



# Python

Ռսկան Ռսկանյան

# Ծրագիր | Մեքենայական կոդ

00110001	00000000	00000000
00110001	00000001	00000001
00110011	00000001	00000010
01010001	00001011	00000010
00100010	00000010	00001000
01000011	00000001	00000000
01000001	00000001	00000001
00010000	00000010	00000000
01100010	00000000	00000000

# Assembly language

```
.MODEL SMALL
```

```
.DATA
```

```
    msg DB 'Hello World',13,10,'$'
```

```
.CODE
```

```
START:
```

```
    mov ax, @DATA
```

```
    mov ds, ax
```

```
    mov ax, 0900h
```

```
    lea dx, msg
```

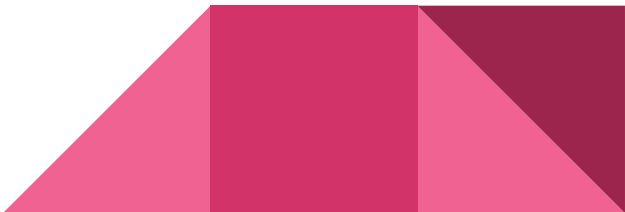
```
    int 21h
```

```
    mov ax, 4C00h
```

```
    int 21h
```

```
END START
```

Wikipedia: [Assembly language](#)



# Մեքենայական կոդի վերածում

- Compiler
- Interpreter



# Ծրագրավորման լեզուներ

- C / C++
- Go
- Java
- Python
- C#
- Ruby
  
- JavaScript
- PHP
- Objective-C
  
- Prolog
- Lisp
- Erlang

[Список языков программирования по категориям](#)



# Python

Google

You Tube



IBM



Яндекс



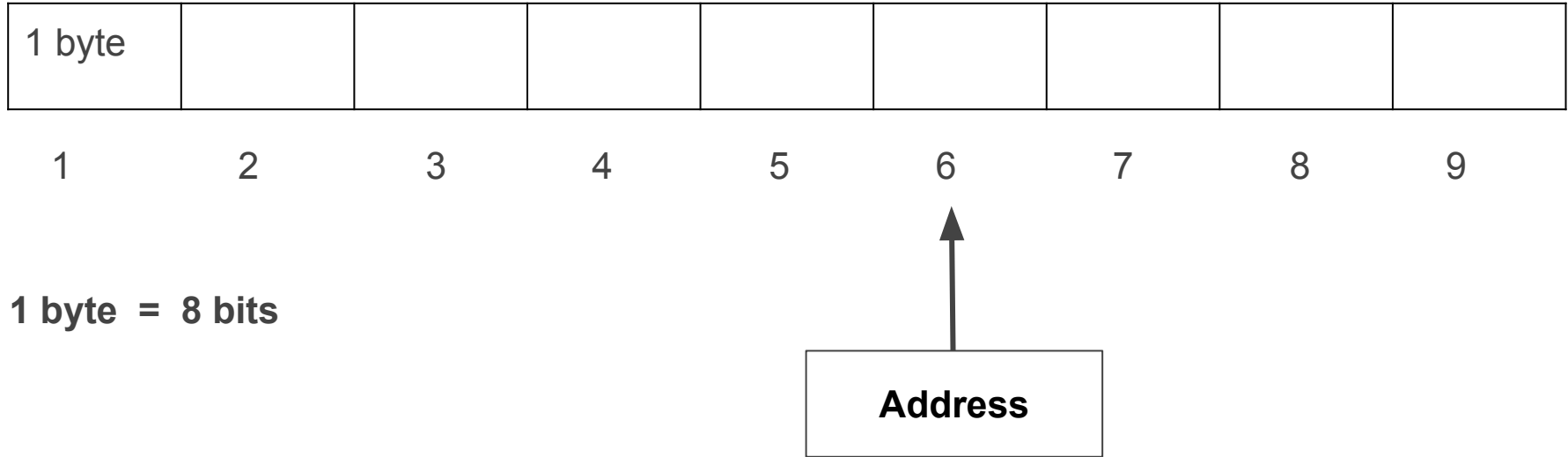
Seagate 

iRobot

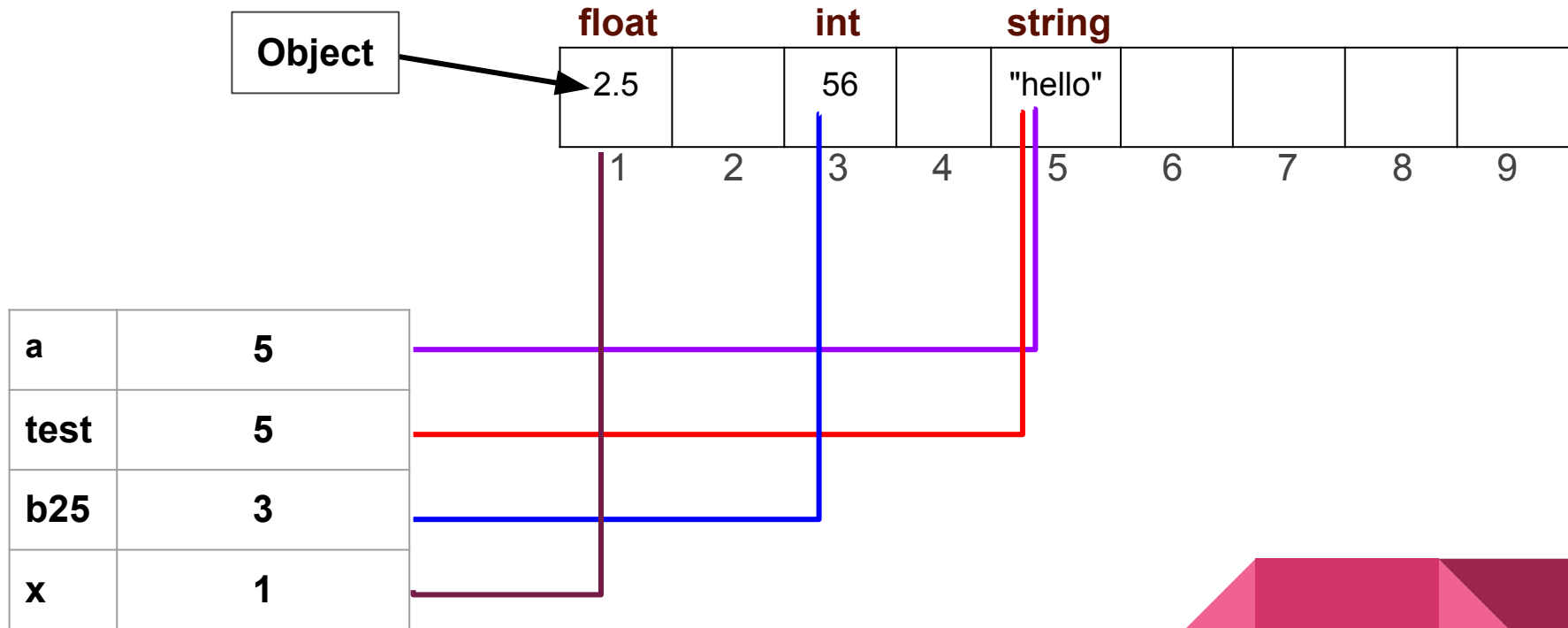
  
CISCO

 Dropbox

# Համակարգչի հիշողություն



# Փոփոխականներ





# Փոփոխականներ

```
# int
```

```
i = 23
```

```
# float
```

```
f = 1.5
```

```
# string
```

```
s = "hello"
```

```
print(i, f, s)
```



# Տիպերի փոփոխություն

```
i = int("23")
```

```
f = float("45")
```

```
s = str(10)
```

```
b = bool("hi")
```

```
print(i, f, s, b)
```



# Օպերատորներ

*# Մաթեմատիկական գործողություններ (օպերատորներ)*

```
print(2 * 3)
```

```
print(2 - 3)
```

```
print(2 + 3)
```

```
print(3 / 2)
```

```
print(3 // 2)
```

*# Մաթեմատիկական օպերատորները տողերի հետ*

```
print("Hello" * 5)
```

```
print("Hello" + "World")
```

*# Տողի սիմվոլների քանակի ստացում*

```
print(len("Hello")) # result - 5
```



# String

```
s1 = 'Lorem Ipsum is simply dummy text of the printing and typesetting industry.\nLorem Ipsum has been the industry\'s standard dummy text ever since the 1500s...'
```

```
s2 = "Lorem Ipsum is simply dummy text of the printing and typesetting industry.\nLorem Ipsum has been the industry's standard dummy text ever since the 1500s..."
```

```
s3 = '''Lorem Ipsum is simply dummy text of the printing and typesetting industry.\nLorem Ipsum has been the industry's standard dummy text ever since the 1500s...'''
```

```
s4 = """Lorem Ipsum is simply dummy text of the printing and typesetting industry.\nLorem Ipsum has been the industry's standard dummy text ever since the 1500s..."""
```

```
print(s1)
```

```
print(s2)
```

```
print(s3)
```

```
print(s4)
```



# Տողերի Ֆորմատավորում

```
name = "John"
```

```
age = 15
```

```
s0 = "My name is " + name + "My age is " + str(age)
```

```
s1 = "My name is {}. My age is {}".format(name, age)
```

```
s2 = f"My name is {name}. My age is {age}"
```

```
print(s0)
```

```
print(s1)
```

```
print(s2)
```



# String - lower, upper

```
name = "JOHN"
```

```
s1 = name.lower()
```

```
print(s1)    # result - john
```

```
s2 = s1.upper()
```

```
print(s2)    # result - JOHN
```



# Ինչպես օգտվել help-ից

```
help(len)
```

```
help(str)
```

```
help(str.lower)
```



# Մոդուլներ - random

*# Գրադարանի / մոդուլի միացում*

```
import random
```

*# գեներացնում է 0-ից 4 պատահական թվեր*

```
r1 = random.randrange(5)
```

*# գեներացնում է 2-ից 9 պատահական թվեր*

```
r2 = random.randrange(2, 10)
```

```
print(r2)
```





# Գործնական ածխատանք

Գրե՛ք ծրագիր, որը գեներացնում է երկու պատահական թվեր՝ 10-ից մինչև 30, ցուցադրում այդ երկու թվերը օգտվողին:

Օգտվողը պետք է մուտքագրի այդ երկու թվերի գումարը, եթե ճիշտ է մուտքագրված, տպում ենք՝ Ճիշտ է, հակառակ դեպքում՝ Սխալ է:



# Գործնական ածխատանքի պատասխանը

```
import random
```

```
num1 = random.randrange(10, 30)
```

```
num2 = random.randrange(10, 30)
```

```
answer = input(f"Մուտքագրե՛ք '{num1} + {num2}' գործողության պատասխանը: ")
```

```
answer = int(answer)
```

```
if answer == num1 + num2:
```

```
    print("Ճիշտ է")
```

```
else:
```

```
    print("Միտալ է")
```



# `str.isdigit()`

*# եթե տողում նշված է թիվ, ապա վերադարձնում է True, հակառակ դեպքում՝ False*

```
print('23'.isdigit())    # True
```

```
print('h'.isdigit())     # False
```



# str.replace()

*# Առաջին պարամետրով նշվում է, թե տողում որ սիմվոլը պետք է փոխվի*

*# Երկրորդ պարամետրով, թե ինչ սիմվոլով պետք է փոխվի*

*# Երրորդով, թե քանի անգամ պետք է փոխվի*

```
print('23.65689'.replace('.', ''))
```

```
print('23.656.89'.replace('.', ''))
```

```
print('23.656.89'.replace('.', '', 1))
```



# Оператор not

```
print(not True)
print(not False)
print(not 0)
print(not 34)
print(not '')
print(not 'hello')
```



# Օպերատոր pass

```
n = 11
```

```
# Նրե դեռ չենք որոշել if-ում, elif-ում կամ else-ում
```

```
# ինչ գործողություն պետք է անենք, կարող ենք բաց թողնել
```

```
# օգտվելով pass օպերատորից
```

```
if n > 10:
```

```
    pass
```

```
else:
```

```
    print("Hello")
```



# Գործնական աշխատանք

Գրե՛ք ծրագիր, որը զենեքացնում է 1-ից 15 պատահական թվեր.

Օգտվողը պետք է գուշակի այդ թիվը, եթե ճիշտ է, տպում ենք որ ճիշտ է:

Եթե մեծ է կամ փոքր, տպում ենք մեծ է կամ փոքր և ասում որ կրկին մուֆագրի: Ե՛վ այդպես այնքան մինչև ճիշտ գուշակի:



# while ghqL

```
int i = 0;  
while i < 5:  
    print(i)  
    i += 1
```

```
while True:  
    break  
    continue
```





# for in

```
for i in range(2, 7):  
    print(i)
```

```
for i in range(7):  
    print(i)
```



# Ցուցակներ - Llist

```
l1 = [3, 6, 9, 1]
```

```
l2 = ["hello", 45, True, "test"]
```

```
l3 = [34, "test", [4, 5, ["hello"]], 6, False, "hello"]
```

```
print(l1)
```

```
print(l2)
```

```
print(l1[1])
```

```
print(l2[2])
```



# Ցուցակներ - Llist

```
myList = ["hello", 45, True, "test"]
```

```
for i in myList:  
    print(i)
```



# Llist - append

```
myList = ["hello", 45, True, "test"]
```

*# Ավելացնում է ցուցակի վերջում*

```
myList.append(45)
```

```
print(myList)
```



# Llist

```
r = range(2, 9)
```

```
myList = list(r)
```

```
print(myList)
```



# len()

```
print(len([3, 4, 6, 9]))
```

```
print(len(range(1, 9)))
```

```
print(len("hello"))
```



# list

```
s = 'hello'
```

```
# տողը քերում ենք list տեսակի
```

```
print(list(s))
```



# list - pop

```
l = [2, 6, 'hello', 78]
```

```
# list-ից վերցնում է վերջին էլեմենտը
```

```
e1 = l.pop()
```

```
# list-ից վերցնում է ըստ ինդեքսի
```

```
e1 = l.pop(3)
```

```
print(e1)
```

```
print(l)
```





## list - sort

```
l = [34, 6, 7, 1, 12]
```

```
# list-ը սորտավորում է ըստ անման կարգի
```

```
l.sort()
```

```
# list-ը սորտավորում է ըստ նվազման կարգի
```

```
l.sort(reverse=True)
```

```
print(l)
```



# list - sort

```
l = ['webstorm', 'pycharm', 'vscode', 'clion']
```

```
# տողերի սորտավորում
```

```
l.sort()
```

```
print(l)
```



## list - sort

```
def myFunc(e):  
    return len(e)
```

```
l = ['webstorm', 'pycharm', 'vscode', 'clion']
```

```
l.sort(key=myFunc, reverse=True)
```

```
print(l)
```



# list

```
l1 = [34, 67, 8]
```

```
# l2 -ը հղվում է l1-ի հիշողության տիրույթ
```

```
l2 = l1
```

```
l2.pop(1)
```

```
print(l1, l2)
```


```
a1 = [34, 67, 8]
```

```
# ստեղծվում է նոր list a1-ի կրկնօրինակի հիման վրա
```

```
a2 = a1.copy()
```

```
a2.pop(1)
```

```
print(a1, a2)
```



# list

```
l = [34, 67, 8, 10, 5, 3, 77]
```

```
# Վերադարձնում է 3 ինդեքսով էլեմենտը
```

```
print(l[3])
```

```
# Վերադարձնում է սկսած 4-րդ ինդեքսից մինչև վերջի էլեմենտը
```

```
print(l[4:])
```

```
# Վերադարձնում է սկսած մինչև 5-րդ ինդեքսով էլեմենտները
```

```
print(l[:5])
```

```
# Վերադարձնում է 2-ից սկսած մինչև 4-ը
```

```
print(l[2:4])
```

```
# Վերադարձնում է բոլորը
```

```
print(l[:])
```



# list

```
l = 'Hello, John'
```

```
# Վերադարձնում է 3 ինդեքսով էլեմենտը
```

```
print(l[3])
```

```
# Վերադարձնում է սկսած 4-րդ ինդեքսից մինչև վերջի էլեմենտը
```

```
print(l[4:])
```

```
# Վերադարձնում է սկսած մինչև 5-րդ ինդեքսով էլեմենտները
```

```
print(l[:5])
```

```
# Վերադարձնում է 2-ից սկսած մինչև 4-ը
```

```
print(l[2:4])
```

```
# Վերադարձնում է բոլորը
```

```
print(l[:])
```



# list - in operator

```
l1 = 'Hello, John'
```

```
l2 = [34, 67, 8, 10, 5, 3, 77]
```

```
# in օպերատորով կարող ենք ստուգել
```

```
# ցուցակում կա այդ էլեմենտը, թե ոչ
```

```
# վերադարձնում է True կամ False
```

```
print("H" in l1)
```

```
print(67 in l2)
```

```
print(67 not in l2)
```



# Գործնական աշխատանք

`l1 = [2, 4, 5, 1, 6, 11]`

`l2 = []`

`l1 == l1`

**Ունենք l1 list, այդ ցուցակի հիման վրա ստեղծեք l2 ցուցակում,  
որում պահվելու են l1-ի էլեմենտների քառակուսիները**





# Գործնական աշխատանք

```
l1 = [2, 4, 5, 1, 6, 11]
```

```
l2 = []
```

```
# տարբերակ 1
```

```
for el in l1:
```

```
    l2.append(el ** 2)
```

```
# տարբերակ 2
```

```
l2 = [el ** 2 for el in l1]
```

```
print(l2)
```



# Գործնական աշխատանք

```
l1 = [2, 4, 5, 1, 6, 11]
```

```
l2 = []
```

```
# տարրերի 1, պայմանով
```

```
for e1 in l1:
```

```
    if e1 > 2:
```

```
        l2.append(e1 ** 2)
```

```
# տարրերի 2, պայմանով
```

```
l2 = [e1 ** 2 for e1 in l1 if e1 > 2]
```

```
print(l2)
```



# Tuple

```
t = (3, 6, 9)
```

```
# tuple - քի տարբերություն list-ի չի կարող փոփոխվել
```

```
print(type(t))
```

```
print(t[1])
```

```
print(len(t))
```



# Sets


*# set-ում բոլոր կրկնվող տվյալները ջնջվում են*

```
s1 = {2, 3, 6, 2, 7, 6, 2}
```

```
s2 = {"hello", "test", "hello"}
```

```
s3 = {"hello", 23, "hello", 23, True}
```

```
print(s1, s2, s3)
```



# Գործնական աշխատանք

Ունենք հետևյալ ցուցակը՝

l = [2, 5, 2, 8, 5, 3, 2]

այդ ցուցակից վերցրեք/ստացեք միայն ունիկալ արժեքները



# Dictionaries

```
d = {"a": 300, "b": "hello", "c": True}
```

```
print(d['a'], d['b'], d['c'])
```

```
d['b'] = "Ola"
```

```
print(d['b'])
```

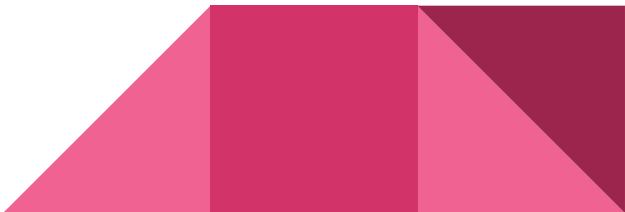


# Dictionaries

```
user = {  
    "name": "John",  
    "age": 23,  
    "height": 1.7,  
    "isAdmin": True  
}
```

```
print(user["name"])
```

```
user["newKey"] = "newValue"
```




# Dictionaries - in operator

```
# in օպերատորով ստուգում ենք,  
# մեզ անհրաժեշտ key-ը կա dictionary-ում, թե ոչ  
user = { "name": "John", "age": 23 }
```

```
# եթե կա, ապա վերադարձնում է True  
print("name" in user)
```

```
if "name" in user:  
    print("name անունով key կա")  
else:  
    print("չկա")
```





## Dictionaries - get()

```
user = {"name": "John", "age": 23}
```

```
# get մեթոդը վերադարձնում է ըստ key-ի արժեքը,
```

```
# իսկ եթե նման key չկա, ապա երկրորդ պարամետրով տրված default արժեքը
```

```
data1 = user.get("name", "Guest")
```

```
data2 = user.get("hello", "Guest")
```

```
print(data1, data2)
```



## Dictionaries - items()

```
user = {"name": "John", "age": 23, "height": 1.7}
```

```
# items մեթոդը վերադարձնում է dictionary-ի list-ի տեսքով,
```

```
# որում key: value-ի tuple է
```

```
print(user.items())
```

```
for key, val in user.items():
```

```
    print(key)
```

```
    print(val)
```



## Dictionaries - values(), keys()

```
user = {"name": "John", "age": 23, "height": 1.7}
```

```
# Վերադարձնում է dictionary-ի բոլոր value-ները list-ի տեսքով
```

```
print(user.values())
```

```
# Վերադարձնում է dictionary-ի բոլոր key-երը list-ի տեսքով
```

```
print(user.keys())
```



# Գործնական աշխատանք

Ունենք հետևյալ ցուցակը՝

```
hello = {  
    "hy": "Բարև",  
    "en": "Hello",  
    "ru": "Привет",  
    "default": "Unknown language code"  
}
```

Օգտվողը պետք է մուտքագրի լեզվի կոդը (hy, ru, en...)

```
lang_code = input("Input language code: ")
```

և եթե ճիշտ է մուտքագրել, պետք է ցուցադրենք կոդին համապատասխան ողջույնը, եթե սխալ՝ default-ի արժեքը



# Function

```
def my_function_name():  
    # function body  
    pass
```

```
# function call  
my_function_name()
```



# Function

```
def print_seconds_per_day():  
    hours = 24  
    minutes = hours * 60  
    seconds = minutes * 60  
    print(seconds)
```

```
print_seconds_per_day()
```



# Function arguments / parameters

*# days - function parameter*

```
def print_seconds_per_day(days):
```

```
    hours = days * 24
```

```
    minutes = hours * 60
```

```
    seconds = minutes * 60
```

```
    print(seconds)
```

```
print_seconds_per_day(3)
```



# Function default argument value

*# function parameter default value*

```
def print_seconds_per_day(days=1):  
    hours = days * 24  
    minutes = hours * 60  
    seconds = minutes * 60  
    print(seconds)
```

```
print_seconds_per_day(3)
```

```
print_seconds_per_day()
```





# Function return

*# Բոլոր ֆունկցիաները միշտ ինչ որ բան վերադարձնում են,  
# եթե return մենք չենք արել, ապա ֆունկցիան կվերադարձնի None*

```
def seconds_per_day(days=1):  
    hours = days * 24  
    minutes = hours * 60  
    seconds = minutes * 60  
    return seconds
```

```
res = seconds_per_day(3)  
res = seconds_per_day()
```

```
print(res)
```



## Function parameters

```
def my_function(a, b=15, c=30):  
    print(a, b, c)
```

```
my_function(12)
```

```
my_function(12, 67)
```

```
my_function(12, 67, 45)
```

```
my_function(12, c=45)
```

```
my_function(12, c=45, b=22)
```



# Function parameters

*# Ստացում ենք արժեքները որպես tuple*


```
def my_function(*params):  
    print(params)
```

```
my_function()
```

```
my_function(12)
```

```
my_function(12, 67)
```

```
my_function(12, 67, 45)
```



# Գործնական աշխատանք

Գրե՛ք ծրագիր, որը պետք է աշխատի `print()` ֆունկցիայի նման



# Function parameters

*# եթե նշվում է 2 աստղանիշ,*

*# ապա տվյալները ստացվում են dictionary-ում*

```
def my_function(**params):  
    print(params)
```

```
my_function()
```

```
my_function(John=18)
```

```
my_function(John=18, Mike=25)
```



# Function comments

```
# ինչպես նիստ նկարագրել ֆունկցիան
```

```
def area_of_disk(radius):
```

```
    """Help on function:
```

```
    :type: (float) -> float
```

```
    :param radius: disk radius
```

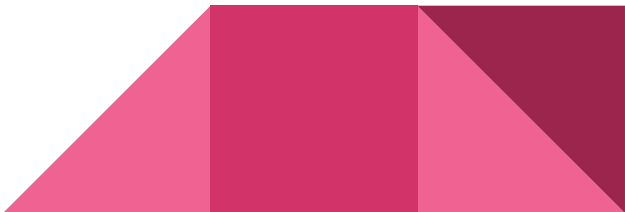
```
    :return: area of disk by radius
```

```
    """
```

```
    return 3.14 * radius ** 2
```

```
area_of_disk(34)
```

```
help(area_of_disk)
```



# dir()

```
import math
```

```
# dir ֆունկցիան վերադարձնում է մոդուլում եղած
```

```
# ֆունկցիաների ցուցակը
```

```
print(dir(math))
```



# import

*# Այս import-ը ամբողջ math գրադարանը միացնում է մեր պրոեկտին՝*

*# ֆյուն հիշողության մեջ*

```
import math
```

```
print(math.pi)
```

*# Այս տարբերակով մենք միայն միացնում ենք*

*# pi ֆունկցիան math գրադարանից*

```
from math import pi
```

```
print(pi)
```

*# Կարող ենք մի քանի ֆունկցիա միացնել*

```
from math import pi, cos, sin
```





import

*# Կոնֆլիկտներից խուսափելու համար կարող ենք*

*# ֆունկցիան անվանափոխել միացնելու ժամանակ*

from math import pi as my\_pi

print(my\_pi)



# Install Packages

PyPI - <https://pypi.org/>

*# տեղադրում եճք մոդուլը*

```
pip install package_name
```

```
pip install numpy
```

*# ջնջում եճք մոդուլը*

```
pip uninstall package_name
```



# numpy

```
import numpy
```

```
# բազմապատկում է ցուցակի էլեմենտները և վերադարձնում արժեքը
```

```
res = numpy.product([1, 2, 3, 10])
```

```
print(res)
```



# module

```
# երբ ֆայլը համարվում է մուտֆային,  
# նա ունի __main__ անունը,  
# իսկ երբ ֆայլը աշխատում է որպես մոդուլ, նա ունի այլ անուն  
print(__name__)
```

```
if __name__ == "__main__":  
    print("main module")
```



# Create package

*# Եթե մենք ցանկանում ենք ստեղծել թղթապանակ, որում պահվելու են մեր մոդուլները, այդ թղթապանակը պետք է դարձնել package: package ստեղծելու համար թղթապանակում պետք է ունենանք `__init__.py` ֆայլը*

```
import package_name.module_name
```



# Գործնական աշխատանք

Ունենալ հետևյալ ցուցակը՝

```
languages = ["Python", "Go", "JavaScript", "PHP", "Perl", "Swift", "Flask"]
```

Պետք է գրենք խաղ՝ Գուշակի բառը:

1. Խաղը միացնելուց հետո օգտվողը պետք է մուտքագրի անունը
2. Մենք պետք է օգտվողին ցույց տանք, որ նա ունի 3 հնարավորություն բառը գուշակելու համար և բացենք `input`, որով կարող է օգտվողը նշել բառ կամ տառ
3. Եթե օգտվողը սխալ բառ կամ տառ է նշում, պետք է ցույց տանք որ մնացել է  $-1$  – ով ֆիշ գուշակելու հնարավորություն
4. Եթե 3 անգամ սխալ բառ կամ տառ է մուտքագրում, ապա խաղը ավարտվում է և գրվում որ օգտվողը պարտվեց
5. Եթե մի տառ է գրում և այդ տառը կա բառում, ապա օգտվողի ֆայլերը չեն նվազում և գրվում է որ նման տառ կա
6. Եթե բառը ճիշտ է գուշակում, ապա գրում ենք որ օգտվողը հաղթեց խաղը:

# Ֆայլային համակարգ

```
# Default mode - read
```

```
# Բացում ենք ֆայլը կարդալու հոսք (stream)
```

```
open("test.txt")
```

```
# Երկրորդ պարամետրով նշում ենք ֆայլի հետ աշխատելու mode-ը
```

```
# r - read, կարող ենք միայն ֆայլից կարդալ
```

```
open("test.txt", "r")
```

```
# r+ կարդալ և գրել
```

```
open("test.txt", "r+")
```



# Ֆայլային համակարգ

*# w - ով ֆայլի պարունակությունը ամբողջությամբ ջնջվում է*

*# w - write, բացվում է stream ֆայլում գրելու համար*

```
open("test.txt", "w")
```

*# w+ գրել և կարդալ*

```
open("test.txt", "w+")
```





# Ֆայլային համակարգ

*# ֆայլը բացելուց հետո կուրսորը ամենավրջում է դրվում*

*# a - գրել*

```
open("test.txt", "a")
```

*# a - գրել և կարդալ*

```
open("test.txt", "a+")
```



## File - encoding

*# Նշում ենք ֆալի տեքստի կոդավորումը*

```
f = open("test.txt", "a", encoding="utf-8")
```



## File - close

*# Նշում ենք ֆալի տեքստի կոդավորումը*

```
f = open("test.txt", "r", encoding="utf-8")
```

*# աշխատեցինք ֆայլի հետ...*

*# վերջում պետք է ֆայլը փակենք*

```
f.close()
```



File - with - as

```
with open("test.txt", "r", encoding="utf-8") as f:
```

```
    # f - ում գտնվում է ֆայլը
```

```
pass
```




# File - read

*# read մեթոդը վերադարձնում է ֆալի պարունակությունը*

```
with open("test.txt", "r", encoding="utf-8") as f:  
    text = f.read()  
    print(text)
```

*# read մեթոդը կարող է ընդունել կարդացվող սիմվոլների քանակի*

```
with open("test.txt", "r", encoding="utf-8") as f:  
    text = f.read(5)  
    print(text)
```



## File - read

*# երբ որ read-ը կարդում է, կուրսորը տեղափոխվում է*

```
with open("test.txt", "r", encoding="utf-8") as f:  
    text1 = f.read(5)  
    text2 = f.read(5)  
    print(text1)  
    print(text2)
```



# File - readline

*# Փայլը կարդում ենք տող-տող*

```
with open("test.txt", "r", encoding="utf-8") as f:
```

```
    line1 = f.readline()
```

```
    line2 = f.readline()
```

```
    print(line1)
```

```
    print(line2)
```

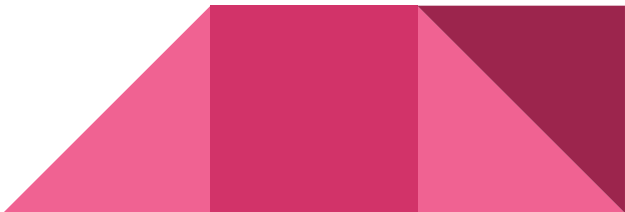
```
with open("test.txt", "r", encoding="utf-8") as f:
```

```
    line = f.readline()
```

```
    while line:
```

```
        print(line)
```

```
        line = f.readline()
```




## File - readlines

*# վերադարձնում է ցուցակ, ամեն տողը ցուցակի առանձին էլեմենտ*

```
with open("test.txt", "r", encoding="utf-8") as f:
```

```
    lines = f.readlines()
```

```
print(lines)
```





## File - write

*# a flag-ը կուրսորը տանում է ամենավերջ* *և*

*# վերջից կարող ենք write մեթոդով գրել*

```
with open('test.txt', 'a', encoding='utf-8') as f:  
    f.write('\nNew line')
```



# File

```
with open('test.txt', 'a', encoding='utf-8') as f:
```

```
    # Եթե ֆայլը դեռ բաց է է վերադարձնում է False, հակառակ դեպքում՝ True
```


```
    print(f.closed)
```

```
    # Վերադարձնում է mode-ը
```

```
    print(f.mode)
```

```
    # Վերադարձնում է ֆայլի անունը
```

```
    print(f.name)
```



## File - csv - reader

```
import csv
```

```
with open('data.csv', 'r', encoding='utf-8') as f:
```

```
    # reader մեթոդը առաջին պարամետրով ստանում է ֆայլը
```

```
    # երկրորդով բաժանման նշանը
```

```
    reader = csv.reader(f, delimiter=',')
```

```
    for row in reader:
```

```
        print(row)
```



## File - csv - writer

```
import csv
```

```
with open('data.csv', 'a+', encoding='utf-8') as f:
```

```
    writer = csv.writer(f, delimiter=',')
```

```
    # Ֆայլում ավելացնում ենք նոր տող
```

```
    writer.writerow(['John', 'Doe', 23])
```



# OS

```
import os
```

```
# Վերադարձնում է run եղած ֆայլի հասցեն
```

```
print(os.getcwd())
```

```
# Հնդուկում է թղթապանակի կամ ֆայլի հասցե և ստուգում,
```

```
# եթե ֆայլը կամ թղթապանակը կա, ապա վերադարձնում է True, հակառակ դեպքում`
```

```
False
```

```
print(os.path.exists('test.txt'))
```

```
print(os.path.exists('./hello/'))
```



# OS

```
import os
```

```
# Հնդուներն է հասցե և վերադարձնում այդ հասցենում եղած
```

```
# ֆայլերի և թղթապանակների ցուցակը
```

```
print(os.listdir('.'))
```

```
# Ստեղծում է թղթապանակ
```

```
# Առաջին պարամետրով ստանում է հասցեն/անունը,
```

```
# երկրորդով mode-ը ըստ linux օպ. համակարգի
```

```
os.mkdir('hello', 777)
```



# OS

```
import os
```

```
# Ջնջում է թղթապանակը
```

```
os.rmdir('hello')
```

```
# Ստուգում ենք թղթապանակ է, թե ոչ
```

```
print(os.path.isdir('test'))
```

```
# Ստուգում ենք ֆայլ է, թե ոչ
```

```
print(os.path.isfile('data.csv'))
```

```
# Վերադարձնում է ֆայլի չափը բայտերով
```

```
print(os.path.getsize('data.csv'))
```



# Գործնական աշխատանք

1. Ստեղծե՛ք ֆայլ `data.txt` անունով, որում ավելացրե՛ք ինչ որ տե՛քստ:
2. Գրե՛ք ֆունկցիա՝ `read_file('data.txt')`, որը որպես պարամետր ընդունում է ֆայլի անուն և վերադարձնում ցուցակ, որում այդ ֆայլի բոլոր ունիկալ/չկրկնվող բառերն են
3. Ստեղծե՛ք `data.csv` ֆայլ և ցուցակի ստացած բոլոր բառերը ավելացրե՛ք այդ ֆայլում

