Python Programlama

Ders 1

Ali Mertcan KOSE Msc.
amertcankose@ticaret.edu.tr

İstanbul Ticaret Üniversitesi

Python

Python 90'lı yılların başlarında Amsterdam'da Guido Van Rossum tarafından geliştirilmeye başlanan bir programlama dilidir. Zannedilenin aksine ismini piton yılanından değil, Rossum'un çok sevdiği MontyPtyhon isimli bir komedi grubunun sergilediği gösteriden alır. Ancak bu gerçeğe rağmen, oluşturduğu çağrışımdan dolayı Python dili pek çok yerde yılan figürü ile temsil edilir.

Python, nesne yönelimli, yorumsal, modüler ve etkilişimli, yüksek seviyeli bir dildir. Programlama dilleri makine mantığı ile insan mantığı arasında bir köprüdür. Bir dilin makine mantığına daha yakın olması makine mantığına yaklaşmak, insan mantığından uzaklaşmayı gerektirir ve haliyle dili öğrenmesi daha zor hale getirir. Makine mantığı düşük seviyede, insan mantığı ise yüksek seviyededir. Python insan mantığına daha yakın olması nedeniyle yüksek seviyeli bir dil olarak sınıflanır. Pek çok dile göre öğrenilmesi kolaydır. Daha önce programlama deneyiminiz olmamışsa Python sizin için ideal bir başlangıç olabilir.

Python

Python ile ihtiyaç duyduğunuz pek çok işi, az sayıda kod satırı yazarak yerine getirebilirsiniz. Masaüstü uygulamaları, web uygulamaları, finansal hesaplamalar, veri analizi ve görselleştirme uygulamaları gibi pek çok programı **Python** ile kolaylıkla yazabilirsiniz.

Neden Python ?

Python yorumsal bir dildir, java, C ve C++ gibi dillerin aksine derlenmeye gerek olmadan çalıştırılabilmektedir. Python içerisinde bir programı yazarken ihtiyaç duyacağınız pek çok şey, veri yapıları, fonksiyonlar hazır olarak size sunulmaktadır. Bu sayede diğer dillerde olduğu gibi bir problemi çözmek için en ince ayrıntılara kadar tasarım yapmanıza gerek kalmadan size sunulan altyapı ile çok daha seri bir şekilde program yazabilirsiniz.

Python basit bir söz dizimine sahiptir. Bu sayede hem program yazmak daha kolay ve keyifli hale gelir hem de başkalarının yazdığı programlar daha rahat anlaşılabilir. Python az kod ile çok şey yapmayı sağlar.

Neden Python ?

Python dili sahip olduğu avantajlar ile dünyaca ünlü pek çok kuruluşun ilgi odağıdır. Google, Youtube, Yahoo gibi kuruluşları Python programcılarına her zaman ihtiyaç duymaktadır. Python dilini geliştrien Russom, 2012 yılına kadar Google'da çalışmış daha sonra Dropbox şirketine transfer olmuştur. Bu durum, Python dilinin güncellik ve popülerliğine bir işarettir.

Python Dili ve Modülleri

Sınıftaki alıştırmalar ve örnekler Anaconda sürümünden Python'un en son versiyonu kullanılacaktır (python-3.11.5).

Python programını indirdikten sonra Visiual Studio code yazılımı kullanarak, uygulamalar yapılabilir. Diğer taraftan *Collab* ile google server'ı üzerinden herhangi bir program kurulumu yapmadan uygulamalar yapılabilir.

Dersimiz kapsamında Standart Python'a dahil olmayan ek paketlerde ele alınacaktır.

- `numpy` : Python'da bilimsel hesaplamalar için kullanılan pakettir.
- `matplotlib` : Python'da statik, animasyonlu ve etkilişimli görselleştirmeler oluşturmak için kullanılan kapsamlı bir kütüphanedir.

Tavsiye Edilen Yazılımlar

Dersimiz kapasamında visiual studio programı üzerinden çalışmalar *Python-3.11.5* ile yapılacaktır.

Visiual Studio içinde;

- **Python:** Bilgisayar programlarını yazdığımız bir programlama dilidir.
- **Python Paketleri:** Bilimsel hesaplama ve hesaplamalı modelleme için, Pythonın standart kütüphanesinin bir parçası olmayan kütüphanelere (paketlere) ihtiyacımız vardır. Bunlar grafik oluşturmamıza, matrisler üzerinden çalışmamıza ve özel sayısal yöntemler kullanmamıza izin verir. Örnek olarak, numpy, pyplot vb gibi.
- **Spyder:** SPYDER adı "Scientific Python Development EnviRoment (SPYDER) kelimesinden türetilmiştir. Gelişmiş düzenlemeler, etkilişelimli testler, hata ayıklama özellikleri ile Python dili için entegre bir geliştirme ortamıdır (IDE: Integrated Development Environment).
- **Jupyter Notebook:** Birçok programlama dilinde etkileşimli bilgi işlem için açık kaynaklı bir yazılımdır, açık standartlar ve hizmetler geliştirmek için kullanılan alternatif bir araçtır.

Problem Çözme

Bir program yazmanın amacı bir problemi çözmektir.

Bir problemi çözmek için birden fazla faaliyet oluşur.

1. Problemi anlamak.
2. Çözümü tasarlamak.
3. Alernatifleri düşünün ve çözümleri iyileştirin.
4. Çözümü uygulayın.
5. Çözümü test edin.

Bu faaliyetler tamamen doğrusal değildir- bu faaliyetler birbirleriyle örtüşmekte ve etkileşimlidir.

Algoritmalar

Bir sorunu çözmek için kullanılan adımlar:

Bir algoritmada adımlar tam olarak tanımlanmalı ve adımların sıralanması çok önemlidir.

Bir algoritma aşağıdaki gibi tanımlanır:

- Basit adımların sırası.
- Her adımın yürütüldüğünü belirten kotnrol süreci akışı.
- Ne zaman duracağını belirlemenin yolu.

Algoritmalar dilden bağımsızdır; Bir çözüm herhangi bir programlama dilinde uygulanabilir.

Akışkan şemaları ve sanal kod yöntemler algoritmaları geliştirmek için kullanılan tekniklerdir.

Akışkan Şemalar ve Sanal Kod

Örnek olarak; Kullanıcıya adını soran bir program *'Merhaba [İsim]'* :

- **Sanal kod:** Yukarıdaki gibi kodunuzun makine anlayışından ziyade insan anlayışı için yazılmış sıralı bir versiyonudur.

1. `Input name`
2. `Output 'hello[name]'`

Akışkan Şemalar ve Sanal Kod

Akış Şeması: Bir çözümün diyagramlı gösterimi

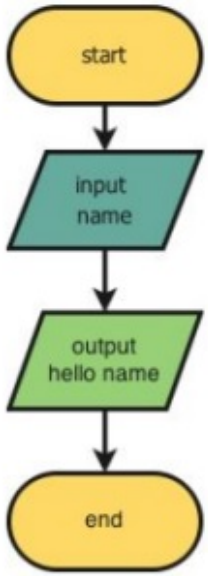

Figure 1: Akış şeması.

Bilgisayar Programları

Bir program geliştirme mekaniği çeşitli aktiveteler içerir.

- Programı belirli bir programlama dilinde (Python gibi) yazmak.
- Programı bilgisayarın yürütebileceği bir forma çevirme
- Oluşabilecek çeşitli hata türlerini araştırmak ve düzeltmek.

Yazılım araçları bu sürecin tüm bölümlerine yardımcı olmak için kullanılabilir.

Spyder/jupyter program geliştirilmesindeki bütün adımlar için araçları sağlar.

Programlama Dilleri

Yüzlerce programlama dili var, bunların içinde en iyi dil ya da en kötü dil yoktur.

Bazı diller farklı uygulamalar için avantaj sağlarken, bazı diller dezavantaj sağlamaktadır.Bu durumda her bir programlama dilinin; ilkel yapıları ('kelimeler'): sayılar, strings ve operatörler(+-/), sözdizimleri (kelimelerin sırasının tanımlanması), anlambilimi (cümlelerin anlam tanımlanması) bakımından farklılık gösterebilir.

Syntax ve Semantics (Sözdizimi ve Anlambilimi)

- Bir dilin **sytax rules**(sözdizimi kuralları), geçerli bir program oluşturmak için sembolleri, ayrılmış kelimeleri ve tanımlayıcıları nasıl bir araya getirebileceğini tanımlar.
- Bir program ifadesininin **sematics**(anlambilimi), bu ifadenin ne anlama geldiğini (programdaki amacı veya rolü) tanımlar.
- Sözdizimsel olarak doğru bir program mutlaka mantıksal (anlambilimi olarak) doğru değildir.

Programlardaki Hatalar

Syntatic Hatalar

- Yaygın hatalar olup, kolayca tespit edilebilir.

Semantic Hatalar

- Öngörülmeyen davranışlara neden olabilir.
- Java gibi bazı diller programı çalıştırmadan önce bunları kontrol eder, Python gibi programlar bunları hata öncesi az kontrol eder.
- Kod, programcının amaçladığından farklı bir anlama sahiptir.
- Program çöküyor, çalışmaya durduruyor.
- Program sonsuza kadar çalışır.
- Program beklenenden farklı bir şekilde cevap verebilir.

Python'a Giriş

Proglamlama dilinin genel amacı:

Hemen hemen her tür programı oluşturmak için etkili bir şekilde kullanılmasıdır.

Avantajları:

- Göreceli olarak basit bir dildir.
- Öğrenmesi kolaydır.
- Çok sayıda ücretsiz kütüphane mevcuttur.
- Tüm işletim sistemleri altında çalışır.

Dejavantajları:

- Minimal statik sematik kontrol nedniyle aşağıdakiler için uygun değildir.
  - Yüksek güvenilirlik kısıtlamaları olan programlar (hava trafiği, kontrol sistemleri, tıbbi cihazşar)
  - Büyük ekip projeleri veya genişletilmiş uzunluk projeleri (çok fazla aşçı olması yemeğin kötü olmasına sebep olur)

Python'a Giriş

Python, derlenmiş bir dile karşı derlenmiş bir dille iç içe geçmiş bir dildir.

Araya giren dillerde, talimatların sırası (kaynak kodu) doğrudan yürütülürken, derlenmiş dillerde kaynak kodu ilk önce bir makine kodu dizine dönüştürülür.

Burada her iki durumunun da avantajları ve dezavantajları vardır.

Python Programları

Bir Program (komut dosyası), tanımların ve komutların bir dizisidir.

- Tanımlar değerlendirilir
- Python Tercüman komutları (ifadeleri) tarafından yürütülen komutlar tercümana bir şeyler yapmasını öğretir.

Komutlar doğrudan bir kabuğa (shell) yazılabilir veya yorumlayıcı tarafından okunan ve değerlendirilen bir dosyada saklanabilir.

Bir programın yürütülmesi başlandığında yeni bir kabuk oluşturulur.

Genellikle bir pencere bu kabuklarla ilişkilidir.

Shell vs Script

`Hello World` iki farklı şekilde kabuk (komut satırı) veya bir komut dosyası (python dosyası) olarak kullanılır.

`Print` () komut ekrana yazdırılır böylece `Print` (′`Hello World`′) konsola verilen metin yazılacaktır.

Built-In Functions

Python yorumlayıcısı, her zaman kullanılabilir olan ve içine yerleştirilmiş bir dizi işleve sahiptir.

Built-in Functions				
abs()	dict()	help()	min()	setattr()
all()	dir()	hex()	next()	slice()
any()	divmod()	id()	object()	sorted()
ascii()	enumerate()	input()	oct()	staticmethod()
bin()	eval()	int()	open()	str()
bool()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	
delattr()	hash()	memoryview()	set()	

Figure 2: Built-in Functions.

Objeler

Programlarda veri nesneleri işlenir.

Nesneler, programların kendilerine yapabileceği şeylerin türlerini tanımlayan bir türe sahiptir.

- Örnek olarak 5 değeri çoğaltılabilir veya eklenebilir.
- `Hello world` arabilen ve dilimlenebilen bir dizidir.
- Örnek olarak Ana bir öğrencidir, böylece derse kaydolabilir, not ortalamasını hesaplayabilir.
- Örnek olarak Milo bir çalışandır, bu yüzden terfi edebilir, işinden istifa edebilir.

Objeler:

- Scalar (alt bölümlere ayrılmaz)
- non-scalar (erişilen iç yapıya sahiptir.)

Objeler

`int` - integers olarak temsil edilir, ex. 5

`float` - gerçek sayı olarak temsil edilir, ex. 3.27

`bool` - boolean değerler Doğru ve Yanlış olarak temsil edilir

`NoneType` - özel ve tek değeri vardır, None

`type` () kullanılan obje türlerini aşağıda inceleyebiliriz :

- `type ("hello world")`

  `str`
- `type(5)`

  `int`
- `type(3.5)`

  `float`
- `type(True)`

  `bool`

Tür Dönüştürme

Bazen verileri bir tüden diğerine dönüştürmemiz gerekir. Örneğin, 3.2 sayısının tam değerini bulmak isteyebiliriz.

Bu türdeki nesneleri diğerine dönüştürmek için yerleşik işlevleri kullanabiliriz.

- `float`(3) - integer 3 den 3.0 olarak dönüştürülür.
- `int`(3.9) - 3.9 ile tam sayısı 3 değerini verir.
- `bool`(5) - sıfır olmayan değer yani booelen Doğruya dönüştürülür.
- `bool`(0) - sıfır olan değer yani booelen Yanlışa dönüştürülür.

Değişkenler

Bi değişkenin üç şeyi vardır:

- label (etiket)
- a type (türü)
- a value (değeri)

Değişkenleri Adlandırma

Python'daki değişken adları aşağıdaki kurallara uygun olmalıdır.

- Değişken adları bir harfle veya alt çizgi ile başlamalıdır.
- Değişken adları yalnızca harf, sayı alt çizgi (character_) karatlerini içebilir.
- Değişken adları boşluk içermez.
- Değişken adları noktalama işareti içeremez.
- Değişken adları tırnak işaretleri veya parantez içine alınmaz.
- Aşağıdaki adlar geçerli değişken adlarıdır:

  `newvariable`, `my2rules`, `SQUARES`

- Geçersiz değişken adları şunlardır:

a `constant`, `3newVariables`

Ayrılmış Kelimeler

Ayrılmış kelimeler, yerleşik anlamlar olan ve değişken adları olarak kullanılamayan anahtar kelimelerdir.

Python'un her sürümü farklı bir anahtar kelime listesine sahip olabilir.

Python'da ayrılmış kelimelerin listesini görmek için komutlar yazılabilir.

```
import keyword
```

```
keyword.kwlist
```

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

Figure 3: Ayrılmış Kelimeler.

Değişkenleri ve Değerleri Bağlama (Atama)

Atama deyimi değişkenin değerini değiştirir.

Atama işleci semboldür

Atama deyiminin sözdizimidir.

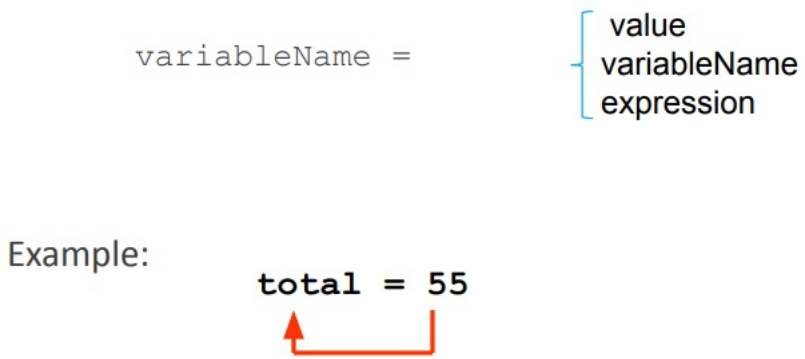

Figure 4: Değişkenleri Atama.

Bağlamları Değiştirme

Python'da yeni atama deyimlerini kullanarak değişken adlarını yeniden bağlayabiliriz.

Önceki değer hala bellekte saklanabilir, ancak değere yapılan başvuru artık mevcut değildir.

Siz bilgisayara hesaplamayı yeniden yapmasını söyleyene kadar alan değeri değişmez.

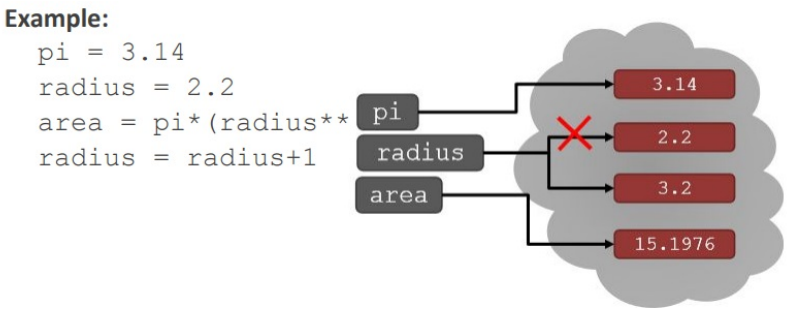

Figure 5: Bağlamları Değiştirme.

Çoklu Proje

Bir projenin sağ tarafındaki ifadeler, herhangi bir bağlama değiştirilmesden önce değerlendirilir.

- Örneğin:

$\mathtt{x, y = 2, 3}$

$\mathtt{x, y = y, x}$

$\mathtt{print('x =', x)}$

$\mathtt{print('y =', y)}$

Sonuç olarak:

$\mathtt{x = 3}$

$\mathtt{y = 2}$

İfadeler

İfadeleri oluşturmak için nesneneleri ve operatörleri birleştirin.

İfadenin bir türü olan bir değeri vardır.

Syntax için basit bir ifade:

$< \texttt{object} > \texttt{operator} >< \texttt{object} >$

INTS ve FLOAT Operatörleri

i+j= toplam

i-j=fark

i*j= çarpım

i/j=float nokta bölmesi

i//j= tamsayı bölme(değer kesri)

i%j= i j'ye bölündüğünde kalan (mod)

i**j= i'nin j kuvveti

Python Aritmetik İşlem Önceliği

Değerlendirme sırası alt ifadeleri gruplandırmak için parantez kullanılarak değiştirilebilir.

Örneğin (x+y)*2, önce x ve y toplanır, sonra sonuç 2 ile çarpılır.

Aşağıdaki şekilde aritmatik operatör önceliği gösterilmektedir.

Operator	Description
**	Exponentiation
+, -	Positive/negative sign
*, /, //, %	Multiplication, division, floor division, modulo/remainder
+, -	Addition/subtraction

Figure 6: Aritmetik İşlemler.

Quick Check

Aşağıdaki İfadelerin Sonuçları nedir?

12/5

12//5

10/4.0

10//4.0

4/10

4//10

12%3

10%3

3%10

Quick Check

12/5=2.4

12//5=2

10/4.0=2.5

10//4.0=2.0

4/10=0.4

4//10=0

12%3=0

10*%3=1

3%10=3

Proje Operatörleri

Operator	Example	Equivatent to
=	x = 5	x = 5
+=	x += 5	x = x + 5
-=	x -= 5	x = x - 5
*=	x *= 5	x = x * 5
/=	x /= 5	x = x / 5
%=	x %= 5	x = x % 5
//=	x //= 5	x = x // 5
**=	x **= 5	x = x ** 5

Figure 7: Proje Operatörleri.

Quick Check

```
a + b + c + d + e
```

```
a + b * c - d / e
```

```
a / (b + c) - d % e
```

```
a / (b * (c + (d - e)))
```

Figure 8: Örnek.

Quick Check

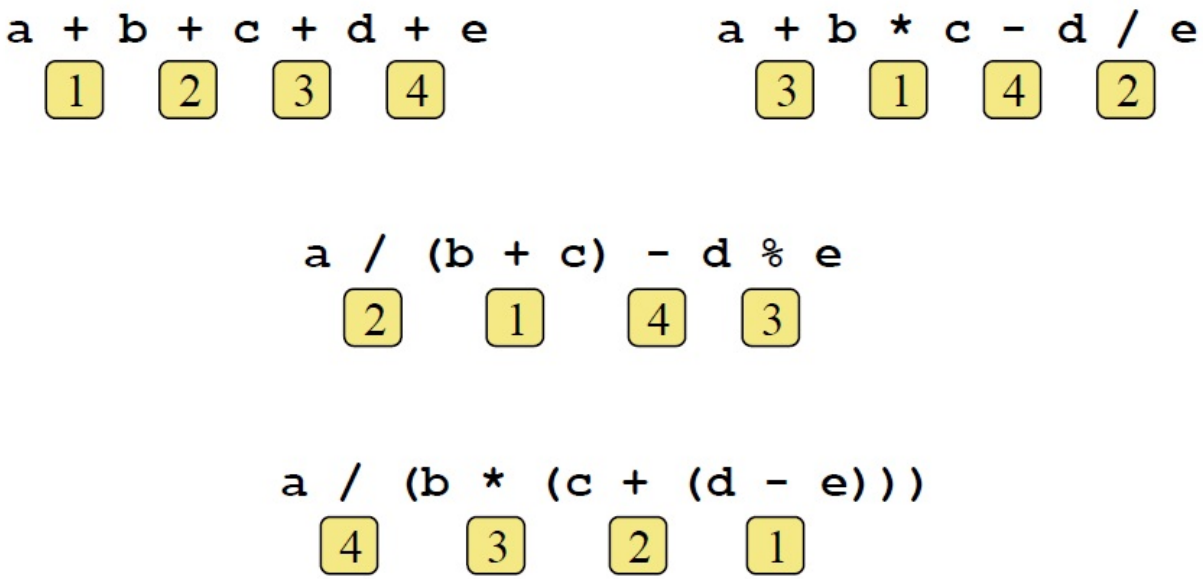

Figure 9: Örnek.

İyi Kod Yazımı

- Kodlama kuralları, kodun okunabilirliğini arttırmak ve Python sürümlerinin geniş yelpazesinde tutarlı hale getirmek için kullanılır.
  - Kod düzeni ve sitili hakkında yönergeler sağlayın
  - Sözleşme örnekleri:
    - Mantıksal bölümleri belirtmek için boş satırları kullanın.
    - Operatörlerden önce ve sonra tek bir alan kullanın (istisnalar mevcuttur.)

Açıklamalar

Açıklamalar amaç ve süreçleri açıklar.

Açıklamalar yorumlanmaz ve programın yürütülmesini etkilemez.

Python açıklamaları iki formda alınabilir:

```
# single line comment multi-line
# """comment,continues to end symbol,across line breaks"""
```

Açıklamalar nasıl dan ziyade neden olarak tanımlanmalıdır.

Çok satırlı yorumlar docstrings olarak da adlandırılır.

Docstrings, daha sonra tartışacağımız operatörler, modüller vesınıflar için belgeler sağlar.

Biçimlendirme Dizeleri- f- dizesi (f-string)

Sayısal veya dizi dize değerleri biçimlendirmek için f dizesi biçimlendirme tekniğini kullanın.

Syntax:

f' dize formatı'

'Biçimlendirilecek dize' kıvrımlı ayıraç $\{\ldots\}$ içine alınmış biçim alanlarını içerir.

Biçim alanları, biçimlendirilmiş değerleri biçim dizesine gömmek için kullanılır ve değerlerin nasıl biçimlendirilmesi gerektiğini (veri türü ve nasıl görüntüleneceği) belirten özel karakter içerir.

Biçimlendirme Dizeleri- f- dizesi (f-string)

salary= 7512.3265

print (**f'** Your salary is {salary}')

Output

Your salary is 7512.32165

vs

salary = 7212.32165

priny(f'Your salary is {salary:.2f}')

Output

Your salary is 7512.32

Format Alanı {Variable_index:format_type}

print(**f'** Your salary is {salary:m.nf}')

m: places reserved for value,including decimal point.

n: number of decimal places

f: format specifier for float values.

print(**f'** Your age is {age:md}')

m: places reserved for value

d: format specifier for int values

print(**f'** Your name is {name:ms}')

m: places reserved for value

s: format specifier for string values

Format Alanı Örneği

salary=7510.2685

age=32

name='Jane'

print(**f'** Name: {name:10s} Age: {age:05d} Salary: {salary:.2f}')

Output

Name: Jane Age:00032 Salary:7510.27

print(**f'** Salary amd age of {name:s} are {salary:12.3f} {age:5d}')

Output:

Salary and age of Jane are 7510.269 32

Alıştırmalar

1. 400 derece Kelvin ısısını Celcius derecesine çevirip alan formatında gösteriniz.
2. Yarıçapı 1.5 cm olan bir dairenin alanını ve çevresini bulan ve sonucu görünteleyen bir program yazınız. `pi` 3.14 alınız.
3. Dört basamaklı bir sayının ilk ve son basamığının toplamını bulan bir program yazınız.
4. Bir araba saatte 110 km hız da 500 km yol kat ettiğine göre, bu yolu kaç saat ve dakikada gideceğini hesaplayan bir program yazınız.