

Ortamın Hazırlanması

- Python Çalışma Ortamının Hazırlanması:
 - python.org sitesinden engine indir ve kur.
 - Sonra Terminal, Notebook veya bir IDE ile kodla.
 - Terminal
 - Visual Studio Code
 - Jupyter Notebook
 - Spyder
 - Jupyter Lab
 - PyCharm
- Not: anaconda.com sitesinden Anaconda dağıtımı kurarak, tüm paket ve gerekli araçları bağımsız sanal ortamlar vasıtasıyla kolayca yönetebilirsiniz.
- Çalışma Modları:
 - Interactive Mode: Terminalde yaz ve sonuç al.

```
Terminal
> py --version
> py --list
> python
>>> print("merhaba", "dünya", sep=" ", end="\n")
>>> exit()
```

- Script Mode: Dosyalara (.py) yaz ve toplu çalıştır.

```
./selam_ver.py
```

```
print("merhaba...")
```

```
> py ./selam_ver.py
```

- Not: `py`, `py3`, `py -3` vs. ile interpretera bağlanır.

- Python Yorumlayıcısı: Kodlar makine diline çevrilir.



- Package / Module Kavramı Temelleri:

- Hazır builtins modül üyelerine direkt erişilebilir.
 - Örn: `dir`, `help`, `print`, `input`, `List`, `map` vs.
 - `dir(_builtins_)` gömülü nesneleri listeler.
- Gömülü nesneler dışındakileri paketlerden alırs.
- Python kurulumuyla bazı standart paketler gelir.
 - Örn: `math`, `datetime`, `random`, `urllib`, `json` vs.
- `pip` aracı ile Python Repositoryden paket indirilir.

```
Terminal
> pip install paketAdi
> pip show paketAdi
> pip list
> pip uninstall paketAdi
```

- Paket içeriği `import` ile çalışma ortamına çağırılır.

```
import paketAdi1, paketAdi2 as takmaAd

import paketAdi.modulAdi
kullan = paketAdi.modulAdi.nesneAdi

from paketAdi1.modulAdi import nesneAdi

from paketAdi import modulAdi
kullan = moduleAdi.nesneAdi
```

Değişkenler ve Standart Veri Tipleri

- Değişken Tanımlama: Python Dynamic Type'dır

- değişken = "metinsel değer"
- değişken = 2025
- del değişken #siler

- Standart Veri Tipleri:

- Primitive Tipler:
 - `i = 1` #int(1)
 - `f = 1.23` #float(1.23)
 - `c = 3+4j` #complex(3, 4)
 - `b = True` #bool(True)
 - `n = None` #NoneType
- Mutable (Değiştirilebilir)
 - list, set, dict
- Immutable (Sabit)
 - int, float, complex, bool
 - str
- Nesneleri keşfet
 - help(nesne)
 - type(nesne)
 - dir(nesne)

- Container Tipler:

- Sequence Tipler: Elemanları sıralıdır.
 - `str()`:
 - `m = "metin"` #çift veya tek tırnak ile
 - `m = '''çok satırda yazılmış metin'''` #tırnaklardan üçer tane.
 - `tuple()`: Değiştirilemez gruplar için kullanılır.
 - `t = (1,2,False,True, 'metin', None)`
 - `t = 1,2,False,True, 'metin', None`
 - `list()`:
 - `l = [1,2,False,True, 'metin', None]`
- Set Tipler: Sıra garantisiz ve tekrarsız tutar.
 - `set()`: Elemanlar immutable olmalıdır.
 - `s = {0,1,1,1}` #0,1
 - `frozenset()`:
 - `fr = frozenset({0,1,1,1})` #immutable settir.
- Mapping Tip: key:value şeklinde tutar.
 - `dict()`: keyler immutable olmalıdır.
 - `d = { "ad": "ali", "yas":33}`

Container Elemanlarına Erişmek

- Sequenceler (str, list, tuple): `Lst = [0,1,2,3,4]`
 - index Yönetim: `Lst[index]`
 - `lst[0]` #0
 - `lst[-1]` #4
 - slice Yöntem: `Lst[startIndex:stopIndex:step]`
 - `lst[1:4:2]` #1,3
 - `lst[::1]` #4, 3, 2, 1, 0
- Mapping (dict): `p={"ad":"Ali", "yas":33}` için
 - `p["ad"]` #Ali #key yoksa hata verir.
 - `p.get("yas")` #33 #key yoksa None döner.

Operatörler

Aritmetik Operatörler	Örnek	Sonuç
+, -, *, /	Dört işlem	8/2*(2+2)-1
%	Mod alma	5%2
**	Kuvvetini alma	3**2
//	Kalansız bölme	5//2
Atama Operatörleri	Örnek	Aynı
=	Atama, zincir atama	x=3
+=	Kümülatif aritmetik	x+=3
Karşılaştırma Operatörleri	Örnek (x=0 için)	Sonuç
==	Eşit	x==0
!=	Eşit değil	x!=0
>, <, >=, <=	Sırayla küçük, büyük, büyük eşit, küçük eşit	x>0
Mantıksal Operatörleri	Örnek (x=0 için)	Sonuç
and	Her iki taraf True ise sonuç True	x+1 and x<5
or	Herhangi bir taraf True ise sonuç True	x>1 or x<5
not	Sağdaki mantıksal durumun tersi	not(x==0)
Üyelik Operatörleri	Örnek	Sonuç
in	Soldaki sağdaki bir üyesi	1 in [1,2,3]
not in	Soldaki sağdaki bir üyesi değil	5 not in [1,2]
Kimlik Operatörleri	Örnek (x = yiçin)	Sonuç
is	Soldaki ile sağdaki aynı nesne	x is y
is not	Soldaki ile sağdaki aynı nesne değil	x is not y

Tarih ve Zaman Tipleri

- from datetime import date, datetime, timedelta

```
bugun = date.today()
bugun.year #2025
```

```
simdi = datetime.now()
simdi.date() #2025-09-17
simdi.date().month #9
```

```
simdi.time() #11:05:40.591238
simdi.time().hour #11
```

```
#-----tarih oluşturun-----
#1: tip üzerinden
t = date(year=2025, month=9, day=17)
```

```
#2: strptime ile. %Y yıl, %m ay, %d gün
t=datetime.strptime("16-09-2025", "%d-%m-%Y")
# strftime ile tarih formatını belirtilir.
t.strftime("%A %d. %B %Y")
#Wednesday 17. September 2025
```

```
#3: replace ile tarihi güncelle.
t.replace(day = 17)
```

```
#4. şimdiye 3 tane 7 gün ekle
datetime.now() + timedelta(days=7) * 3
```

Tiplere Ait Bazı Fonksiyonlar

list	Örnek: l = list([1,2,3]) olsun.	Sonuç
append	l.append(1000,1001)	[1,2,3,(1000,1001)]
extend	l.extend([1000,1001])	[1,2,3,1000,1001]
insert	l.insert(0,-99)	[-99,1,2,3]
remove	l.remove(2)	[1,3]
pop	l.pop()	[1,2]
sort	l.sort()	[1,2,3]
tuple	Örnek: t = 1,2,3 olsun.	Sonuç
add	t.add((4,5,6))	(1,2,3,4,5,6)
count	t.count(3)	1
index	t.index(1)	0
set	Örnek: a={1,2,3,"dört"} ve b={3,4,5,6}	Sonuç
add	a.add(5)	{1,2,3,"dört",5}
remove	a.remove("dört")	{1,2,3,5}
update	a.update({4,5})	{1,2,3,"dört",4,5}
union	a.union(b)	{1,2,3,"dört",4,5,6}
intersection	a.intersection(b)	{3,4}
difference	a.difference(b)	{1,2,"dört"}
dict	Örnek: p = {"ad":"Ali", "yas":33}	Sonuç
keys	p.keys()	['ad', 'yas']
values	p.values()	['Ali', 33]
items	p.items()	[('ad', 'Ali'), ('yas', 33)]
get	p.get("ehliyet", "key mevcut değil")	key mevcut değil
str	Örnek	Sonuç
upper	str.upper("merhaba")	"MERHABA"
title	str.title("merhaba dünyalı")	"Merhaba Dünyalı"
strip	str.strip(" merhaba ")	"merhaba"
split	str.split("merhaba dünyalı".split(" "))	["merhaba", "dünyalı"]
join	str.join(["Ali", "veli", "Ayşe"])	"AliveliAyşe"
replace	str.replace("merhaba", "selam")	"selam"
isnumeric	str.isnumeric("2024")	True
isalpha	str.isalpha("1234567890")	False
format	str.format("Ad: {}, Maas: {:.1f}".format("Ali", 1234.49))	"Ad:Ali Maas:1234.5"
f	f"Ad: {ad} Maas: {maas:.1f}"	"Ad:Ali Maas:1234.5"

Metni biçimlendirme üzerine: Yer tutucuları için [indexVeyaAlas: formatString](#) kalıbını kullanıyoruz.

indexVeyaAlas içinde size numarası verilmiş format fonksiyonunun parametre sırası, takma ad ve veritise format fonksiyonundaki eşleştirme dikkate alınır.

formatString ifadesi için aşağıdaki kalıp kullanılır. Veri tipi belirtirken str için s, float için f, int için d kullanılır. hizalamada `!align` başlatıcı karakter | hizala (<=) hizalanacakAlan | belirtilenAyarlar | basamakVetip

Örn: `Ad: (0,3s)!maas: (1;->30,1f);format(ad,maas)`

(0,3s) ile 0. yani ilk parametreye geçen metinsel ad değişkeni değerinden 3 harf al.

(1;->30,1f) ile 1. yani ikinci parametreye geçen sayısal maas değişkenini. 1f ile 1 basamak ondalıklı yaz.

Belitlik ayarç. olsun. 30 karakter içinde 9 ile sağa hizala. kalan karakterlere - ifadesini yazdır.

Genel

Örnek: l = [0,1,2,3]

Sonuç

max

min

sum

len

Tip Dönüşümü

- Tip Dönüşümü: İki tür dönüşüm mevcut.
 - implicit (Örtülü): Dönüşüm otomatiktir.
 - Örn: `1+1.2 = 2.2` yani `int + float = float`
 - Explicit (açık): Dönüşüm yapılmazsa hata verir.
 - Örn: `7 + "kişi"` yani `int + str` sonuç **HATA**
- #kullanıcıdan bilgi al. Metinsel tiptedir.
- `s1 = input("birinci sayıyı giriniz:")` #10
- `s2 = input("ikinci sayıyı giriniz:")` #20
- `print(s1+s2)` #1020 yani yanyana yazdı.

```
# Girilen verileri int tipine dönüştürelim.
s1 = int(s1)
s2 = int(s2)
print(s1+s2) #30 yani sayıları topladı.
```

Karar Yapıları ve Döngüler

- Karar Yapıları:
 - Ternary: Şart True ise if Şart else Şart False ise
 - x = 6
 - "Çift Sayıdır" if x%2==0 else "Tek Sayıdır"
 - if Yapısı: İlk şarta uyan blok. Yoksa else çalışır.
 - if x>10:
 - print("çift haneli sayı")
 - elif x in [0,2,4,6,8]:
 - print("Tek haneli pozitif çift sayı")
 - elif x>0 and x%2!=0:
 - print("Tek haneli pozitif tek sayı")
 - else:
 - print("Negatif sayı")
 - match: desen eşleştirme yapar.
 - deger = 'apple'
 - match deger:
 - case 'apple':
 - print("Elma seçildi.")
 - case 'banana':
 - print("Muz seçildi.")
 - case _:
 - print("Bilinmeyen meyve.")

- Döngüler:
 - for: itarable grup elemanlarını tek tek gezer.
 - çiftKare = [] #boş liste
 - for i in range(5): #range [0,1,2,3,4] tutar
 - if i%2==0:
 - çiftKare.append(i) #çiftler listeye

- !! alternatif olarak list comprehension yöntemi
- çiftKare = [i**2 for i in range(5) if i%2==0]
- #sonuç: Bir liste döner. [0, 4, 16]
- while-else: Şart sağlandığı sürece döngü çalışır. sayac = 0
- while sayac<10:
 - sayac +=1 #her iterasyonda 1 artsın
 - #-----
 - if sayac == 7:
 - break #döngüyü sonlandırır
 - if sayac%2==0:
 - continue #sonraki iterasyona geçer
 - #-----
 - print(sayac)

```
else:
    print(f"Şarta uymayan değer {sayac = }")
```

Çalışma Zamanı Hatalarını Yakalamak

- x = 12
- y = 0 #sayı ve "metin" gir
- sonuc = ""
- #try'daki hatalar exceptlerde yönetilir.
- try:
 - sonuc = x/y #hata olması muhtemel kodlar
- except ZeroDivisionError:
 - print("Bölüm sıfır olamaz.")
 - sonuc = "Sonsuz"
- except Exception as ex: #veya except:
 - print("Sistem hata mesajı: ", ex)
 - sonuc = "Beklenmedik hata"
- else: #else hata olmazsa çalışacak blok
- print("İşlem hatasız şekilde tamamlandı.")
- finally: #hata olsa da olmasa da çalışır
- print(f"İşlem tamamlandı. {sonuc = }")

Fonksiyonlar

- Klasik Fonksiyonlar: Çok satırda işlem yapılabilir.
- def c2f(c):
 - f = 1.8*c + 32
 - return f
- Lambda: Tek satırlı işler ve tek kullanımlık yerler.
- #lambda parametreler : returnEdilecekDeğer
- c2f = lambda c: 1.8*c + 32