

Online konzol: <https://repl.it/>

Google tananyag: <https://developers.google.com/edu/python>

Feladatok

- 1.) Írjon kódot amely bekér egy fájlnévet és kiírja a kiterjesztését
- 2.) Kérje be egy kör sugarát és számítsa ki a területét és írassa ki 2 tizedesjegy pontossággal.
- 3.) Kérjen be egy számjegyet (n) és képezze az $n + nn + nnn$ számot. Pl. $n = 6 \rightarrow 738$
- 4.) Készítsen egy `foo()` függvényt minta dokumentációval és írassa ki azt.
- 5.) Írjon programot, amely két megadott pont közötti távolságot számítja a síkban: `p1[1,2]` `p2[3,4]`
- 6.) Készítsen egy listát 10 elemmel, ahol minden elem négyzetszám és írja ki az 3. elemtől: eredeti lista: `[1,2,9,16,25,36...]` eredmény: `9,16,25`
- 7.) Adott a és b lista, írassa ki a közös elemeket: `a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]`, `b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]`
- 8.) Kérjen be egy mondatot a felhasználótól és írja ki szavanként visszafelé: pl. `aa bb cc -> cc bb aa`
- 9.) Készítsen függvény dekorátort, amely hívás előtt és után kiírja a 'before' és 'after' szavakat
- 10.) Készítsünk egy cache dekorátort, amely paraméterben adott ideig tárolja egy függvény visszatérő értékét

További Python feladatok és megoldások (angol):

<https://github.com/zhiwehu/Python-programming-exercises/blob/master/100%2B%20Python%20challenging%20programming%20exercises.txt>

Minta megoldás

1.)

```
filename = input("Input filename: ")
splitted = filename.split(".")
print ("here you are the extension of the file: %s" % splitted[-1])
```

2.)

```
from math import pi
r = float(input ("Radius : "))
print("r = %0.2f, area = %0.2f" % (r, pi * r**2))
```

3.)

```
a = int(input("type an integer: "))
```

```
n1 = int("%s" % a)
n2 = int("%s%s" % (a,a))
n3 = int("%s%s%s" % (a,a,a))
print(n1+n2+n3)
```

4.)

```
def foo():
    """
    Documentation for foo() function.
    No return value and args
    """

print(foo.__doc__)
```

5.a)

```
import math

p1 = [1, 2]
p2 = [3, 4]
distance = math.sqrt( ((p1[0]-p2[0])**2)+((p1[1]-p2[1])**2) )

print(distance)
```

5.b)

```
from collections import namedtuple
import math

Point = namedtuple('Point', ['x', 'y'])

p1 = Point(0,0)
p2 = Point(x=2,y=3)

dist = math.sqrt((p1.x - p2.x) ** 2 + (p1.y - p2.y) ** 2)

print(dist)
```

6.)

```
l = list()
for i in range(1,11):
    l.append(i**2)

print(l[3:])
```

“list comprehension”-el rövidebb:

```
k = [ i ** 2 for i in range(1,11) ]  
print(k[3:])
```

7.)

```
a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]  
b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]  
  
c = set(a).intersection(b)  
print(set(x for x in a if x in b))  
print(c)
```

8.)

```
s = input('Sentence :')  
  
l = s.split(' ')  
r = [item for item in l[::-1]]  
  
print(' '.join(r))
```

9.)

```
def logger(f):  
    def decorator():  
        print('before')  
        f()  
        print('after')  
    return decorator  
@logger  
def my_func():  
    print("my_func")  
  
my_func()
```

10.)

```
import time  
  
current_time_millis = lambda: int(round(time.time() * 1000))  
  
def cache(expiration_time):  
    def cache_decorator(f):  
        memo = {}  
        def helper(x):  
            if x not in memo:  
                memo[x] = {'value':f(x), 'time': current_time_millis()}  
            else:  
                if current_time_millis() - memo[x]['time'] > expiration_time:  
                    memo[x] = {'value':f(x), 'time': current_time_millis()}
```

```
        else:
            print("from cache: %s" % memo[x]['value'])
            return memo[x]
        return helper
    return cache_decorator
@cache(5000)
def long_calc(val):
    print(val)
    return val

long_calc(12)
time.sleep(6)
long_calc(12)
long_calc(12)
long_calc(10)
```

From:
<https://edu.iit.uni-miskolc.hu/> - **Institute of Information Science - University of Miskolc**

Permanent link:
https://edu.iit.uni-miskolc.hu/tanszek:oktatas:informacios_rendszerek_integralasa:python_gyakorlatok

Last update: **2023/03/22 15:45**

