1. Merhaba Dünya(Hello World)

Bilgisayarınızda programlama dilleri için gerekli yazılımları kurduysanız kurduğunuz yazılımları denemek için ekrana "Merhaba Dünya" veya "Hello World" yazdıran kodu çalıştırabilirsiniz. "Hello World" yazdıran kodu çalıştırmak üzerinde çalıştığınız programlama dilinin kod satırları hakkında size ön bilgi verebilir.

"https://helloworldcollection.github.io/" Sitesinde programlama dilleri için "Hello World" kodu bulunmaktadır.

1.1. Hello World Örnekleri

Aşağıda çeşitli programlama dilleri için ekrana nasıl "Hello World" yazdırılacağı görülmekte.

C dilinde "Hello World"
```c
#include

int main(void)
{
        puts("Hello, world!");
}
```

C# dilinde "Hello World"
```csharp
class HelloWorld{
    static void Main(){
        System.Console.WriteLine(
"Hello, World!");
    }
}
```

C++ dilinde "Hello World"
```cpp
#include

int main()
{
    std::cout << "Hello, world!";
    return 0;
}
```

Java dilinde "Hello World"
```java
class HelloWorld {
  static public void main( String
args[] ) {
    System.out.println(
"Hello World!" );
  }
}
```

PHP dilinde "Hello World"
```php
<?php echo '<p>Hello World</p>';?>
```

Swift dilinde "Hello World"
```swift
println("Hello, world!")
```

JavaScript dilinde"Hello World"
```javascript
document.write('Hello, world!');
```

Python dilinde "Hello World"
```python
print("Hello, world!")
```

2. Yorum Satırları

Program içerisinde kodlar üzerinde açıklama yapmak gerekiyorsa yorum satırları kullanılmalıdır. Yorum satırları Python tarafından görülmez ve çalıştırılmaz. Çok yorum satırı kullanmak yazdığınız kodun daha karmaşık olmasına yol açar. Fakat hiç yorum satırı kullanmadan kod yazarsanız kodunuzda ne yapmak istediğiniz zor anlaşılır.

```
>>> # Tekli Yorum Satırı

>>> """ Çoklu yorum satırı
içinde istediğiniz kadar
satır kullanabilirsiniz """
```

3. Temel Veri Tipleri ve Değişkenler

3.1. Basit Veri Tipleri

3.1.1. Tam Sayı (int) Veri Tipi

Pozitif ve negatif sayıları içeren kesirsiz ve ondalıksız sayıların tamamı tam sayılardır. Integer tam sayı tiplerinin İngilizcesi olduğunda Python' da Integer ifadesi kısaca int olarak kullanılır. Örnek: 32, -41, 0, 10 -190, 1923.

3.1.2. Ondalıklı Sayı (float) Veri Tipi

Ondalıklı sayılar matematikte olduğu gibi iki tam sayının birbirine oranı ile ifade edilebilen sayılardır. Ondalıklı Python 'da "float" olarak geçmektedir. Örnek: 3.14, -1.22, 7.05, 3.0, -5.96323, -4.0, 0.17,-0.23.

3.1.3. Basit Matematik İşlemleri

```
>>> # Toplama
>>> 3 + 5
8      → int

>>> #Çıkarma
>>> 7-12
-5     → int

>>> #Bölme
>>> 5/2
2.5    → float

>>> #Çıkarma
>>> 13-5
8      → int

>>> #Çarpma
>>> 17 * 3
51     → int

>>> #Bölme
>>> 6/2
3.0    → float
```

Yukarıda 4 işlem örneği verildi. Python 'da bölüm her zaman float çıktığı için son işlemin sonucu float çıkar.

3.1.4. Karakter Verileri (string - str)

Tırnak içerisinde yazılan her ifade karakter verisi olarak algılanır. Python' da " ", ' ' veya """ """ tırnak sembolleri kullanılabilir. """ """ birkaç paragraftan oluşan karakter verileri için kullanılır. Karakter verileri aritmetik işlemlere giremez. Python' da karakterler (string) ifadesi kısaca str olarak kullanılır. Örnek: "Merhaba", "Dünya", "12322"

```
>>> #Tek tırnak ile
>>> 'Tuna POTUR'
'Tuna POTUR'

>>> #Çift Tırnak ile
>>> "Tuna POTUR"
'Tuna POTUR'

>>> #Üç Tırnak ile
>>> """Tuna POTUR"""
'Tuna POTUR'

>>> """üç tırnak
bir den çok satırlı metinler için kullanılır"""
```

3.1.5. Alıştırmalar

Aşağıdaki işlemlerin sonuçlarını kendiniz hesaplayın ve aynı işlemi etkileşimli kabukta(IDLE) deneyin.

3 + 5	15.3 + 4.7	5 – 2*(7 – 12) + 6/3
17 – 2 * 4 -5 + 3	4 * (1/2)	5*3-7*2+(6/3) -2*4*(0.5)
18.5 – 3.2	18 * (0.5)	((3 + 5) * 4 - (9 - 5) * 3) / 5

3.2. Değişkenler ve Değişken Tanımlama

Bir değişken bir değer taşıyan isimdir. Değişkenler programın çalışması sırasında gerekli olan değerlerin değiştirilmesi ve güncellenmesi amacıyla bellekte depolanması için kullanılır. Diğer programlama dillerinin aksine Python' da değişkenleri önceden tanımlamanız gerekmez. Bir değişkene bir değer atamak için atama ifadesi olan eşittir (=) operatörü kullanılır.

```
>>> # Değişken ismi ve Değişkenin değeri
>>> yaş = 16
>>> boy = 1.85
>>> adı = "Can"
>>> print("Adınız:",adı,"\n","Yaşınız:",yaş,"\n","Boyunuz:",boy,"\n")
 Adınız: Can
 Yaşınız: 16
 Boyunuz: 1.85
```

Değişkenler taşıdıkları verinin tipindedir.

Örnekte görüldüğü gibi değişkenlere hangi tipte değer verildiyse değişken o tipte oluşturulmuştur.

- yaş değişkenine 16 tam sayısı girildiği için yaş değişkeni int tipinde oluşturuldu.
- boy değişkenine 1.85 kesirli sayısı girildiği içi boy değişkeni float tipinde oluşturuldu.
- adı değişkenine "Can" karakter değeri atandığı için adı değişkeni str*(string)* tipinde oluşturuldu.

print()

Fonksiyonlar konusunu ilerde detaylı bir şekilde göreceğiz ama bazı fonksiyonları şimdiden tanımamız gerekiyor. print() *fonksiyonu parantez içine girilen değeri ekrana yazdırılır.*

print("Adınız:",adı,"\n","Yaşınız:",yaş,"\n","Boyunuz:",boy,"\n") *ifadesinde*

"\n" *bir satır alta geçmek için kullanıldı.*

"Adınız:", "Yaşınız:" *ve* "Boyunuz" *karakter verileri doğrudan ekrana yazdırılmak için fonksiyona girildi.*

adı, yaş *ve* boy *değişkenleri içinde tutulan değerlerin ekrana bastırılması için fonksiyona girildi.*

3.2.1. Değişken Tiplerinin Öğrenilmesi ve Değişken Tipinin Kod Akışı İçerisinde Değiştirilmesi

type()

Değişkenlerin tipleri, o anda hangi değeri taşıyorsa o değerin tipi ile aynıdır. Değişkenlerin tiplerini öğrenmek için type() fonksiyonu kullanılır. type(değişken) şeklinde çalıştırılan type() fonksiyonu bize değişkenin hangi tipte olduğunu döner.

```
>>> type(16)
<class 'int'>

>>> type(1.85)
<class 'float'>

>>> type("Can")
<class 'str'>
```

`type(16)` işlemiyle 16 değerinin **`int`**
`type(1.85)` işlemiyle 1.85 değerinin **`float`**
`type("Can")` işlemiyle **`"Can"`** değerinin **`str`** olduğu program içerisinde öğrenilebilir.

```
>>> yaş = 16
>>> type(yaş)
<class 'int'>
>>> boy = 1.85
>>> type(boy)
<class 'float'>
>>> adı = "Can"
>>> type(adı)
<class 'str'>
```

`type(yaş)` işlemiyle yaş değişkeninin **`int`**
`type(boy)` işlemiyle boy değişkeninin **`float`**
`type(adı)` işlemiyle adı değişkeninin **`str`** olduğu program içerisinde öğrenilebilir.

```
>>> yaş = 16
>>> type(yaş)
<class 'int'>

>>> yaş="on altı"
>>> type(yaş)
<class 'str'>
```

```
>>> yaş = 15.5
>>> type(yaş)
<class 'float'>

>>> yaş = 15
>>> type(yaş)
<class 'int'>
```

Bir değişkenin hem değeri hem de tipi kod akışı içerisinde değiştirilebilir.

- yaş değişkenine 16 değeri girilerek yaş değişkeni int tipinde tanımlandı.
- Sonradan yaş değişkenine "on aldı" değeri girilerek yaş değişkeni str tipine dönüştürüldü.
- Daha sonra yaş değişkenine 15.5 değeri atanarak yaş değişkeni float tipine dönüştürüldü.
- Son olarak yaş değişkenine 15 değeri atanarak yaş değişkeni int tipine dönüştürüldü.

Değişkenin program çalıştığı sürece belli bir anlamı ve rolü olduğu için tipi genellikle değişmez. Bu yüzden değişkenleri oluşturduğumuz tiple kullanmalıyız. Çok nadir durumlar dışında değişkenlerin tipini değiştirmekten kaçınmalıyız.

3.2.2. Değişkenlere İsim Verirken Dikkat Edilmesi Gereken Kurallar

1. Değişkene içerdiği değer ile tutarlı isimler veriniz.
2. Değişkenlere isim verirken boşluk kullanmayınız.
3. Değişken isimleri sayı ile başlayamaz. Değişken isimleri bir karakter ile başlamalıdır.
4. Python3 de değişken adı olarak kullanabileceğimiz karakterler:
   ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyzÇĞİÖŞÜçğıöşü_0123456789
5. Programlama dillerinde kullanılan komut isimleri değişken olarak kullanılamaz.

Python' da Değişken Olarak Kullanılamayan Komut İsimleri								
and	as	assert	break	class	continue	def	del	elif
else	except	False	finally	for	from	global	if	import
in	is	lambda	None	nonlocal	not	or	pass	raise
return	True	try	while	with	yield			

6. Değişken isimlendirmelerinde boşluk karakteri yerine alt çizgi (_) karakteri kullanılabilir.
   Örnek: tc_Kimlik_No
7. Değişken isimleri kelimeler arasına boşluk koymadan ilk kelimeden sonra gelen kelimelerin baş harfleri büyük yazılarak oluşturulabilir. Buna "Camel Karakter" kullanımı denir.
   Örnek: tcKimlikNo
8. +-*/%:;'"<>?|\()!@#$%^&~ gibi özel karakterler değişken ismi içinde kullanılamaz.
   (Sadece "_" sembolü kullanılabilir).
9. Değişken isimleri büyük küçük harf duyarlılığa sahiptir.

```
>>> TELEFON = "417 00 01"
>>> telefon = "415 11 12"
>>>
>>> TELEFON
'417 00 01'
>>> telefon
'415 11 12'
```

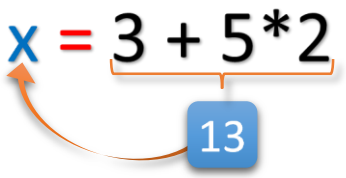
TELEFON = "417 00 01" ve telefon = "415 11 12" iki ayrı değişkenlerdir.

Hatalı Değişken İsimleri Örnekleri

1sayı, ali@veli, +yaş, cc.com, Okul No, soru?, ara-toplam, and, print…

3.2.3. Matematik İşleminin Sonucunu Değişkene Atama (Önce İşlem Sonra Atama)

```
>>> #Önce işlem sonra atama
>>> x = 3 + 5*2
>>> print("X değeri :",x)
X değeri : 13
```



x değişkenine değer atamadan önce 3 + 5*2 işlemi yapılmış. Python 'da eğer değişkene değer atanmadan önce işlem yapıldıysa, önce eşittir operatörünün sağındaki işlem yapılır sonra sonuç değişkene atanır.

3.2.4. Değişkenlerin Değerini Arttırma

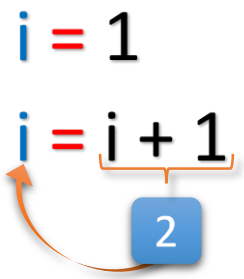
```
>>> #Önce işlem sonra atama
>>> x = 5
>>> x = x*2 + 3
>>> print("X değeri :",x)
X değeri : 13
```

Burada sizce bir matematik kuralını mı ihlal ettik? Sorusu akla gelebilir. Eşittir operatörünün kullanımını düşünürsek önce sağ taraftaki işlem yapılacak sonra sonuç x değişkenine atanacaktır.

```
>>> #Önce işlem sonra atama
>>> #x' in değerini 1 arttırma
>>> x = 13
>>> x = x + 1
>>> print("X değeri :",x)
X değeri : 14
```

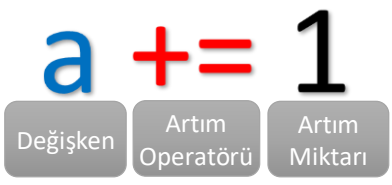
x = x + 1 işlemiyle x değişkenini değeri 1 artar. Değişkenlerin değerlerini kendisiyle 1'i toplayarak arttırmak ileride göreceğimiz döngüler konusunda çok sık kullanılan bir yöntemdir.

```
>>> i = 1
>>> i
1
>>> i = i + 1
>>> i
2
>>> i = i + 1
>>> i
3
```



Bu kod örneğinde görüldüğü gibi i = i +1 işlemi yapmak değişkenin değerini 1 arttırılıyor.

```
>>> a = 1
>>> a
1
>>> a+=1
>>> a
2
>>> a+=1
>>> a
3
```



i = i +1 yerine Python 'da aynı işlemi yapmak için i+=1 ifadesi kullanılır. i+=1 ile i = i +1 aynı işi yapar fakat i+=1 yazmak daha anlaşılır ve okunabilir olduğu için tercih edilir. i+=1 işleminde += operatörü i değişkeninin artırılacağını, 1 de i değişkenin değerinin birer artırılacağını belirtiyor.

```
>>> b = 6
>>> b-=2
>>> b
4
>>> b-=2
>>> b
2
```

Aynı mantıkla değişkenlerin değerleri azaltılabilir. b -= 2 işlemi b değerini 2 azaltır.

```
>>> c = 4
>>> c*=3
>>> c
12
>>> c*=3
>>> c
36
```

c*=3 işlemi c değeri 3 ile çarparak arttırır.

3.2.5. İki Değişkenin Değerini Birbiriyle Değiştirme

```
>>> a = 3
>>> b = 5
```

a ve b değişkenlerinde bulunan değerleri bir biriyle yer değiştirmek gerekebilir.

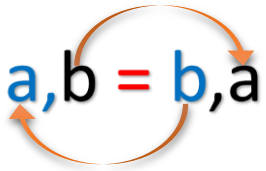
```
>>> a = b
>>> a
5
>>> b
5
```

a = b gibi bir eşitlik yaparsak b değişkeninde bulunan değer a değişkenine geçecektir ve a değişkeni b değişkeni ile aynı değere sahip olacaktır.

```
>>> a = 3
>>> b = 5
>>> takas = a
>>> a = b
>>> b = takas
>>> a
5
>>> b
3
```

İki değişkenin değerini birbiriyle değiştirmek için takas değişkenine a değişkeni atanır, a değişkenine de b değişkeni atanır. Son olarak b değişkenine takas değişkeni atanır.

```
>>> a = 3
>>> b = 5
>>> a,b = b,a
>>> a
5
>>> b
3
```



Python takas işlemi yapmak için bize kolay bir yol sunar. a,b = b,a yapmak iki değişkenin değerlerinin bir biriyle değiştirilmesini sağlar.

3.2.6. Aynı Değere Sahip Değişkenler Tanımlama

```
>>> # kod içerisinde aynı değere sahip değişkenler tanımlama
>>> a = 5
>>> b = 5
>>> c = 5
>>> print("a =",a,"b =","c =",c)
a = 5 b = c = 5
```

Yazdığınız kod içerisinde aynı değere sahip değişkenler tanımlamaya ihtiyaç duyabilirsiniz. Değişkenlere tek tek aynı değeri vererek tanımlayabilirsiniz.

```
>>> # kolay yoldan aynı değere sahip değişkenler tanımlama
>>> a = b = c = 9
>>> print("a =",a,"b =","c =",c)
a = 9 b = c = 9
```

Python' da a = b = c = 9 şeklinde tanımlaya yaparak aynı değere sahip değişkenleri bir seferde tanımlayabilirsiniz.

```
>>> ocak = mart = mayıs = temmuz = ağustos = ekim = aralık = 31
>>> nisan = haziran = eylül = kasım = 30
>>> şubat = 28
```

Yıl içerisinde aynı değere sahip birçok ay bulunmakta. Hangi ayın kaç gün çektiğini değişkelere tanımlamak için aynı değere sahip değişkenleri üstte görüldüğü gibi kolay bir şekilde tanımlayabilirsiniz.

3.2.7. Tip Dönüşümleri

3.2.7.1. Otomatik Tip Dönüşümleri

İki aynı sayı tipiyle bölme hariç işlem yapıldığında işlemin sonucu işlemin yapıldığı veri tipinde olacaktır.

```
#iki int sayı toplamı
>>> 3 + 5
8
>>> type(3 + 5)
<class 'int'>
```

İki int sayı arasında yapılan işlemin sonucu yine int tipinde olacaktır.

```
#float iki sayı toplamı
>>> 2.25 + 3.75
6.0
>>> type(2.25 + 3.75)
<class 'float'>
```

İki float sayı arasında yapılan işlemin sonucu tam sayı çıksa bile sonucun tipi her zaman float olur.

```
#int ve float iki verinin toplamı
>>> type(2)
<class 'int'>
>>> type(3.14)
<class 'float'>
>>> 2 + 3.14
5.140000000000001
>>> type(2 + 3.14)
<class 'float'>
```

float ve int sayı arasında işlem yapılırsa sonuç her zaman float sayı tipinde olur.

```
#int ve float iki değişkenin toplamı
>>> a = 12
>>> type(a)
<class 'int'>
>>> b = 23.5
>>> type(b)
<class 'float'>
>>> a + b
35.5
>>> type(a+b)
<class 'float'>
```

int ve float değişkenlerin toplamı yine float tipinde olacaktır.

```
#float iki değişken toplamı
>>> a = 12.3
>>> b = 23.7
>>> a + b
36.0
>>> type(a + b)
<class 'float'>
```

float tipinde iki değişkenin toplamı sonucu tam sayı olsa bile sonuç yine float tipinde olacaktır.

Görüldüğü gibi float ve int tipinde veriler toplandığında sonuç her zaman float çıkar. Aslında programlama dillerinde iki farklı tipte veri birbiriyle işleme giremez. Fakat birçok programlama dilinde olduğu gibi Python' da tip dönüşümünü otomatik olarak yapar. Python int sayı tipindeki veriyi arka planda float sayı tipine çevirir ve işlemi float sayılar arasında yapar. Özetle float ve int sayı tipleriyle işlem yapıldığında int sayı tipleri float sayı tipine çevrildiği için sonuç her zaman float çıkar.

```
#sonucu tam sayı çıkan bölme işlemi
>>> 12 / 4
3.0
>>> type(12 / 4)
<class 'float'>
```

Python tip güvenliğini sağlamak için bölme işleminin sonucu int çıksa bile sonuç her zaman float çıkar. Bunun sebebi bölme işlemlerinin sonuçlarının çoğunlukla float çıkmasından kaynaklanmaktadır.

3.2.7.2. İsteğe Bağlı Tip Dönüşümleri

İşlemler içerisinde otomatik tip dönüşümleri olduğu gibi isteğe bağlı olarak tip dönüşümü yapılabilmektedir.

int(), float(), str()

Bu fonksiyonlar değer olarak aldıkları verileri kendi tiplerine dönüştürür.

int(3.14) → 3 tam sayı

float(5) → 5.0 ondalıklı sayı

str(12) → '12' karakter verisi

```
>>> # ondalıklı sayıyı tam sayıya çevirme
>>> int(3.14)
3
>>> pi = 3.14
>>> int(pi)
3
```

Ondalıklı sayıyı tamsayıya çevirmek için int() fonksiyonu kullanılır. float 'dan int sayıya çevrim yapıldığında ondalıklı sayının tam kısmı değer olarak kullanılır.

```
>>> # tamsayıyı ondalıklı sayıya çevirme
>>> float(7)
7.0
>>> float(-9)
-9.0
>>> a = 53
>>> float(a)
53.0
```

Tam sayıyı ondalıklı sayıya çevirmek için float() fonksiyonu kullanılır. Tam sayıların kesir hanesi olmadığı için tam sayılar ondalıklı sayıya dönüştürüldüğünde 5.0 gibi bir değer oluşur.

```
>>> # tam ve ondalıklı sayıyı string' e çevirme
>>> str(47)                                    >>> str(3.141592)
'47'                                           '3.141592'
>>> a = 63                                     >>> pi = 3.141592
>>> str(a)                                     >>> str(pi)
'63'                                           '3.141592'
```

Sayıları karaktere çevirmek için str() fonksiyonu kullanılır (*str, String*' in kısaltmasıdır). str() fonksiyonuyla sayıyı oluşturan tüm rakamlar veya nokta birer karaktere dönüşecektir.

```
# string veriyi tam ve ondalıklı sayıya çevirme
>>> int("93")                                  >>> float("3.141592")
93                                             3.141592
```

Bir string'i, tamsayıya çevirmek için int() fonksiyonu, ondalıklı sayıya çevirmek için float() fonksiyonu kullanılır.

```
>>> # sayısal ifadenin yanında karakter kullanılıyor
>>> int("ahmet1981")
Traceback (most recent call last):
  File "<pyshell#8>", line 1, in <module>
    int("ahmet1981")
ValueError: invalid literal for int() with base 10: 'ahmet1981'
>>> int("1981") # tip dönüşümü hatasının düzeltilmesi
1981

>>> # karakter ifadesinin içinde iki kere nokta karakteri kullanılıyor
>>> float("3.14.324324")
Traceback (most recent call last):
  File "<pyshell#10>", line 1, in <module>
    float("3.14.324324")
ValueError: could not convert string to float: '3.14.324324'
>>> float("3.14324324") # tip dönüşümü hatasının düzeltilmesi
3.14324324
```

Karakterler sayısal veriye çevrileceği zaman tip dönüşümü yapılacak verinin sayısal karşılığı olması gerekir. string 'den int' e çevrilirken oluşan hatanın nedeni sayısal ifade ile sayısal olmayan karakter kullanılmasıdır. string 'den float' a çevrilirken oluşan hatanın sebebi karakter ifadesinin içinde iki kere nokta kullanılmasıdır.

```
>>> # string verilerin sayısal veriye çevrilerek işleme sokulması
>>> 71 + int("17")                             >>> 12 + float("2.17")
88                                             14.17
>>> int("28") + 37                             >>> 19.23 + float("-5.5")
65                                             13.73
```

Bu örnekte int ve float sayı değerlerinin string tipli değerler ile matematiksel işlemleri görülüyor. string veriler tip dönüşümüyle sayısal verilere dönüştürülerek işlem yapılabilmiş.

```
>>> # string ve int değişkenlerin işleme girmesi
>>> a = 9
>>> b = "11"
>>> c = 3
>>> a + b + c
Traceback (most recent call last):
  File "<pyshell#15>", line 1, in <module>
    a + b + c
TypeError: unsupported operand type(s) for +: 'int' and 'str'
>>> a + int(b) + c
23
```

Bu örnekte string b değişkeni ve int. a ve c değişkenleri ile a + b + c işlemi yapılmış. int ve string değişkenler birlikte işleme giremeyecekleri için tip hatası oluşur. Bu hatanın çözümü için int(b) işlemiyle b değişkenin değeri int'e dönüştürülür.

input()

Fonksiyonu kullanıcıdan değer almak için kullanılır. Kullanıcıya mesaj vermek için input fonksiyonuna değer girilir. input fonksiyonu kullanıcının girdiği bilgiyi string olarak döner.

```
>>> input()
33
'33'
>>> input("Bir Değer Giriniz:")
Bir Değer Giriniz:47
'47'
```

input() fonksiyonu işletildiği anda ekranda beliren imleç bilgi girilmesini bekler.

```
>>> a = input("Değer Giriniz:")
Değer Giriniz:85
>>> a
'85'
>>> type(a)
<class 'str'>
```

Kullanıcıdan aldığınız değeri bir değişkene atamazsanız program içerisinde kullanıcıdan gelen değer kullanılamaz. input fonksiyonuna girilen değer kullanıcıya değer girmesi için yönlendirici nitelikte mesaj verecektir.

```
>>> a = input("Değer Giriniz:")
```

İşlemiyle kullanıcıdan alınan string değer a değişkenine atanır.

```
>>> a = input("Değer Giriniz:")
Değer Giriniz:44
>>> print("Kullanıcının Girdiği Değer:", a, "Değerin Tipi:",type(a))
Kullanıcının Girdiği Değer: 44 Değerin Tipi: <class 'str'>
>>> b = input("Değer Giriniz:")
Değer Giriniz:ahmet1981
>>> print("Kullanıcının Girdiği Değer:", b, "Değerin Tipi:",type(a))
Kullanıcının Girdiği Değer: ahmet1981 Değerin Tipi: <class 'str'>
```

Görüldüğü gibi input fonksiyonu kullanıcıdan string tipinde veri alır.

```
>>> a = input("Değer Giriniz:")
Değer Giriniz:8
>>> a + 2
Traceback (most recent call last):
  File "<pyshell#7>", line 1, in <module>
    a + 2
TypeError: Can't convert 'int' object to str implicitly
>>> type(a)
<class 'str'>
```

input fonksiyonu kullanıcıdan sting tipte değer alır. Kullanıcıdan aldığınız değeri doğrudan bir değişkene atarsanız değişkenin tipi string olacaktır. Bildiğiniz gibi string tipte değişken ile float ve integer değişken işleme giremez.

```
>>> a = int(input("Değer Giriniz:"))
Değer Giriniz:8
>>> a + 2
10
```

Kullanıcıdan aldığınız değerin int ve float sayılarla işleme girmesi için input fonksiyonunu int fonksiyonun içinde kullanmanız gerekir. Böylelikle kullanıcıdan aldığınız string değer int fonksiyonuyla tam sayıya dönüştürülür. Tam sayıya dönüştürülen kullanıcı verisi tam sayılarla işleme girebilir.

```
>>> r = float(input("Çemberin Yarıçapını Girin:"))
Çemberin Yarıçapını Girin:5.25
>>> alan = 3.14 * r * r
>>> print("Çemberin Alanı:",alan)
Çemberin Alanı: 86.54625
```

Bu örnekte kullanıcı çemberin yarıçapını float değer olarak giriyor. float sayılar arasında işlem yapmak için input fonksiyonu float fonksiyonunun içinde kullanılmış. Böylelikle kullanıcının input ile girdiği string değer float fonksiyonuyla float değere dönüştürülüyor.

```
>>> c = int(input("Değer Giriniz:"))
Değer Giriniz:can1993
Traceback (most recent call last):
  File "<pyshell#21>", line 1, in <module>
    c = int(input("Değer Giriniz:"))
ValueError: invalid literal for int() with base 10: 'can1993'
```

Eğer kullanıcıdan int değer girmesini beklerseniz ve kullanıcı string değer girerse değer hatası oluşacaktır. Kullanıcının hatalı giriş yapmasını engelleyen "Hatalar ve İstisnalar" konusunu ilerde göreceğiz.

3.2.8. Örnekler

3.2.8.1. IDLE Yardımıyla Klavyeden Girilen İki Tam Sayının Toplamını

```
# girilen iki sayı ile yapılan işlemler
>>> birinciSayı = int(input("Birinci Sayıyı Girin: "))
Birinci Sayıyı Girin: 37
>>> ikinciSayı = int(input("İkinci Sayıyı Girin: "))
İkinci Sayıyı Girin: 43
>>> toplam = birinciSayı + ikinciSayı
>>> fark = birinciSayı - ikinciSayı
>>> çarpım = birinciSayı*ikinciSayı
>>> bölme = birinciSayı/ikinciSayı
>>> ortalama = toplam/2
```

```
>>> print("Toplam :",toplam)
Toplam : 80
>>> print("Fark :",fark)
Fark : -6
>>> print("Çarpım :",çarpım)
Çarpım : 1591
>>> print("Bölme :",bölme)
Bölme : 0.8604651162790697
>>> print("Ortalama :",ortalama)
Ortalama : 40.0
```

Bu alıştırmada kullanıcı klavyeden sırasıyla iki değer alarak alınan değerlerle işlem yapan bir uygulama yazdık. IDLE ile yaptığınız çalışmalarda her kod bir komutmuş gibi girilir. IDLE bize kod yazmak için gerçekçi bir kullanım sunmaz sadece çok kısa denemeleri yapma imkânı sunar.

3.2.8.2. Pycharm Yardımıyla Klavyeden Girilen İki Tam Sayının Farkı

Pycharm kod yazmak için özelleşmiş bir editör programıdır. Bu tip programlara ide(*Integrated Development Environment*)denir. Yazdığımız kodlar ".py" uzantılı dosyalarda saklanır. Kodları içeren Python dosyası çalıştırıldığında yazdığınız program da çalışacaktır. Pycharm Python kod dosyasını oluşturma ve bu dosyayı çalıştırmak için tüm araçları bize sunar.

3_2_7_2_IkiSayiFarki.py

```
# Girilen iki sayının farkını alma
print("Girilen İki Sayının Farkı(a - b)")
birinciSayı = int(input("Birinci Sayıyı Girin:"))
ikinciSayı = int(input("İkinci Sayıyı Girin:"))

fark = birinciSayı - ikinciSayı

print("Fark :", fark)

# Ekran çıktısı
"""
Birinci Sayıyı Girin:33
İkinci Sayıyı Girin:13
Fark : 20
"""
```

İki sayının farkı uygulamasında kodlarımız "3_2_7_2_IkiSayiFarki.py" dosyasında saklanır. Pycharm üzerinde hazırlanan kod dosyası çalıştırıldığında programımız çalışacak ve sizden iki tam sayı değeri alıp farklarını hesaplayarak size sunacaktır.

3.2.8.3. Kişilerin Yaşı

3_2_7_3_KisilerinYasi.py

```
# Kişilerin yaşını bulma
print("Belli Bir Yıla Göre Kişinin Yaşını Hesaplama")
doğumYılı = int(input("Doğum Yılını Girin:"))
hesapYılı = int(input("Hesaplama Yılını Girin:"))

yaş = hesapYılı - doğumYılı

print(yaş, " yaşında")
```

```
# Ekran çıktısı
"""
Doğum Yılını Girin:1881
Hesaplama Yılını Girin:1923
42  yaşında
"""
```

Bir kışının doğum yılını biliyorsanız yaşını hesaplamak için bir program kullanabilirsiniz. Üstte yazılan kodu Mustafa Kemal ATATÜRK 'ün Cumhuriyet ilan edildiğinde kaç yaşında olduğunu bulmak için kullandık.

3.2.8.4. Çemberin Çevresi ve Alanı

3_2_7_4_CemberinCevresiAlani.py

```
# Çemberin Çevresi ve Alanı
print("Yarı Çapı Girilen Çemberin Çevresini ve Alanını Bulma")
r = float(input("Çemberin Yarı Çapını Girin:"))
pi = 3.14159

print("Çemberin Çevresi:", 2*pi*r)
print("Çemberin Alanı  :", pi*r*r)

# Ekran çıktısı
"""
Çemberin Yarı Çapını Girin:4.12
Çemberin Çevresi: 25.8867016
Çemberin Alanı  : 53.326605296
"""
```

Çemberin yarı çapı biliniyorsa çevresi ve alanı bulunabilir. Çemberin alanını ve çevresini buluna kodda çemberin yarı çapı kullanıcıdan float olarak alınmış.

3.2.8.5. Girilen İki Sayının Ortalaması

3_2_7_5_GirilenIkiSayininOrtalamasi.py

```
# Kullanıcı tarafından girilen iki sayının ortalaması
a = float(input("Birinci Sayıyı Giriniz: "))
b = float(input("İkinci Sayıyı Giriniz : "))

print("İki Sayının Ortalaması:",(a + b) / 2)

# Ekran çıktısı
"""
Birinci Sayıyı Giriniz: 10.4
İkinci Sayıyı Giriniz : 30.6
İki Sayının Ortalaması: 20.5
"""
```

Bu örnekte kullanıcıdan alınan değerler a ve b değişkenlerine atandı. Ortalama hesabı print fonksiyonun içerisinde ekrana yazdırılırken (a + b) / 2 işlemi yapılarak yazdırıldı.

3.2.9. Alıştırmalar

3.2.9.1. IDLE Değişken Kullanılarak Yapılan Hesaplamalar

x = 3 için alttaki her işlemin sonucunda x değişkeninin son değerini IDLE kullanarak bulun.

1. x = 5
2. x = x +1
3. x = x*3 + 2
4. x = 2*x + 7
5. x = (x + 9)/3 + 4
6. x = x + 9/3 + 4
7. x +=1
8. x *=5
9. x = ((7 + x)/2)*4 + x

Altta yazan işlemleri IDLE kullanarak çözün.

1.

```
t = 6
y = 9
t = t + y +z

t = ?
```

2.

```
c = 6
a = c
t = 5
d = 4
t = d + a + t

t = ?
```

3.

```
a = 3
b = 2
c = 4
t = 1
t = a*b -c + t + 5

t = ?
```