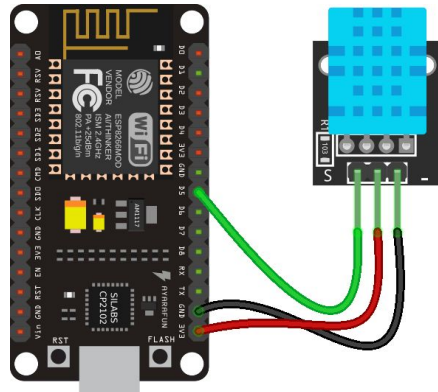
# Beküldött hőmérséklet kijelzése

```
C:\python27x86\python.exe
Server Starts - 0.0.0.0:8000
192.168.0.13 - - [23/Oct/2018 16:09:58] "POST /temp HTTP/1.1" 200 -
temp : 22
192.168.0.28 - - [23/Oct/2018 16:14:02] "POST /temp HTTP/1.0" 200 -
temp : 22
192.168.0.28 - - [23/Oct/2018 16:14:39] "POST /temp HTTP/1.0" 200 -
temp : 22
192.168.0.28 - - [23/Oct/2018 16:15:09] "POST /temp HTTP/1.0" 200 -
temp : 22
192.168.0.28 - - [23/Oct/2018 16:15:40] "POST /temp HTTP/1.0" 200 -
temp : 22
192.168.0.28 - - [23/Oct/2018 16:16:10] "POST /temp HTTP/1.0" 200 -
temp : 22
192.168.0.28 - - [23/Oct/2018 16:16:40] "POST /temp HTTP/1.0" 200 -
temp : 22
192.168.0.28 - - [23/Oct/2018 16:17:10] "POST /temp HTTP/1.0" 200 -
temp : 22
-
```

```
import sys, os
if (sys.version_info > (3, 0)):
    from http.server import BaseHTTPRequestHandler, HTTPServer
else:
    from BaseHTTPServer import BaseHTTPRequestHandler, HTTPServer

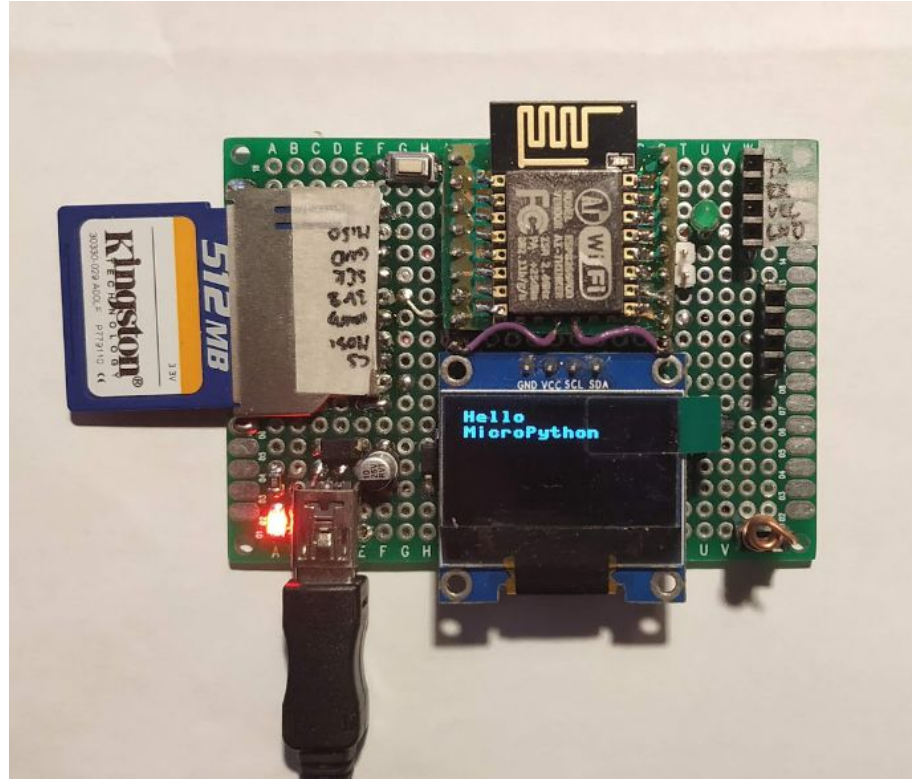
class RequestHandler(BaseHTTPRequestHandler):
    def do_POST(self):
        if self.path=="/temp":
            content_length = int(self.headers['Content-Length'])
            post_data = self.rfile.read(content_length)
            post_data = post_data.decode("utf-8")
            self.send_response(200)
            self.send_header("Content-type", "text/plain")
            self.end_headers()
            print( "temp : " + post_data )
            resp="ok"
            if (sys.version_info > (3, 0)):
                self.wfile.write( bytes( resp , "utf-8" ) )
            else:
                self.wfile.write( resp )

if __name__ == '__main__':
    serverAddress = ("0.0.0.0", 8000 )
    server = HTTPServer(serverAddress, RequestHandler)
    server.serve_forever()
```



## OLED vezérlés

```
1 import machine
2 import ssd1306
3
4 i2c = machine.I2C( \
5     sda=machine.Pin(5), scl = machine.Pin(16) )
6 oled = ssd1306.SSD1306_I2C(128, 64, i2c)
7 oled.text('Hello', 0, 0)
8 oled.text('MicroPython', 0, 10)
9 oled.show()
10
```



## SD kártya kezelése és file IO

```
1 import machine
2 import sdcard
3 import os
4 sd = sdcard.SDCard(machine.SPI(1), machine.Pin(2))
5 os.mount(sd, '/sd')
6 print( os.listdir("/") ) #1
7 print( os.listdir("/sd") ) #2
8 # file IO
9 f = open('data.txt', 'w')
10 f.write('1234')
11 f.close()
12
13 f = open('data.txt')
14 print( f.read() )
15 f.close()
16
```

```
1 ['sd', 'boot.py', 'sdcard.py', 'ssd1306.py']
2 ['NIKON001.DSC', 'DCIM']
3 1234
4
```

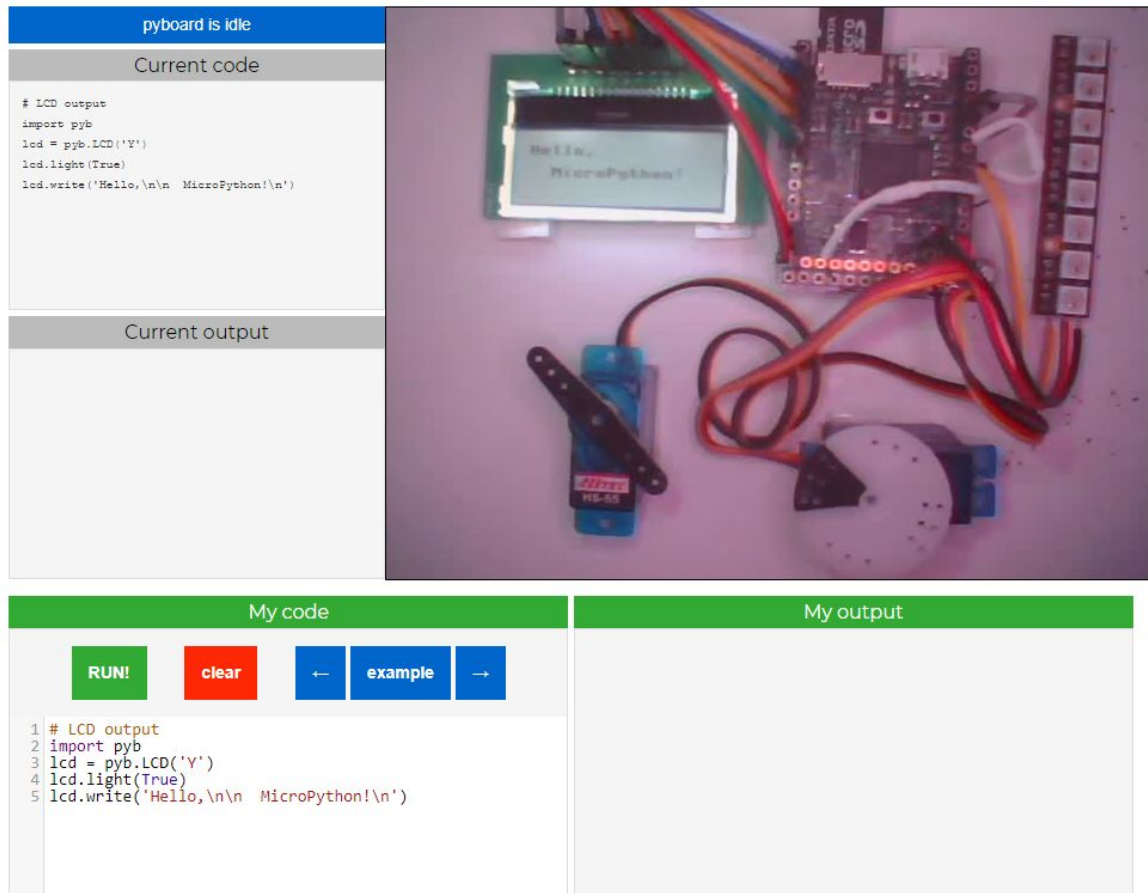
# MicroPython Live

Érdekes lehetőség, hogy egy pyboard bekamerázva elérhető és futtatható rajta tetszőleges program és vezérelhető a hozzá kapcsolódó periféria is.

Az oldal elérhető

<http://micropython.org/live/>

Az example gomb melletti nyilakkal előre megírt tesztprogramok lapozgathatóak, de tetszőleges program is írható vagy a meglévő módosítható futtatás előtt. Futtatni a RUN! gombbal lehetséges a kódot.



The image displays the MicroPython Live interface, which is split into several sections. On the left, there is a code editor with a blue header 'pyboard is idle'. Below it, the 'Current code' section shows the following Python code:

```
# LCD output
import pyb
lcd = pyb.LCD('Y')
lcd.light(True)
lcd.write('Hello,\n\n MicroPython!\n')
```

Below the code is the 'Current output' section, which is currently empty. To the right of the code editor is a photograph of a Pyboard microcontroller board. The board is connected to a green LCD screen displaying 'Hello, MicroPython!'. Other peripherals connected include a blue USB cable, a white fan, and a breadboard with several components. Below the photograph is a green header 'My code'. Underneath, there are five buttons: a green 'RUN!' button, a red 'clear' button, a blue left arrow button, a blue 'example' button, and a blue right arrow button. Below these buttons is a code editor with the same code as above, numbered 1 to 5. To the right of the code editor is a green header 'My output', which is currently empty.