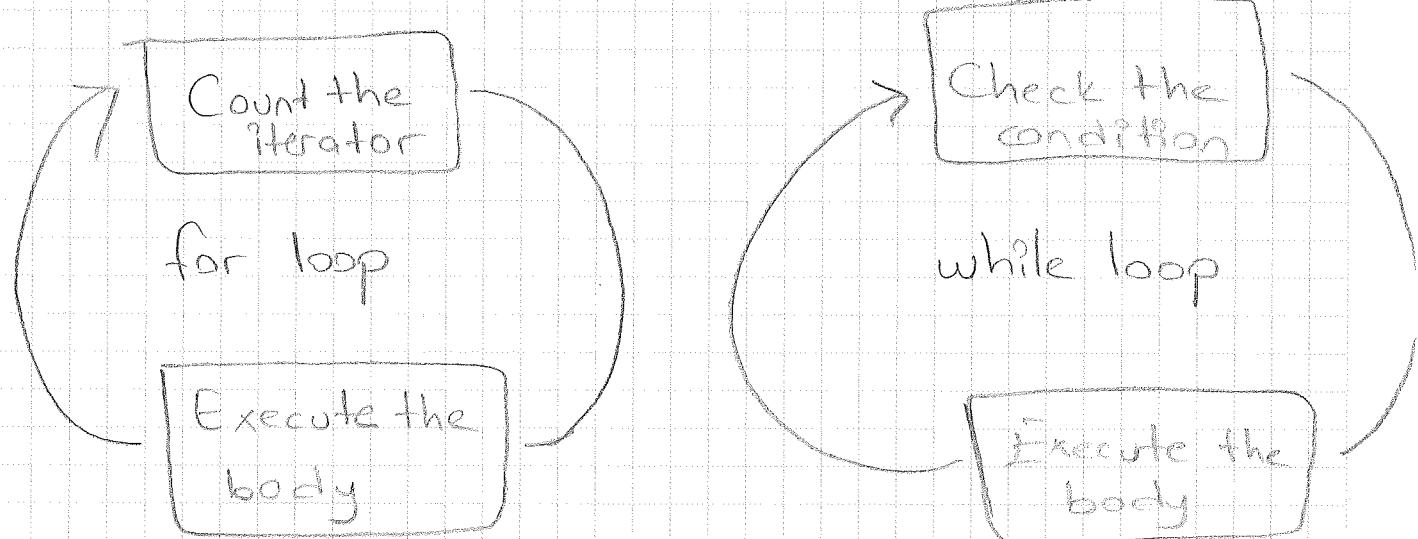


LOOPS



True döndürücü sırada
while devam eder.

TASK

Take the age of the user using `input()` and while loop.

Write a program that;

Takes the age from user.

Check the age if it is correct numeric format.

`age = input("Enter your age in correct format: ")`

`while not age.isdigit():`

~~Breaks the loop~~ print ("You entered incorrect! Write out correct format.")

`age = input("Enter your age in correct format: ")`

`print ("Your age is: ", age)`

Brüder gründen

Brunnen



white O:

print("bir")
print("ki")

→ bir ~~ki~~ Dangye gives birki while O = 1 digit



white []:

print("white")
print("outside of white")

→ outside of white



white None:

print("white")
print("outside of white")

→ outside of white

.isdigit() → int warcline False verir.

TASK

Guessing number:

answer = 28

question = 'What a two-digit number am I thinking of?'
print ("Let's play the guessing game!")

while True:

 guess = int(input(question))

 if guess < answer:

 print ('Little higher')

 elif guess > answer:

 print ('Little lower')

 else:

 print ('Are you a MINDREADER!!!')

 break

TASK

Find and print the length of the longest word.

Write a program that;

Takes a string sentence consisting of a couple of words from the user,

Compares and find out the longest word and prints the whole sentence and the length of the longest word as int type

Use while loop

```
sentence = input ("Give me a sentence without any punctuation")
```

```
words = sentence.split() (Punjabi style)
```

```
i = 0
```

```
longest = 0
```

```
while i < len(words):
```

```
    if len(words[i]) > longest:
```

```
        longest = len(words[i])
```

```
    i += 1
```

```
print ("The length of the longest word is:", longest)
```

For Loop

* Write a program to say "hello names" from the following list

Print the result such as : "hello Samuel"
"hello Victor"

names = ["Ahmed", "Aisha", "Adam", "Joseph", "Gabriel"]

```
for i in names:  
    print("hello", i)
```

TASK

Write a program to create a list consisting of numbers from 1 to 5.

Print the result such as : [1, 2, 3, 4, 5]

```
list1 = []  
for i in range(1, 6)  
    list1.append(i)  
print(list1)
```

→ [1, 2, 3, 4, 5].

* list(range(1, 6))
→ [1, 2, 3, 4, 5]

* tuple(range(1, 6))
→ (1, 2, 3, 4, 5)

Range in
will print
range function
range get rid
value

* print(*range(1, 6))
→ 1 2 3 4 5

Iterable \Rightarrow Elemanlarına tek tek ayırabılırlar
eleman. (String gibi)

* for i in range(1, 6):
 print(i, end=" ")
 $\rightarrow 1 \ 2 \ 3 \ 4 \ 5$

* list(range(-4, 4))
 $\rightarrow [-4, -3, -2, -1, 0, 1, 2, 3]$

* list(range(-4, 4, -1)) \rightarrow step
 $\rightarrow []$ (Sadece sırsa git dediği yer sıralayamaz.)

* list(range(4, -4, -1)) (Burada sıralar)
 $\rightarrow [-4, -3, -2, -1, 0, 1, 2, 3]$

TASK 1

Write a program to separate the string taken from the user into its characters using for loop.

Print the result such as:

Input : "Clarusway"

desired output : c - l - a - r - u - s - w - a - y

word = "clarusway"

for i in word :

print(i, end = " - ") → Sonda da - olasızdır bunu olmasa
rstrip ile en sonakki - kesiilebilir

count = 0

Xeza;

for i in word :

count += 1 (harflemi sayyorum.)

if count < len(word) :

i = i + " - " (i deki mi harfler.)

print(i, end = "")

→ c - l - a - r - u - s - w - a - y



```
User = {  
    "name": "Daniel",  
    "surname": "Smith",  
    "age": 35}
```

```
for attribute in user:  
    print(attribute)
```

→ name

surname

age

} Python dictionary ler
iterate ettiğimizde
key ler iterate eder.



```
User = {  
    "name": "Daniel",  
    "surname": "Smith",  
    "age": 35}
```

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

```
    print(
```

}) Value ları alır.

User = {

 "name": "Daniel"
 "surname": "Smith"
 "age": 35

for key, value in user.items():
 print(key, ":", value)

→ name : Daniel

surname : Smith

age : 35

Operations with the for loop

TASK

This time, write a code block that asks the user a number between 1 and 10 then puts that number into the multiplication table.

for example, the output for 5 should be as follows:

$$5 \times 0 = 0$$

$$5 \times 1 = 5$$

:

$$5 \times 10 = 50$$

```
number = int(input("Enter a number between 1-10: "))
for i in range(11):
    print("{}x{} = {}".format(number, i), number * i)
```

TASK

```

1 2 2
3 3 3
4 4 4 4
5 5 5 5 5
:
```

}

Bunu nasıl yaparsın?

for i in range(1, 10):

 print(str(i)*i)

(Bakis acınızı değiştirdiniz. Bir seyi tekrar etmenin yolu str kullanmaktır.)

* print("8"*8)

→ 88888888

* a = set(range(0,10))

print(a)

→ {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

* a = tuple(range(0,10))

→

* print(*range(5,25,2)) # 2'ser atlayarak git

→ 5 7 9 11 ... 25

* print(*range(10,0,-2))

→ 10 8 6 4 2

zip → 2 iterable's, index no larına göre
match eder ve bir tuple seni yapar.

text = ['one', 'two', 'three', 'four', 'five']
numbers = [1, 2, 3, 4, 5]
for x, y in zip(text, numbers):
 print(x, ':', y)

→ one : 1
two : 2
three : 3
four : 4
five : 5

tuple
→ fermvar
olmayan gal-
siyot.
(zip)

for x, y in list(zip(text, numbers)):
 print(x, y)

→ one 1
two 2
three 3
four 4
five 5

$\% \rightarrow$ modulus

TASK

Write a program to choose and collect the even and odd numbers (1 to 10) in two different list.

Print the result such as: evens : [0, 2, 4, 6, 8]

odds : [1, 3, 5, 7, 9]

even = []

odd = []

for i in range (1, 10):

if i % 2 == 0:

even.append(i) (Bos listeye elemeet ısin append
else: yapılı.)

odd.append(i)

print(even)

print(odd)

→ [2, 4, 6, 8]

→ [1, 3, 5, 7, 9]

ASSIGNMENT 1

PRIME NUMBER

```
num = int(input("Enter a number:"))

flag = False

if num < 1:
    print(num, "is not a prime number.")

elif num == 1:
    print(num, "is a prime number.")

else:
    for i in range(2, num):
        if (num % i) == 0:
            flag = True
            break

    if flag:
        print(num, "is not a prime number.")
    else:
        print(num, "is a prime number.")
```

FUNCTIONS

A function is a block of code that executes some logic for you e.g. prints a text, deletes some data or square a number.

{ A function is a piece of code that only runs when it is called.

Functions simplify the coding process, prevent redundant logic, and make the code easier to follow.

Functions help eliminate mess in your code because it saves you from unnecessary repetitions.

{ To define a Python function, the def keyword is used.

parameter - argument

There is no significant difference between them in Python.

Calling function \Rightarrow use it!

You can enter or input data, known as arguments, into a function and it returns something good that you want.

Built-in Functions

Explain Python functions:

A function is a section of the program or a block of code that is written once and can be executed whenever required in the program.

A function is a block of self-contained statements which has a valid name, parameters list and body.

Functions make programming more functional and modular to perform modular tasks. Python provides several built-in functions to complete tasks and also allows a user to create new functions as well.

There are two types of functions → Built-in functions

(copy(), len(), count())

User-defined functions

Functions which are defined by a user known as user-defined functions.

`print()`, `int()`, `list()`, `input()`, `range()`
`max()`, `min()`, `sum()`, `round()`

Built-in
Tackle numbers

functions

DEFINING (CREATING) A FUNCTION

Write your own function.

That is called the user-defined function.

`def [function name] (Argument(s)):`
 `[Execution body]`
 `'''`
 `Indentation`
 `(leave four spaces)`

The keyword `def` introduces the name of the function.

Argument lists are optional but the parentheses are not.

* `def first_function(argument_1, argument_2):`
 `print(argument_1 ** 2 + argument_2 ** 2)`
 `first_function(2,3) # 23 + 33 = 13`
 `→ 13`



```
def multiply(a, b): # İstenilen gibi
    print(a*b)
    multiply(3,5)
    multiply(-1, 2.5)
    multiply('amazing', 3)
    → 15
    → -2.5
    → amazing amazing amazing
```



```
def motto():
    print("Don't hesitate to reinvent yourself!")
    motto()
    → 'Don't hesitate to reinvent yourself!'
```



How do we write a function in Python?

We can create a Python function in the following

manner.

Step-1 : to begin the function, start writing with the keyword def and then mention the function name.

Step-2 : We can now pass the arguments and enclose them using the parentheses. A colon, in the end, marks the end of the function header.

Step-3 : After pressing on enter, we can add the desired Python statements for execution.

- ! Function names should be written in lowercase and with underscores between words.

Execution of a Function, (Funktionen ausführen)

The functions you have seen so far did not return any types or values but executed some actions. In order to use later the output and data types generated by the functions in our program flow, we need to define our function using the keyword `return` in addition to `def`. Let's see what happens in the following example:

```
(*) def multiply_1(a,b):  
    print(a*b) # It prints something  
  
def multiply_2(a,b):  
    return a*b # Returns any numeric data type value  
  
multiply_1(10,5)  
print(multiply_2(10,5))  
→ 50  
→ 50
```

- ! The first function just prints some data that you passed into.

The second one generates a numeric type value.

If you check their types you will see:

```
print(type(multiply_1(10,2)))
```

```
print(type(multiply_2(10,5)))
```

→ <class 'NoneType'>

→ <class 'int'>

So, when we need it in our program, we can't use the result of the first function since it is NoneType data. But, the second one is Integer data that we can use it in the future when we need it.

* shadow_var = print("It can't be assigned to any variable")
print(shadow_var) # NoneType value can't be used

→ It can't be assigned to any variable

→ None



In the example above, we can't assign the result of print() function to a variable.

► If there are more than one keyword [return] in a function, then the execution of that function will end after the first [return].

Q) What is the return keyword used for in Python?

A) The purpose of a function is to receive the inputs and return some output. The return is a Python statement which we can use in a function for sending a value back to its caller.

* In order to use later the output and data types generated by the functions in our program flow, we need to define our function using the keyword `return` in addition to `def`.

IN CLASS (29.07.2021)

TASK 1 →

Python Program to sum the amount of odd and even numbers in a tuple / list.

Write a code that counts the odd and even numbers in a given list or tuple.

Print the result such as :

example list : [11, 2, 24, 61, 48, 33, 3]

example output : The number of even numbers : 3

The number of odd numbers : 4

number = [11, 36, 33, 66, 89, 21, 32, 16, 10]

odds = 0

evens = 0

for i in number :

if not i % 2 :

 evens += 1

else :

 odds += 1

print("The sum of odd numbers are:", odds)

print("The sum of even numbers are:", evens)

i % 2 → 0
not define de
not 0 our = 1
1 is true.

TASK

Python Program to sum of the numbers from 1 to between 1-75 (including)

Use for loop to make this calculation.

toplam = 0 (asiki işte ekleyeceğiz)

for i in range (1, 75):

 toplam += i
print (toplam)

→ 2775

NESTED FOR LOOP (BİLGİ İÇİNE)

for variable1 in iterable1:

 for variable 2 in iterable 2:

 body



who = ['I am', 'You are']

mood = ['happy', 'confident']

for i in who

 for ii in mood

 print(who, mood)

JARIN
YALDI
PERCUPSS
tar.

TASK

```
names = ["susan", "tom", "edward"]
```

```
mood = ["happy", "sad"]
```

example output : susan is happy

susan is sad

tom is happy.

```
for ? in names:
```

```
    for ?? in mood:
```

```
        print (? , "is" , ??)
```

* listem = []

```
for ? in range (5):
```

```
    listem.append(?)
```

listem

→ [0, 1, 2, 3, 4]

Aynisine list comprehension ile yapacagiz.

```
{ for item in iterable:
```

```
    expression }
```

[expression for item in iterable]

expression'la append'e gerek kalmayacak.

Gözüm

appendiz Virgülle Verdik Koyuy

[expression for i in range(5)] }
→ [0, 1, 2, 3, 4]

[item for item in range(5)] }
→ [0, 1, 2, 3, 4] }
Tek satırda
as öndeği lütfen
yaptık.

(*) [i**2 for i in range(5)] # Her elementin karesini alır,
[0, 1, 4, 9, 16] Virgil koyar

TEK SATIRDA IF YAPILIR MI?

(Ternary "if" Statement)

condition = True } condition = False }
if condition : | if condition else 0 }
 a = 1 # if body | } Tek
else : | satırda
 a = 0 | (a kullanıysa
print(a) | gerek yok)
→ 1
a = 1 if condition else 0
print(a)
→ 1

if-body if condition else else-body

↳ Tek satırda yazma formu

$i \% 2 \rightarrow$ tek denek

List comprehension'a calış
peki anlamadın,

print(1 if False else 0)

$\rightarrow 0$

(*) $listkm = [1, 2, 3, 4, 5, 6]$

$[i ** 2 \text{ for } i \text{ in } listkm]$

vv

Sadece teklerin karesini istiyorum. Ne yapacğım?

condition $\Rightarrow i \% 2$

body $\Rightarrow [i ** 2 \text{ for } i \text{ in } listkm]$

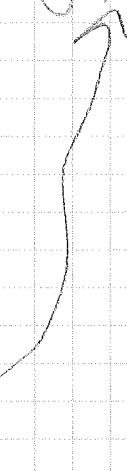
[if-body if $i \% 2$]

$\underbrace{\quad}_{\text{body}}$ $\underbrace{\quad}_{\text{condition}}$

X

$[i ** 2 \text{ for } i \text{ in } listkm \text{ if } i \% 2]$

$\rightarrow [1, 9, 25]$



\downarrow
Bu doğruya sa sol
taraf calisir.

KAHOOT

- ① What happens when a while loop is executed?
The program repeats until the condition false.

* for i in "bus":
 print(i, end="")
→ bus

* count = 0
for i in ['bus']:
 count += 1
 print(i, count, end="")
→ bus 1

Iterable var liste
Bir tane str element var
Bir tane str count = 1, dır.

FUNCTIONS - IN CLASS

Functions free us from chaos.

Bir kere def ile tanımla. Gözde yapın kullan.

Gözde = Kullan aynı sey.

(*) multiply(2,5)

↙
name of
the function

↘
arguments of
the function

(*) print("Say: I love you!")

↙
name of
the function

↘
arguments of
the function

BUILT-IN FUNCTIONS

print(), int(), list(), input(), range()

all(iterable), any(iterable), callable(object) } { Boolean type

bool(), float(), int(), str() } { Convert
(convert) character

dict(), list(), tuple(), set(), len(), zip() } { Collection type

filter()

Yazılımcılar

pre - lesson notes ver.

all funkisyon → True - False venir.

max(), min(), sum(), round() } tackle numbers

[ALL FUNCTION]

(*) names = ["susan", "Tom", "false"]

mood = ["happy", "sad", 0]

empty = {}

print(all(names), all(mood), all(empty), sep="\n")

→ True

False

True



! all → and gibi salınır. Bir tane
bile false varsa false
venir.

Collection boş ise True venir.
(Haydiaaa)

[ANY FUNCTION, (All'in tersi)]

list A = ["susan", "Tom", false]

list B = [None, (), 0]

empty = {}

print(any(listA), any(listB), any(empty), sep="\n")

→ True

False

False

! Bos collection False döner.
En az bir tane true veya
True olunur. Or gibii.

filter() Function

none iterable veya fonksiyon alır.

True ya da False verir.

`filter(function, iterable)`

any veya all
gelebilir

İlla fonksiyon kullanıacaksan none 'ı da kullanabilirsin.

None kullanırsan iterable'in elemanlarından sadece True'ları alır.

(*) `listA = ["susan", "tom", False, 0, "0"]`

`filtered_list = filter(None, listA)`

`print("The filtered elements are: ")`

`for i in filtered_list:`

`print(i)`

→ susan } filter içinde
 tom } None kelimesi kullanıldığı için
 0 } sadeceTruthy'ler sevi

(Filter ketim fonk. Gerekçe hale getirmek
için for döngüsünün içinde saklıyoruz.)

Team work

→ 150 den kisalma
kisalma
a)

* list L = [12, 15, 32, 42, 55, 75, 122, 132, 150, 180, 200]
divisible = []
for i in list L:
if i % 5 == 0 and i <= 150:
divisible.append(i)
print(divisible)

* Sayın kaç basamaklı olduğunu bul.
num = int(input("Enter a number:"))

count = 0
while num != 0: ← veya → while num > 0:
num // 10
count += 1
print("Total digits are:", count)



body

↑

```
generate = (i**2 for i in range(6))  
print(*generate)  
→ 0 1 4 9 16 25
```



```
generate = (i**2 for i in range(6))  
print(next(generate))  
→ 0  
print(next(generate))  
→ 1  
print(next(generate))  
→ 4  
:  
print(*generate)
```

{next bütün elementleri sırayla iterate etti}

→ generate kalan iterate edil meyveleri * ile iterate etmisi oldugu

ASSIGNMENT (ARMSTRONG NUMBER)

$$371 = 3^3 + 7^3 + 1^3$$

While döngülerinde kollarıda girilen sayıya
takip eden kader döndürür.

while True :

```
    number = input("Enter a positive integer number: ")
    digits = len(number) → #Bir hepsini numaraya
    if not number.isdigit(): →
        print(number, "is invalid entry. Enter valid input!")
    elif int(number) >= 0
        for i in range(digits)
            sum = sum + int(number[i]) ** digits
        if sum == int(number)
            print(number, "is an Armstrong number")
            break
        else:
            print(number, "is not an Armstrong number. Sorry!")
            break
```

Belli 50-60
gözümüz var

ASSIGNMENT (Prime number)

"Ornegim 17 asal midir?

1'den 17'ye kadar bütün sayılarca 17'yi bölmeli

lazım

```
n = int(input("Enter a positive number to check if it is  
prime number: "))  
count = 0 (Sayac)  
for i in range(1, n+1):  
    if n % i == 0:  
        count += 1  
if (n == 0) or (n == 1) or (count >= 3):  
    print(n, "is not a Prime Number")  
else:  
    print(n, "is a Prime Number")
```

FUNCTIONS - IN CLASS

Built-in Functions

enumerate (iterable, start=0)

tuple tuple iterable varit.

grocery = ['bread', 'water']

enum_grocery = enumerate(grocery)

print(type(enum_grocery))

print(list(enum_grocery))

enum_grocery = enumerate(grocery, 10)

print(list(enum_grocery))

→ ? Yasmeen dum

max (iterable), min (iterable)

Max ve min degelerini verir.

Parentez faire
yandırma argument hr. (arg)

sum (iterable, start)

sum (iterable, start)

Y) Liste lâncı toplanır istenilen birek
toplular ekler.

round (numbers, ndigits), Yuvarla

print(round(12))

print(round(10.8))

print(round(3.665, 2)) Noktadan sonra 2 tane

print(round(3.675, 2)) yuvarla diyor.

→ 12

11

3.67

3.67

built-in \Rightarrow Python'a bir türde iken tanımlanmış
tekrar tanımlanma gerek yok.

DEFINING (CREATING) A FUNCTION

`def function name (Parameter(s)) :`
 Execution body
 `Indentation`
 x, y

(*) `def first_function(argument_1, argument_2):`
 `print(argument_1 ** 2 + argument_2 ** 2)`
 `argument_1 ** 2 + argument_2 ** 2`

(*) Fonk. arguments parameter'ları yer değiştirebilir argument
`max(arg)`

`first_function(2, 3)`
 $\rightarrow 13 \quad \# 2^2 + 3^2 = 13$

(*) `def motto():`
 `print("Don't hesitate to reinvent yourself!")`
 motto() # it takes no argument
 \rightarrow Don't hesitate to reinvent yourself!

~~Task~~

YAP MUTLAKA

Define a function named add to sum two numbers and print the result.

```
def add(a, b):  
    print(a+b)  
add(-3, 5)
```

→ 2

KAHoot

* def who(first, last):
 print(first + last)

```
who(last = 'Clarus', first = 'Way') # Siralı deşifre  
who("Clarus", "Way") # Siralı deşifre
```

→ Way Clarus

ClarusWay

* def my_func(first = "1", second = "2"):
 print(int(first + second))

my_func()

→ 12

1+2 oldugu icin yan yana
yazar ~1'2

(*) def my_func(first="1", second="2"):
 print(int(first+second))

my_func(2,3) # Argument int.

→ 5

Fonk. on tanımlama my_func(2,3)

değeri 2 in print arguments o zaman

$$2+3=5 \text{ olur}$$

(*) def my_func(first="1", second="2"):
 print(str(first+second))

my_func(second=2, first=3)

→ 5

int istedigl iin

$$2+3=5$$

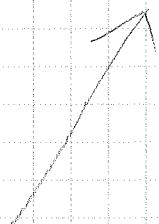
TASK → Min max
her ol gidiş
bir neşesi

Define a function named [calculator] to calculate
for math operations with two numbers and print
the result

Warn user in case of wrong entry: "Enter valid
arguments."

calculator(88,22,"+")

→ 110



(+, -, /, *) herci birsey gelirse bunu
desin

```
def calculator(a, b, opr):  
    if opr == "+":  
        return a+b  
    elif opr == "-":  
        return a-b  
    elif opr == "/":  
        return a/b  
    else:  
        return a*b  
    print("Enter valid argument")
```

calculator(5, 2, "-")

→ 3

} print yerhe
return
yazdirak
arthaklik
parin
print(calculator
 ('9/2', '+'))
demekiz
→ 11

Execution of a Function

```
def multiply_2(a,b):  
    return (a*b)  
print(multiply_2)(10,5)
```

Task

Define a function named abs_val to calculate and return absolute value of the entered number.

You can add docstring for an explanation.

```
print(abs_val(3.3))
```

```
print(abs_val(-4))
```

→ 3.3

4

```
def abs_val(num):
```

""" This function gives absolute value /
 of the entered number. """

if num >= 0:

return num

else:

return -num

```
print(abs_val.__doc__)
```

→ This function gives absolute ...

.py dosyaları python dosyasıdır.

GROUP - WORK

Bir sayıının faktöriyeli.

```
num = int(input("Enter a positive number: "))

factorial = 1

if num < 0:
    print("Please enter a positive number!")
elif num == 0:
    print("0! = 1")
else:
    for i in range(num):
        factorial = factorial * (i+1)
    print(factorial)
```

TASK

* Kullanıcıdan 3 sayı iste
Büyükten ortaçaya sırasına
Ortalardan küçüğe sırasına.

mylist = []

while True:

 mylist.append(int(input("Değer girin: ")))

 if len(mylist) >= 3:

 break

mylist.sort()

mylist.reverse()

print(mylist)

print("Büyük sayı ilk ortaçının değer =", max(mylist)-mylist[1])

print("Küçük son ortaçının değer =", mylist[1]-min(mylist))

Sıralanıyor
sonra 1.ortak
değer sayılır

PRE-CLASS 05.08.2021

Execution of a Function

```
def multiply_1(a,b):  
    print(a*b) # it prints something  
  
def multiply_2(a,b):  
    return a*b # returns any numeric data type value
```

```
multiply_1(10,5)  
print(multiply_2(10,5))
```

→ 50
50

As you noticed, the outputs are the same. Then what is the difference? Well, the first function just prints some data that you passed into. The second one generates a numeric type value. If you check their types you will see:

```
print(type(multiply_1(10,2)))
```

```
print(type(multiply_2(10,5)))
```

→ 20

→ <class 'NoneType'>

→ <class 'int'>

→ NoneType

→ Integer data

↳ Gelecekte 85 keşen
ağır kullanım.

(*) shadow_var = print("It can't be assigned to any variable")

print(shadow_var) # NoneType value can't be used.

→ It can't be assigned to any variable

→ None

→ Fonksiyonun sonucu bir deşikte atayamaz.

What is the return keyword used for Python?

The purpose of a function is to receive the inputs and return some output. The return is a Python statement which we can use in a function for sending a value back to its caller.

(*) def my_function(a, b):
 area = a * b
 return area

print(my_function(3, 4))

→ 12

(*) def my_function(a, b):
 hypotenuse = (a**2 + b**2)**0.5
 return hypotenuse
print(my_function(3,4))
→ 5.0

(*) def my_function(a,b)
 area = a * b
 return area
print(my_function(3,4))
→ 5.0

(*) def longer(a, b):
 if len(a) >= len(b):
 return a
 else:
 return b
print(longer('Richard', 'John'))

IN-CLASS (05.08.2021)

THE MATTER OF ARGUMENTS

Parameters ~ Neredeyse günler Arguments

Variables
(Değişken)

Farklı türmlere deşerler

def who(first, last):

print('Your first name is:', first)

print('Your last name is:', last)

Values

Farklı nesneler
"William van Rammelde"

who('Guido', 'van Rossum')

print()

who('Mary', 'Bald')

POSITIONAL ARGUMENTS

The positions the order of the parameter matter.
(Sıra önemlidir!) Pozisyon

[TASK] - Hrabisi? antacadim. Göye basit

Define a function named texter to print the following output in accordance with input

a = "I"

b = "love"

c = "you"

texter(c, a, b)

a = "

b = "love"

c = "you"

def texter(text1, text2, text3):

print(text2, text3, text1)

texter(c, a, b)

→ I love you

KEYWORD ARGUMENTS

kwargs → Keyword'un kusatılımlı kullanımı.

kwargs = values

Ön tanımlama

Positional Argument'ler farklı sırayla önemlidir.
olması.

Keyword argumentlerle yapıldığında sırayı önem? Kalmaz.

TASK ("ÖnceTask'n oynın")

Call the `texter` function using keyword arguments
in order to get the same output as the previous
one.

```
def texter(text1, text2, text3):  
    print(f'{text2} {text3} {text1}f')  
texter(text1="you", text3="love", text2="i")
```

(Varsayılan değer atamak)

We can assign default values to the parameters

when defining a function:

```
def func(x="ali", y=11):  
    body  
(Gözlemeşek bile bu fonk. ekti verir.)
```

Position ve keywordlarda olduğu gibi "harde".
Once positionlar yazılır, en sona keywordlar yazılır.

```
def argu(a, c, b = "dunya", d = "saturn")  
    print(a, b, c, d, sep = "\n")
```

```
print(argu())
```

→ Error. (Lütfen atanmamış değerler var, (a ve c))
(Onlara değer atanmamış gerekliydi.)

```
print(argu("uranus", "jupiter"))
```

→ uranus

dunya

jupiter

saturn

} # Silayla hep biri yazılır.

Position ile yapıldıktan sonra silahname
önerili.

ARBITRARY NUMBER of ARGUMENTS

Basına yıldız koymak belirtir sayıda parametre
ve belirtir sayıda* argument formulayabilmemizi

*args and **kwargs

def name (*parameters)

name (multiple args) →  tek yıldız parametresini dictionary'e çevirdi.

def name (**parameters)

name (multiple kwargs)

⚠ Tek yıldız parametresini iterable hale getirir.
Ycollection'us gibi

Positional \Rightarrow Tek *

Keyword \Rightarrow Gitti **

⚠ Sayısı belirsiz olan fonksiyonlarda * kullanılır.

Basın belki 100, 1000 Jane Janımlısk 800 -
yazın tek tek mi formülasyonu. Hani sırmışmış?

TASK

Define a function named `slicer` to collect even numbers into the list `evens`, odd numbers into the list `odds` from the given numbers by using `*args`.

```
slicer (1,2,3,4,5,6,7,8,9)
```

```
def slicer(*num):
```

→ Basina yildiz koydymiz
yeli istegebiym
kader yesem, And, t
elastik jet ketsem,

```
    evens = []
    odds = []
    for i in list:
        if i % 2 == 0:
            evens.append(i)
        else:
            odds.append(i)
```

```
    print("even list:", evens)
```

```
    print("odd list:", odds)
```

```
slicer (1,2,3,4,5,6,7)
```

→ even list : [2,4,6]

odd list : [1,3,5,7]

! dict fonksiyonu nasıl kullanılır di?
keys = value

* test = dict(isim = "ahmet", soyisim = "mehmet")
print(test)
→ {isim = "ahmet", soyisim = "mehmet"}
Dict'e atıfta

* def animals (**kwargs):
 for key, value in kwargs.items():
 print(value, "are", key)
animals (Carnivores = - - - - -)

Task

Define a function named [organizer] to collect the given names into the list names, ages into the list ages by using **kwargs.

organiser (Beth=26, Oscar=42, Justin=18, Frank=33)

Expect output → ['Beth', 'Oscar', 'Justin', 'Frank']
[26, 42, 18, 33]

def organizer (**kwargs):

 names = []

 ages = []

 for key, value in kwargs.items():

 names.append(key)

 ages.append(value)

 print(names, ages, sep="\n")

KALIR
X

ASSIGNMENT 1 (Prime Number)

Prime number

Use nested for loop

Print a list between 1 - 100.

```
my_list = []
for i in range(2, 100):
    prime = False
    for j in range(2, i):
        if (i % j) == 0: or if (i % j) != 0:
            prime = True
    if prime is False:
        my_list.append(i)
print(my_list)
```

ASSIGNMENT 1

Given two integer values, return their sum.

If the two values are the same, then return double their sum.

```
def sum_double(x, y):
    if x == y:
        return 2 * (x + y)
    else:
        return (x + y)
```

`filter()` içinde funk + iterable yazılır.)

↳ Boolean sonus döndürmesi lazer.

IN-CLASS, 09.08.21

- ! In a function call, keyword arguments must follow positional arguments.

*args and **kwargs

`def name(*parameter)`

name(multiple args)

** → dict'e cevibr.

`def name(**parameter)`

name(multiple kwargs)

(*) `def brothers(bro1, bro2, bro3):`

print("Here are the names of brothers: ")

print(bro1, bro2, bro3, sep='\\n')

family = ['tom', 'sue', 'tim']

brothers(*family)

→ Here are the names of brothers:

tom
sue
tim

(*) cümle = "başın olsa almak istenirdi" (Seslik her feli ol)

`def voweler():`

vowels = ['a', 'e', 'i', 'o', 'ö', 'ü', 'ǖ']

if letter.lower() in vowels:

return True

else:

return False

filtered_vowels = filter(voweler, cümle)

list(filtered_vowels)

Birinci
atama
yerine
Strongly
typed

1 TASK

Define a function named `merges` to prints the following output from the given tuple `genius`.

Call the function using variable `genius` as *args
`genius = ('Bill', 'Rossum', 'Guido van', 'Gates')`

`merges()`

For me, Bill Gates and Guido van Rossum are geniuses.

```
def merger(a,b,c,d):
```

return "For me, {} {} and {} {} are geniuses."

`format(c,b,a,d)`

`genius = ("Bill", "Rossum", "Guido van", "Gates")`

`merger(*genius)`

→ For me, Bill Gates and Guido van Rossum are geniuses.

Bir başka çözüm

```
def merger(arg1,arg2,arg3,arg4):
```

`print(f"for me, {arg1} {arg2} and {arg3} {arg4} are geniuses")`

`genius = ("Bill", "Rossum", "Guido van", "Gates")`

`merges(*genius)`

(*) def gene(x="Solomon", y="David"):
 print(x, " belongs to Generation X")
 print(y, " belongs to Generation Y")
dict_gene = { "y": "Marry", "x": "Fred"}
gene(**dict_gene)

→ Solomon belongs to Generation X
 David belongs to Generation Y

def gene(x,y):
 print(x, " belongs to Generation X")
 print(y, " belongs to Generation Y")

dict_gene
→ { "y": "Marry", "x": "Fred"}

gene(**dict_gene)
→ Fred belongs to Generation X
 Marry || || || Y

ziple çiftleri
match ettik

dict yaradık

EVLİNDİRME TASK'ı

dict_couple = { "bride": ["mary", "bella", "linda", "emma"] }

key value

"groom": ["jack", "robert", "eric", "adam"] }

key value

def miruvvet (bride, groom):

couple_list = []

for x in zip(bride, groom)

couple_list.append(x)

return couple_list

miruvvet(**dict_couple)

Bunun gönzini tek satırda yap

def miruvvet_2 (bride, groom):

return [x for x in zip(bride, groom)]

İsim
key yaş
 value

TASK

Friends içinde dict yap.

Names ve ages oluştur.

key value

meaner içinde bir fonk. tanımla, arkadaşları ort. yararla.

"The averages of ages: 46.0" gibi bir çıktı yazdır.

friends = {"Sofia": 35, "Anna": 30, "Nesim": 36}

def meaner(Sofia, Anna, Nesim):

$$\text{avg} = (\text{Sofia} + \text{Anna} + \text{Nesim}) / 3$$

print("The average of their ages are:", avg)

meaner(**friends)

→ The average of their ages are: 37.666666666666664

interview'te
arab.

replace()

üdeğitirme

PYTHON - METHODS FOR SOEV

Valid parenthesis (Dönüş parantez) se True,
yoksa False verin. () → True (Valid)
[] → False (Invalid)

Bos string kalana kadar devam et.

Herkes ekrassın "" kalsın.

While döngüsü kullanacağız.

X = "[{()}(){}][({})]"

def isValid(s):

while "()" in s or "{}" in s or "[]" in s:

s = s.replace("()", "").replace("[]", "").replace("{}", "")

return s == ""

isValid(x)

→ False

"alp-veli".replace("-", "+")
→ 'alp+veli'

Yazarsak → 'aliveli'

LAMBDA FUNCTIONS

LAMBDA : Anonim Bir Fonksiyon

Kullanıat

Tekrar yapılmaması

Bir fonksiyon tanımlanmadır

One-time defining a lambda function is
the best option.

lambda parameters : expression

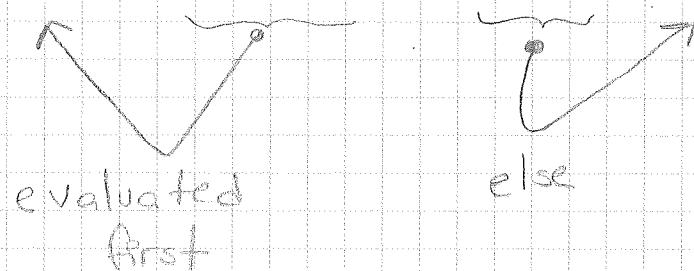
(*) def square(x)
 return x**2

↑ Do it with lambda function

lambda x : x**2
 return e Karsılık gelir

(*) lambda x, y : (x+y)/2

(*) lambda x : 'odd' if x%2 != 0 else 'even'



Normal if statement lar kullanıldığında lambda ile
tek satırda kullanılır.



(lambda parameters : expression)(arguments)

print(((lambda x : x**2)(2)))



average = (lambda x, y : (x+y)/2)(3,5)

print(average)



Positional arguments
(Sıra önceliği)



print(((lambda x, y : (x+y)/2)(3,5)))

TASK

Define a lambda function to reverse the elements
of any iterables.

Use parentheses for arguments and print the result.

print((lambda x : x[::-1])('clarusway'))

şart
Gözleme

print((lambda x : x.reverse())("safı"))

((lambda x : x[::-1])([1,2,3]))

((lambda x : x[::-1])("ita"))

→ ali

TASK

Bir liste verecek. Çiftler 'even', tekler 'odd' olacak.

```
for x in [1, 2, 3, 4]:
```

```
    print(((lambda x: "even" if x % 2 == 0 else "odd"))(x))
```

```
for x in [6, 12, -5, 11]:
```

```
    print(x, ":", ((lambda x: "odd" if x % 2 != 0 else "even"))(x))
```

SECOND USE FOR LAMBDA

```
average = lambda x, y: (x+y)/2
```

```
print(average(3, 5))
```

→ 4

```
((lambda x, y: (x+y)/2)(3, 5)) # İki sayı
```

→ 4

iterable = 'clarusway'

reverse = lambda x: x[::-1]

```
reverse(iterable)
```

→ yawsitalc'

map(), shift + tab + lab (İani? görmek için)

Built-in [map()] function

map() returns a list of the outputs after applying the given function to each element of a given iterable object such as list, tuple, etc.

Bir iterable ile bir iterable arasında bir iş yapın.

[map(function, iterable)]

→ Bu iterable'lar fonk. içinde sadece for döngüsündeymiş gibi teker teker uygulanır.



iterable = [1, 2, 3, 4, 5]

map(lambda x: x**2, iterable)

result = map(lambda x: x**2, iterable)

print(type(result)) # it's a map type

print(list(result))

print(list(map(lambda x: x**2, iterable)))

→ class 'map'

[1, 4, 9, 16, 25]

[1, 4, 9, 16, 25]

TASK 1

Do the same thing using user-defined function (def).

Use the `[def]` in `[map()]` function.

```
def square(n):  
    return n**2
```

```
iterable = [1, 2, 3, 4, 5]
```

result = map(square, iterable)

for i in



```
letter1 = ['o', 's', 't', 'l']
```

```
letter2 = ['n', 'i', 'e', 'w']
```

```
letter3 = ['e', 'x', 'n', 'o']
```

```
numbers = map(lambda x, y, z: x * y + z, letter1, letter2, letter3)
```

```
print(list(numbers))
```

→ [one, six, ten, two]

apple Japan
orange India
banana America
strawberry Australia
kiwi Africa

TASK

Using lambda in map() function, write a program that calculates the arithmetic mean of two element pairs in the following two lists in accordance with their order and collects them into a

List

```
numsl = [3, 6, 7, 4]
```

```
nums2 = [3, 6, 5, 8]
```

```
numbers = map(lambda x, y : (x+y)/2, (numsl, nums2))  
print(list(numbers))
```

TASK

```
kelimeler = ["ali veli deli", "mehmet agora kuzeli",  
"cenilin bacisi"]
```

Bunların uzunluklarını map() ile bul

```
uzunluk = map(lambda x : len(x), kelimeler)  
print(list(uzunluk))
```

list(map(len, kelimeler)) → Böyle de yapılı
bilir.

→ [13, 20, 14]

TASK

```
words1 = ["you", "much", "hard"]
```

```
words2 = ["I", "you", "he"]
```

```
words = ["love", "ate", "works"]
```

Use lambda in map(),

three meaningful sentences,

three lists

with their order

```
sentences = map(lambda X,y,z: x + " " + y + " " + z, words2, words3,  
for i in sentences:  
    print(i)
```

BUILT-IN filter() FUNCTIONS

[filter (function, sequence)]

first_ten = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

True
False
True
False
True
False
True
False
True
False

even = filter(lambda x: x % 2 == 0, first_ten)

print(type(even))

print('Even numbers are:', list(even))

<class 'filter'>

Even numbers are: [0, 2, 4, 6, 8]

→ çift sayıları filtreledi.

[TASK]

words = ["apple", "swim", "clock", "me", "Buy", "banana"]

Use lambda in filter() function,

with list

with less than 5 chars.

print(list(filter(lambda x: len(x) < 5, words)))

Yeye

for ? in filter(lambda x: len(x) < 5, words)

print(?)

→ swim
me

TASK 1

\neg }
 not
 \in }
 in
 \leq }
 less than or equal to
 \geq }
 greater than or equal to
 true veya false
 sonucu.

Sesli harfleri sız

Liste yap.

first_ten = [a, b, c, d, e, f, g, h, i, j]

vowels = [a, e, i, u, ö]

filtered_vowels = filter(lambda x: x in vowels, first_ten)

→ [a, e, i]

TASK 2

commands = ["right20", "right30", "left50", "down20", "up10"]

Koordinatta neye konusık gelmiş kodu yaz,

Beklenen çıktı $\Rightarrow \# [0, -10]$

Sayıla yazılı ayırmak için split() methodu kullanılır.
if sız yapılırsa,

Bunu iterate ederken for döngüsü kullanın.

$$x=y=0$$

```
for i in range(len(command)):
    if command[i].startswith("r"):
        x = x + int(command[i].split()[1])
    elif command[i].startswith("l"):
        x = x - int(command[i].split()[1])
    elif command[i].startswith("u"):
        y = y + int(command[i].split()[1])
    elif command[i].startswith("d"):
        y = y - int(command[i].split()[1])
[x, y]
→ [0, -10]
int(c[0].split()[1])
→ 20
```

Lambda within User-Defined Functions

```
def modular_function(n):  
    return lambda x: x**n
```

```
power_of_2 = modular_function(2) # 2^2
```

```
power_of_3 = modular_function(3) # 2^3
```

```
power_of_4 = modular_function(4) # 2^4
```

```
print(power_of_2(2)) # 2 to the power of 2
```

```
print(power_of_3(2)) # 2 to the power of 3
```

```
print(power_of_4(2)) # 2 to the power of 4
```

```
→ lambda x: x**2
```

→ 4

8

16

TASK

Bir fonk. Zanimla. `def` kullen. Lambda return et
3 kere, 4 kere tekrarla gibi bir fonk. Zanimla.

```
def repeator(n):
```

```
    return lambda x: x**n (x'e gönderdiğim string  
                                n kere tekrar ettim)
```

```
repeat_5 = repeator(5)
```

```
print(repeat_5("clansway"))
```

Lambda da print
kullanabilirsin.

TASK

Yazdıgın her yazının yanına emoji kay.

```
def functioner(emoji)
```

```
    return lambda message: print(message, emoji)
```

```
myprint_smile = functioner(":")
```

```
myprint_sad = functioner(":(")
```

```
myprint_neutral = functioner(":|")
```

```
myprint_smile ("hello")
```

```
myprint_sad ("hello")
```

```
myprint_neutral ("hello")
```

```
→ hello :)
```

```
hello :(
```

```
hello :|
```

```
myprint_sad ([1, "1", False])
```

```
→ [1, "1", False] :(
```

2. isteyi de sıklıkla
verir.

A2 once yaptığınız bu -


 (*) $(\lambda a: a + ":\") ("hello")$
 $\rightarrow "hello:"$

(*) $\text{smile} = \lambda a: a + ":\")$
 $\text{smile}("hello")$
 $\rightarrow "hello"$

(*) $\text{def } x():$
 $\quad \text{return } 1, 2, 3, 4$
 $\text{type}(x()) \rightarrow$ X fonk. çağırıldığında. Return'un içinden
 \quad dönen tuple return eder. Bu yüzden
 $\rightarrow \text{tuple}$

$x()$
 $\rightarrow (1, 2, 3, 4)$

$a, b, c, d = x()$ $\# x'?$ Uzakskere cittadim.
 \quad Buna unpacking denir.

$a, b, c, d = 1, 2, 3, 4$

for i in (x):
 $\quad \text{print}(i)$ $\rightarrow x'?$ for "long" sine de
 \quad sokarım

[Kan kere teksar etti? \Rightarrow iterable.count]



num = [1, 4, 4]

max(num, key=num.count)

$\rightarrow 4$

num.count(4)

$\rightarrow 2$ # 4, 2 kere teksar etmis



num = [1, 4, 4] # Birbirinin aynisi olan rakam

kan kere teksar eder?

def equal(a, b, c):

res = numbers.count(max(num, key=num.count))

if res > 1:

return res

else:

return 0

equal(1, 4, 4)

$\rightarrow 2$

equal(1, 2, 3)

$\rightarrow 0$

equal

| body if condition else body }
 | if
 [1 if True else 0]

Aynı soruda 3 değişkenle sabit kalkınak ifternesin:

```
def equal(*arg):  
    num = list(arg)
```

$res = num.count(max(num, key=num.count))$

```
if res > 1:  
    return res  
else:  
    return 0
```

equal(1, 2, 2, 2, 4)

→ 3

Tek satırda
 Atılık vada
 lambda si olacaksa?
 lambda si
 body

aynı si
 num
 sadece
 değişti

! Aynı soruya lambda yap.

equallambda = lambda x, y, z: [x, y, z].count(max([x, y, z], key=[x, y, z].count))

res'e karşılık
 gelmesi

#res pf res > 1 else 0

Bunu kodun içinde koycaz.

equallambda = lambda x, y, z: [x, y, z].count(max([x, y, z], \\\n key=[x, y, z].count)) if [x, y, z].count(max([x, y, z], keys[x, y, z].\\\n count)) > 1 else 0

! Ayni sayyu *arg manfiyala yapalm.

equalambda = lambda *X: list(x).count(max(list(x), key= lambda x: list(x).count)) if list(x).count(max(list(x), key=lambda x: list(x).count)) > 1 else 0

TASK

def file_function_generator():
 myPrint()
 myMax()
 myBool()
 mySorted()

} Hangi funk. yazýyorsa onu
zaklit etsin.

def function_generator(function)
 return lambda x: function

myPrint = function_generator(print)
myMax = function_generator(max)
myBool = function_generator(bool)
mySorted = function_generator(sorted)

Neyse başka bir çözüm:

```
def function_generator(function):
    return lambda x: function(x)
```

```
myPrint = function_generator(print)
```

lambda x = print(x) ("beni yardım")

LOADING MODULES (BİLİNEN PATH'LER KULLANACAK)

Fundamentals of Modules

Yazdığımız kod blokları → script

script.py

```
python codes }  
python codes } code block 1  
python codes }  
python codes }  
  
python codes } code block 2  
python codes }  
python codes }  
  
python codes } code block 3  
python codes }
```



Bir modül import metoduyla yüklenir.
(Yükle aktar.)

Library'ı bilgisayara indirir mi ne enin
değilim.

import → local computer 'dan

pip install → web'den

(*) import my-module as mym
mym.my-function()

Yazma daır

nickname
pep8

(*) from my-module import my-function as mfunc
mfunc()

→ sadece bire

kullanmak

zoruldugu.

Import module_1

Import module_2

Import module_3

Degiskenler ve fonks. Har Import
edilebilir.

BUILT-IN MODULES

[math]

`print(dir(math))` deyinəsi içindeki bütün fonksiyonları gösterir.

TASK

Let's import pi, factorial and log10 functions from module,

Print the value of pi, 4! and log10 of 1000 using these functions.

```
from math import pi, factorial, log10
```

```
pi
```

```
→ 3.1415
```

```
factorial(4)
```

```
→ 24
```

```
log10(1000)
```

```
→ 3.0
```

String modülü

```
import string  
pr(string)
```

→ : → içindeki fonk.ları gösterir.

* string.punctuation (Fonk. obsayı parentesi istenildi. Bu bir değişken)

Punctuation → Tüm noktalama işaretlerini verir.

* string.digits

→ ! - _ . (Number varyansı)

[datetime] Modül

[TASK]

Şimdiki tarihi ekrana

'import.datetime
dateTime.date.today()'

print(dateTime.date.today())

→ 2021-08-18

Font: 14pt
parantez

now function → Zamani verir.

print(dateTime.datetime.now())

→ 2021-08-18 22:55:07.757862

[TASK]

Using [datetime] module, write a program
calculate the following

According to the general acceptance, Ali
was born on 22nd April 571 AD, and
died on 8th June 632 AD.

How many days have he lived in his life?

| toordinal fonk → Milattan itibaren gün
| bazında sayıya sevindir. int

```
from datetime import date
```

```
birth = date(571, 4, 22)
```

```
death = date(632, 6, 8)
```

```
life_day = date().toordinal(death) - date().toordinal(birth)
```

```
print(life_day)
```

PACKAGES

(*) What is pip?

The package manager for Python.

Bir library'dır. Tüm library'lerin yönetmenini
yayayan komutlar içermektedir.

Jupyter, Colab, GitS'de çalışır.

(*) Bir şeyin versiyonunu görmek için olan kod:

```
Pip --version
```

* official guide'ı gidersen Pip library'sının
ana dokümanına gidersin.

Install komut

En çok kullanılan komut

`pip install 'my-package'`

`pip install my-package == 3.2.1`

Linux, Python her türlü kodu kullanabiliyor.

`pip install python == 3.8.1`

`pip list` → Daha önce install edilen her şey.

Bunların hepsi import edip kullanabileceğin demekti.

`pip show my-package` → Bir modülün özellikleri, nereye kaydedildi, versiyonu, herşeyini gösterir.

`pip uninstall my-package`

Install yapagina uninstall yapabilirim

TASK

$n = 876509755433$

Bu sayının tersen liste içinde tek satır string olarak yazdır.

`list(str(n))[::-1]`

$\rightarrow ['3', '3', '4', '5', '5', '7', '9', '0', '5', '6', '7', '8']$

2.yol

`print(['*' + str(n)[::-1]])`

3.yol

`print(list(reversed(str(n))))`

$\rightarrow \text{list} \times \text{reversed}$

$\rightarrow \text{!widget dosya}$

Bu şekilde github'dan bir dosya indirip, okurum.

Recursive Method \Rightarrow İainde kendini çağırın fonksiyon

```
def my_factorial(n):  
    sonuc = 1  
    for i in range(1, n+1):  
        sonuc *= i  
    return sonuc
```

factorial(5)

$\rightarrow 120$

2.yol

```
def my_factorial(num):  
    sonuc = 1  
    while num > 1:  
        result *= num  
        num -= 1  
    return result
```

my_factorial(5)

$\rightarrow 120$

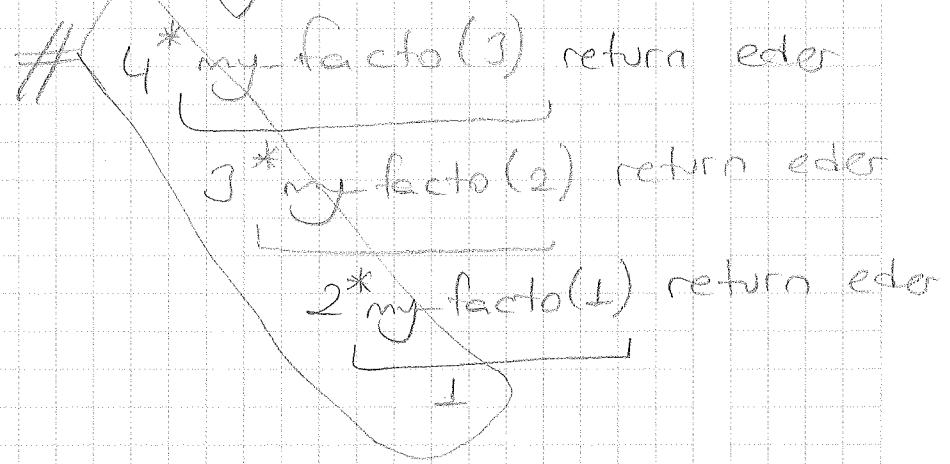
3. Yol 1

Recursive methodyla:

```
def my-facto(n):
    if (n == -1) or (n == 0):
        return 1
    else:
        return n * my-facto(n-1)
```

my-facto(4)

→ 24



Bunlar key'ler

[TASK]

Hesap makinesi sınıflasyonu yapınca,

ops isimli bir dict yap.

* deyince ekrana, / deyince bılsın, ...

Bunu lambda ile yap.

ops = { "+": (lambda x, y: x+y), "-": (lambda x, y: x-y), "*": (lambda x, y: x*y), "/": (lambda x, y: x/y) }

ops["+"](3,5)

→ 8

Value'ler lambda yaptı

Value'ler



bir_diksinir = { "ne_diyorum": "sesimi_duy",
"neden": ["bilmirem"] }

bir_diksinir["ne_diyorum"]

→ 'sesimi_duy'

type(bir_diksinir["neden"])

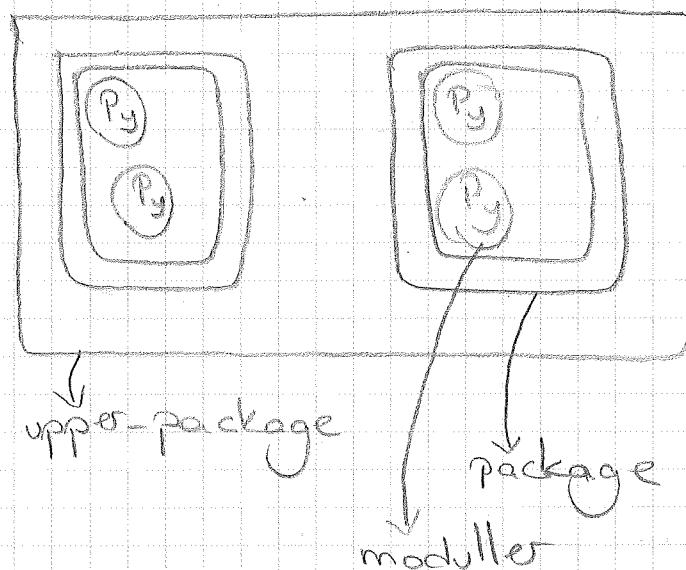
→ list

Dict comprehension yaparsak:

{x: y for x, y in bir_diktin.items()}

→ { 'ne-diyorum' : 'seçimi duy', 'neden': ['bilmirem'] }

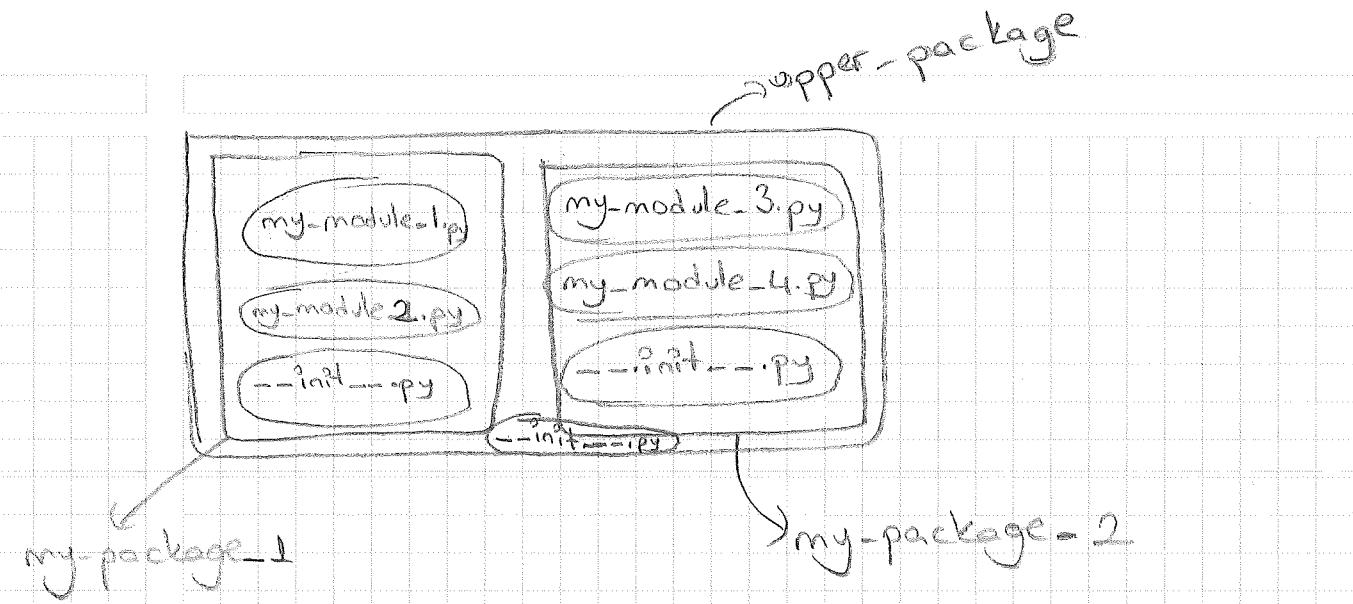
! Colab'da 'Content' klasöründen Working Directory
Galisacagınız yer burası



⇒ Package yapısı.
Klasör içinde klasörler,
onlarla içinde de modüller

Module ve package'lar oluşturulurken特别注意する
gerekli. Bunu init ile yaparız.

Bir txt belgesi oluşturursunuz
---init---.py yapıyoruz.



Jupyter gel:

pwd

→ 'C:\\Users\\Sony\\Desktop' (upper-package bürde
olmali)

import
etlik

{ from upper-package import my-package-1, my-package-2

from upper-package.my-package-1 import my-module-1,
my-module-2

dir(my-package-1) → My-package-1'in içinde
neler var onlar görünü.

→ ['__builtins__',

'__cached__',

'__doc__',

'__file__',

'my-module-1'

'my-module-2']

→ Buralar bir package'ın

default attribute'ler

tongue

Import ettik. Şimdi my-module'ın içinde bir fork yapalım:

my-module-2.divide(10,5)
→ 2.0

my-module-1.addition(4,5)
→ 9

my-module-3.repeater("clarusway", 3)
→ clarusway clarusway clarusway

my-module-4.sqrroot(9)
→ 3.0

Kendi yapmışınız
modül'ü kullanırsınız

(*) my-package-1.spec → Bu module'la
kali her özellik
görünür.

Import upper-package → Bu da package'ı import ettik.

dir(upper-package)

Dogruların fonksiyon import edelim:

from upper-package.my-package-1.my-module-2 import
divide, hello

hello ve divide fonk. import etmiş oldum. Artık
hello'yu galistiririm direkt calisır. Divide galistiririm,
direkt calisır.

hello()

→ hello

divide(22,11)

→ 2.0

math'ı import edelim;

import math

from math import log10

log10(10000)

→ 4.0

math modülünün doc stringine nesil ulasırız?

print(math.__doc__)

→ This module is always available. It provides access
to the mathematical functions defined by the C standard.

-- doc -- \Rightarrow Modül neyle ornekle alakalı kisa bilgi tespit etmek istedim kılavuzu gibi.

print(my_module_2.__doc__)

\rightarrow This is module 2. Welcome! (Kendi yazdığımız modülün doc stringine ulaşık)

1 23.08.2021

shuffle \rightarrow İkne aldığı listenin elementlerini karıştırır.

round(2) \rightarrow 2 kere yuvarla etmek

PROGRAM EXECUTION

Python kodu yazdık. Print ettiğimizde neye neler oluyor?

Python'u indirdiğimizde otomatik olarak Python环境 (env) oluşturuluyor.

Jupyter'in bir hizmeti bir py dosyası.

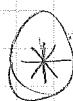
Python Shell → Python'un editörü.

Several Interpreter Tools → Python Shell

OS Console

IDEs

Jupyter



What is the interpretation process?

Yazdığımız kodların soldan sağa, yukarıdan aşağıya satır satır Python kodlarının okunması.



Python nedir?

Bu bir Interpreter iddir. Hesaplayıcıdır.

THE INTERPRETATION

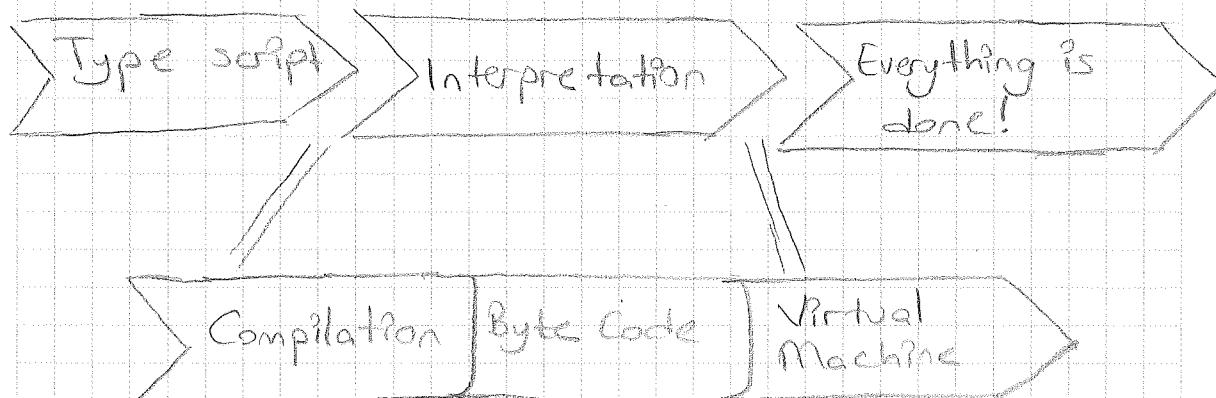
C Python

Python → Java kodlarını bytecode'ya çevirir.

PyPy → Bir tür Python. Sınıfı module'ı oluşturur
Bu yüzden daha hızlıdır.

IronPython

The Interpretation Process



VIRTUAL MACHINE

Bir software' dir.

Mazdığımız byte kodları sisteme tabii tutup ekle-
rir, okur. (Soldan sağa — yukarıdan aşağıya okut.)

ERRORS

① Syntax Errors

② Common Errors

⚠ Error ve exceptions fakta nedir? (Interview sonu)

① Syntax Error

Parantez kapatma hatalı

Hata mesajının en iyi kısmı şunlardır. Detaylı
bilgi verir.

Error satırı:

Traceback (---) ile başlar

(*) What is Traceback?

Bir modül'dür.

Hata ayıklama modülü. Bizi yönlendirir.

Hataları yorumlaştırır :)

639 satırda bir modül, bir py dosyası.

② Common Errors

- Tırnak hatası (Quotes)
- Parantez hatası (Wrong parenthesis)
- Yazım hatası (Wrong spelling & Typos) → Case sensitive
Indentation sensitif
- Indentation

```
X = ["1", "2", "3"]
```

```
y = ["USA", "Japan", "Spain"]
```

```
for i in y:  
    for j in x:  
        result.append(i+j)
```

```
result
```

Buy list
comprehension
in gap

List comprehension ile:

[$i + j$ for i in y for j in x]

! İki kodun farklı: Birinci yukarıdan aşağı istiyor.
Düzeni soldan sağa

Syntax Error

These types of errors are detected during compiling the program into byte-code (Habibir satır çalıştırılamaz.)

Exception Error

These types of errors are detected during the program execution (interpretation) process (Satır satır gidecek hata görünce takılır.)

Exception Error sonraki satırların çalışmasına engel olmaz. Hatalı satırın hâlde yazdırılır sonraki satırları etkisiz kılmaz.

Error'un son satırındaki açıklanmış "associated value" denir. Mesela:

`TypeError: can only concatenate str (not "int") to str`

associated value

COMMON EXCEPTIONS

- ① ValueError
- ② NameError
- ③ TypeError

Value Error → Tipi doğru, value yanlış.

`print(int('ten'))` → Tipi doğru ama yazılıya yazmaysqli.

`print(int("10"))` → Böyle yazıkları

TASK 1

Try to set a code to raise ValueError intentionally using the math module.

`import math`

`math.log(0)` `math.sqrt(-25)`

`math.factorial(-4)` → (-) sayıları faktörieli olur mu
hıca?

Name Error

```
print(variable)
```

```
variable = "Don't ever give up!"
```

→ Tanımlanmadığın için direkt hata verir.

Name error'da en çok : sunlardan hata olur.

Case-sensitive ve is not defined

TASK

Try to set a code to raise Name Error intentionally.

```
b = "string value 1"  
c = "string value 2"  
print(b)  
print(c)
```

↓
Boyuşk C'yi tanımlanmadır. Name error
verir.

Type Error

```
for i in range ('x'):  
    print(i)
```

Task 1

Try to set a code to raise Type Error intentionally

3 + 'a' bool(type()) range ("5")

Exception Handling

Bir Exception hierarchy var. (Arastır bak
(Error) diyor hoca)

try-except block:

→ Exception escrolar
yola getirir.

try:

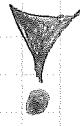
{ code block to be normally executed

except:

{ code block to be exceptionally executed

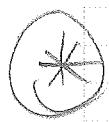
→ Exception ihtimali olan kod satırlarını bunun
içine yazınız.

Bir exception ile karşılaşınca sunu yap'ın
digmiz seyi bunun içine yazınız.



Bunu yapınca "hata çıktısa sunu yap"

demek için kullanılır. En çok while döngülerinde
kullanılır. Kod durmaz çalışmaya devam eder.



try:

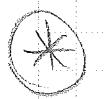
```
print(4 + "4")
```

except:

```
print("Muhtemelen yanlis bir seyler yaptin.")
```

→ "Muhtemelen yanlis bir seyler yaptin."

Yanlis oldugu iin except'e gitte. Kod calismadi.



try:

```
print("4" + "5")
```

except:

```
print("Muhtemelen yanlis bir seyler yaptin.")
```

→ 45 # Ciktiyi verdi qunki hata Jpk. Kod calisti.



except ZeroDivisionError:

Degerimizde yukardan farklı olarak hatayı spesifie ettik.

```
print("Sifira bolunme hatasi var. 0 haricinde  
bir sayi giriniz.")
```

TASK

Try to open a non-existing file named "my-file.txt" to read using try-except block,

Give the following message in case of exception raises:

'There is not such a file or the path is incorrect.'

try:

```
file = open("my-file.txt", "r")
```

```
print(file.read())
```

```
file.close()
```

except:

```
print("There is not such a file or the path is incorrect.")
```

→ Cukti olarak except'i yoldirdı.

FileNotFoundException var.

y

except FileNotFoundError:

print("The file does not exist or is not a file")

try - except statement'in full versiyonu:

try :

statements

(Error akneta bu salisir)

except :

statements

(Exception yükselirse kodu durdurup bunu salistirir.)

else :

statements

(Hata yükselirse else uygulanır. Yani try salisisa bu salisir. Print yaz, döngü yap, İstediğin yap)

finally :

statements

(Her koşulda bu salisir)



! Exception yükselirse else kodu salisir.
Finally her koşulunda salisir.

TASK

try:

sonuc = "4" + "5"

→ Buraya print("") yazarsam
her halükarda çalışır.

except TypeError:

→ Buraya print() yazarsam hata
olursa çalışmaz, hata olmazsa
alışır.

print ("Type hatalı was. Tipi kontrol etsen iyi
olur.")

else:

print ("Denek ki hata yükselenmiş.", sonuc)

finally:

print ("Eh nihayet bana sıra geldi. Onunde
sonunda çalışırım.")

→ try, else, finally çalışır.

try'da hata olduğunda try'in altı
alışmasız, except çalışır, finally çalışır.

else çalışmasız.

! except 'lerin' cogalma bilgisi.

except ValueError:

print("ValueError'a göre istem yap")

except TypeError:

print("TypeError'a göre istem yap")

except ZeroDivisionError:

print("ZeroDivisionError'a göre istem yap")

Böyle birde
fazla except
satırı yazılı
bilir. Hata ya
şınan yorsa
buna göre
istem yap
diyeğim.

Exception as

Kodun hangi hatayı verdiğini Python'a söyleyecez.

while True:

```
    no_one = int(input("The first number please: "))
```

```
    no_two = int(input("The second number please: "))
```

try :

```
    division = no_one / no_two
```

```
    print("The result of the division is: ", division)
```

```
    break
```

except Exception as e:

```
    print("Something went wrong... Try again.")
```

```
    print("Probably it is because of '{}'. error.format(e))
```

```
    break
```

Exception as e

↙

Exception yükselerise exception bloğu calisir

e değişkenine hangi exception olduğunu gösterir ve

assign edilir.

e'in type'i stringdir.

TASK

```
try:  
    a = 10
```

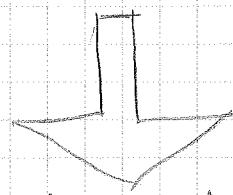
```
    b = 2
```

```
    print("The result of division is: ", c)
```

```
except Exception as e:
```

```
    print("The error message is: ", e)
```

→ The error message is: name 'c' is not defined



Herhangi bir satırda nasıl bir Error alacağımızı
kestiremeyeceğiz veya hata mesajı verdirecek bir
uygulama yazarsak,

hangi exception olursa ola gide bir mesaj
yazmak için bu formatı kullanırız.

Daha önce
tanımlanmadık

1 TASK

try:

x = 2/0

except ZeroDivisionError:

print('Attempt to divide by zero')

except:

print('Something else went wrong')

→ Attempt to divide by zero

→ rest of all exception possibilities

! Birde fazla except kullanılabılır. Bu tür errorları ayrı ayrı except bloklarına tanımlarım. En sona except yazdığında genelde her exception'ı kapsar.
(If - elif - else blogu gibi)

except (ValueError, TypeError):

print("You can only enter numbers consisting
of digits, not text!!")

Value error da gelir Type error da gelir bunu
yap demek. Virgülle ayırıp except'e grupta
yabılırlız.

Exception hierarchy from lowest to highest



...

except ArithmeticError:

print("I will also catch OverflowError
FloatingPointError and ZeroDivisionError")



Arithmetic error → OverflowError } Errorlerini
FloatingPointError } kapsar.
ZeroError

Task

Kullanıcıdan en sevdığı meyvenin index numarasını girmesini isteyelim. Mühtemel bir exception'ı ayrı ayrı kullan. While döngüsü kullan ki yanlış giriş oldursa bir daha sorsun. try-except bloğu kullan.

```
fruits = ["banana", "mango", "pear", "apple", "kiwi",  
          "grape"]
```

Hangi exceptionlar olabilir? Once bunları düşün.

Index error olabilir.

Kullanıcı print değil de str gitirebilse (Value Error)

Kullanıcı index sınırları dışına gitabileceğini (Index Error)

```
fruits = ["banana", "mango", "pear", "apple", "kiwi", "grape"]
```

while True:

Burda de -
meyi yapılıp -
n2. Error
olursa burda
olacak

try:

```
    index = int(input("En sevdiğiniz meyvein indexi:"))
```

```
    print("Favori meyve: ", fruits[index])
```

break

except IndexError:

```
    print("There is no such an index. Try again!")
```

except ValueError:

```
    print("You should enter integer. Try again!")
```

⚠️ Üstteki soruda kullanıcıın istediği kadar hakkı olmasın. 3 hakkı olsun. Bu yüzden bir count belirkeniniz lazım. Her seferinde bir azaltınan lazım.

count = 3

while count > 0:

try:

```
    index = int(input("En sevdiğiniz meyvein indexi:"))
```

```
    print("Favori meyve: ", fruits[index])
```

break

except IndexError:

count -= 1

```
    print(f"There is no such an index. You have {count} right remaining. Try again!")
```

count
burda
durdu

tongue
tongue

```
except ValueError:
```

```
    count -= 1 (*)
```

(Our code
breaks
here)

```
    print(f"You should enter integer. You have {count} tries remaining. Try again!")
```

```
else:
```

```
    print("Congrats! You have entered a valid input!")
```

```
    break
```

```
finally:
```

```
    print("Our fruits are always fresh!")
```

Her halükarda yazdıracak

▼ Exception yükselmesine else devreye girecek.
O yüzden 'break'? else bloğuna yazdık. Kodlar çalışırsa çıktı dedik.

Dosyalarda ilgili istenilen bantlarla yapılır.

READING FILES

write() } da
writeLines() varmış

3 okuma methodu var.

① Dosya object oluşturmak → open() function

.read() { Uzun bir sun methodu olurlar.
.readLine()
.readLines()

Dosya object oluşturmak.
open() Function

(*) file is the only required parameter of the open() function. This parameter is a path-like object which is a `str` or `bytes` that represents a path in the file directory.

`open(file, mode='r', buffering=-1, encoding=None)`

Bir dosyam Okunma yapacağım
var yazma mi?

Operating systeme alakalı
Genelde UTF8 olur.

(*) You can see the current path of your Jupyter using `pwd` command

`pwd`

Google Colab'in working
directory'i "content"

→ `'C:\Users\YD'`

Klasör

→ `open("file_name_and_path")`

* In the following example, you see that we open a .txt file object. We passed the required parameter (file) into the open() function.

```
my_file = open("first_file.txt")  
print(type(my_file)) # This syntax opens a 'txt' file
```

→ <class 'io.TextIOWrapper'>

* We can open a file in the same way by using the following syntax:

```
my_file = open("first_file.txt", encoding="utf-8",  
# we've used 'utf8' encoding format just the same as the previous one.
```

Encoding su an pain' učk' ðeneli? deði.
I am not agree with it.

Dosya açma istenilece
ne yapacığını söyler (r, a, w)

mode Parameter

"r" → Open for reading (default). If the file doesn't exist, FileNotFoundError will raise.

"a" → Open for writing. It will append to the end of the file if it already exists. If there is no file, it will create it.

(Dosyanın en sonuna ilave eder.)

"w" → Open for writing. It will be overwritten if the file already exists. If there is no file, it will create it. (Öncekiyi siler, yenişini ekler)



! import os

dosya = os.listdir()
dosya

Jupyter'e bunu yazınca
current directory'ün içinde
ne varsa liste yapar.
Hoca bunu çok kullanıyor mus.

! import shutil

shutil.

Buraya yazdığınız seye göre bir çok şey yapabiliyoruz. Dosyanızı kopyalama, kesme, zipleme gibi yapabiliriz.

Buraya dosya yolunu yazarsam da okur.

```
file = open("fishes.txt", "r")  
print(file.read())  
file.close()
```

)
with open bloku ile daha kisa yolu var. Sonra görevi?

```
file = open("fishes.txt", "r")  
print(file.read(33))  
file.close()
```

→ 33 karakter döndürür.

```
file = open("fishes.txt", "r")  
content = file.read()  
file.close()  
content
```

→ Tüm içeriği yazar.

len(content)

→ 205

content[:33]

→ 33 karakter döndürür

Satırın sonundaki
newline (\n) karakteri de
her zaman sayılır!
33'in içinde dahil olur.

Aynısı!

```

file = open("fishes.txt", "r")
print(file.read(1))
print(file.read(3))
file.close()

```

Bir dəha,
read yaptığın
da bəstan
yazmaz. Eñ
son kaldığı
yerdə okumaya
devam eder.
Ana sətər satır
geser. Gündü
print bir sətər
satırı gesin.

seek() METHODU

Cursor 'in kaldığı yerde kalmaz. Yenidən başa
döner.

```

file = open("fishes.txt", "r")
print(file.read(3))
print(file.read(3))
file.seek(0)
print(file.read(5))
file.close()

```

Once buntarı ard arda
kaldığı yerdə okur.

Burda tekrar başa döner.
Bəstan okur. 0 yerinə 5
yazarsam 5. karaktər
gider.

- ! • .txt
- .json
- .md
- .csv (virgülle ayrılmış dosya)
- .tsv (tab ilə ayrılmış dosya)

Text tabanlı dosyalar.
Str vən pəncəfiflər.

1 tell() Methodu

Cursor'ın en son kaldığı yer söyleş.

```
file = open("fishes.txt", "r")
print(file.read(3))
print(file.read(3))
file.seek(0)
print(file.read(3))
print(file.tell()) → Cursor'in nerede kaldığını
file.close                                söyleş. (Aktası → 34)
```

TASK

- ① Create a txt file named rumi.txt in your current working directory consisting of the following quotes of Rumi.
- ② Read and display the entire contents of this file at once.
- ③ Display the first two lines using read() method.
- ④ Display the next 13 chars of the content.
- ⑤ Display the current location of the cursor.
- ⑥ Bring the cursor onto beginning of the second line and display the second line again.
- ⑦ Close the file

I want to sing
Like the birds sing, txt
Not worrying about
Who hears or
What they think.

```
rumi = open ("rumi.txt", "r")
```

```
print (rumi.read(35)) → İlk iki satır okutuluk
```

```
print (rumi.read(13)) → Sonraki 13 satır okutuluk
```

```
print (rumi.tell()) → Nerde olduğumu söylemek
```

```
rumi.seek(15) → 15inci satırın basına dönük
```

```
print (rumi.read(21)) → 15inci satırı tekrar yazdırıldı
```

```
rumi.close()
```



Paleynatlı sayfalar derneğinden
ma file bitti.

readline() Method

Read dosyanın türündü okur.

ReadLine → Satır satır okutmak için kullanılır.

Newline ile işaretlenmiş her bir satır
ayrı ayrı okunamızı sağlar.

```
sea = open("fishes.txt", "r")
```

```
print (sea.readline())
```

```
print (sea.readline())
```

```
print (sea.readline())
```

Her print'in sonunda cursor
bir alt satırın basınına kalır.
Her readline()'da görüntüde
fazladan bir boşluk atanır.
Bir satır aralarına birer
boşluk bırakarak yazdırılır.



- Diyelim ki text'i parçalara okutacağız.
- Satırı okumayı bitirmeden alta gelmez. Tüm satırı bitirmemiz lazımdır (NewLine dahil)

Orca is a kind of Dolphin.\n

13

13

5

Buraya da 13
diyebilirim.

print(sea.readLine(13))

print(sea.readLine(13))

 Bu satır ile sadece ilk
 satırı okutabilirim.

print(sea.readLine(13))

 print(sea.readLine(13)) → Burda alt satırı geser.

→ Burda bir boşluk fazla bırakılmış. Çünkü 3. print
newline'a denk gelmiyor. Ordan da bir boş satır
ekler.

Task

- ① Read and display the first line of this file,
- ② Read and display the second line of this file,
- ③ Read and display the third line of this file
using `size` parameter in the method,
- ④ Close the file.

```
rumi = open("rumi.txt", "r")
```

```
print(rumi.readline())
```

ilk satiri okuttuk

```
print(rumi.readline())
```

ikinci satiri okuttuk

```
print(rumi.readline(100000))
```

icine birsey yaz demisti. Bunu yazsak okur. Satirum

```
rumi.close()
```

18 karakterliyse ondan buyuk

sousuz sayi yazabilmem ve

hepsini okusun.

-1 yazsam da okurmus.



Alt + shift + a → Yorum yapar (3 noktali)

Ctrl + # → Yorum yapar (# ile)

