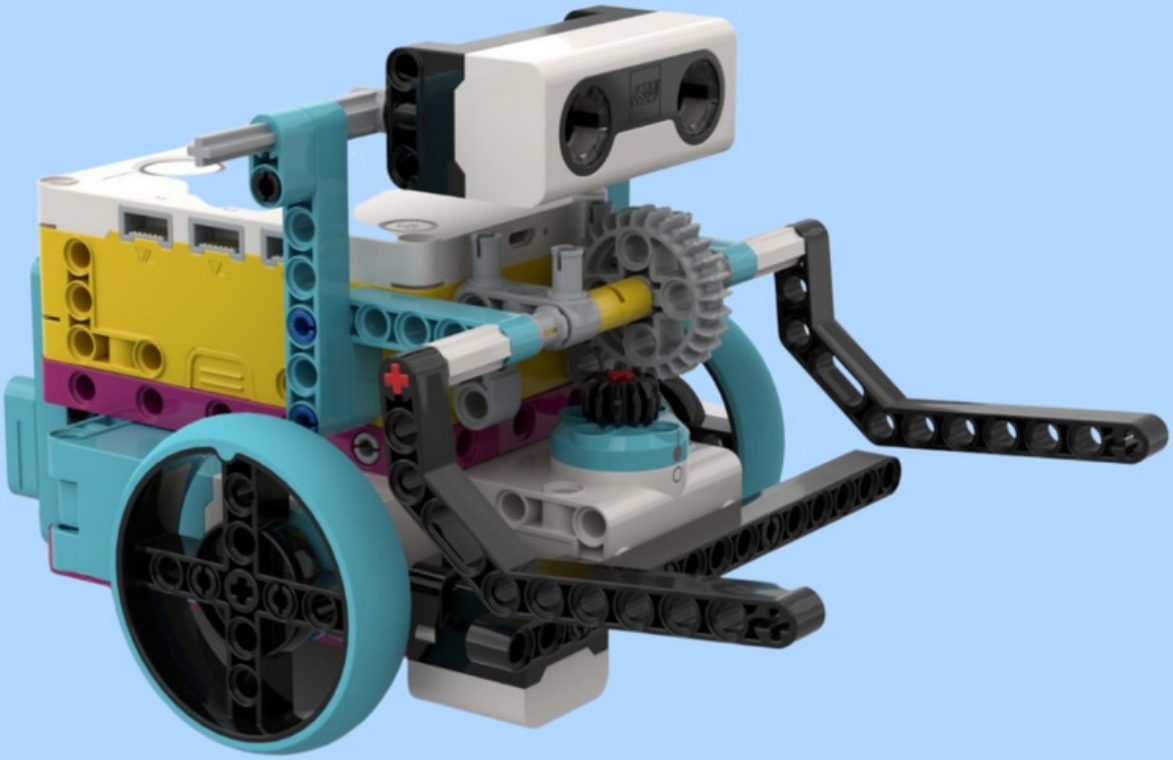
LEGO EDUCATION SPIKE PRIME İLE PYTHON DOKÜMANTASYONU V.1



POWERED BY FRC TEAM 6429 4TH DIMENSION

1. Python’a giriş

Python, yüksek seviyeli bir programlama dilidir ve genellikle okunabilir ve anlaşılabilir sözdizimi ile tanınır. Guido Van Rossum tarafından geliştirilmeye başlanan Python, basit ve etkili bir şekilde kod yazmayı teşvik eder. Geniş bir standart kütüphane içerir ve çeşitli uygulama alanlarında kullanılabilir.

Python'un temel felsefesi "İşleri basit ve açık bir şekilde ifade etmek" olarak özetlenebilir. Bu felsefe, karmaşıklığı azaltmayı ve kodun okunabilirliğini artırmayı amaçlar. Bu nedenle, Python dilinde kod yazmak ve anlamak oldukça kolaydır. Python'u uygulamaya başladığında aynı zamanda 1970'lerde BBC komedi dizisi olan “Monty Python's Flying Circus”un yayınlanmış senaryolarını da okuyan Guido Van Rossum; kısa, benzersiz ve biraz da gizemli bir isme ihtiyacı olduğunu düşündü ve bu dile Python adını vermeye karar verdi.

Python, çeşitli programlama paradigmalarını destekler, bunlar arasında nesne yönelimli, işlevsel ve prosedürel programlama bulunur. Ayrıca, dinamik tipleme özelliği sayesinde değişken türlerini önceden belirtmeniz gerekmez.

Python'un geniş bir topluluğu bulunmaktadır ve bu topluluk sürekli olarak yeni kütüphaneler ve araçlar geliştirerek dilin işlevselliğini genişletmektedir. Bu, Python'u veri analizi, yapay zeka, web geliştirme, bilimsel hesaplama ve daha pek çok alanda popüler bir seçenek haline getirir.

Python aynı zamanda çeşitli platformlarda çalışabilir ve birçok işletim sistemi ile uyumludur. Python betik dosyaları (.py uzantılı) kullanılarak yazılır ve yorumlanır, bu da kodun hızlıca test edilip çalıştırılmasını sağlar.

Sonuç olarak, Python öğrenmesi kolay, geniş kullanım alanlarına sahip ve güçlü bir programlama dilidir.
Hem acemi hem de deneyimli geliştiriciler için ideal bir tercihtir.

2. Python’da sözdizimi

Sözdizimi nedir?

Sözdizimi, bir programlama dilinin nasıl yazılması gerektiğini belirleyen kurallar bütünüdür. Bir dilin sözdizimi, programcıların kodu yazarken takip etmeleri gereken dilbilgisel yapıları ve düzenleri tanımlar. Temel amacı, insanların programları anlayabilir ve düzenleyebilir bir biçimde yazmalarını sağlamaktır.

Python'un sözdizimi de bu amacı taşır ve aşağıdaki temel hedefleri güder:

- ***Okunabilirlik:*** Python, açık ve anlaşılabilir bir sözdizim ile tasarlanmıştır. Kodu okumak ve anlamak diğer dillere göre daha kolaydır. Bu, kodun sadece yazılıp çalıştırıldığı bir araç değil, aynı zamanda diğer geliştiriciler tarafından da anlaşılabilir bir belge olarak hizmet etmesini sağlar.
- ***Kodun Düzeni:*** Python'ın girinti tabanlı sözdizimi, kodun yapısını ve hiyerarşisini belirlemek için kullanılır. Bu, kodun okunabilirliğini artırır ve mantıksal blokları net bir şekilde ifade etmeyi sağlar.
- ***Hata Azaltma:*** Sözdizimi kuralları, geliştiricilerin hataları daha erken aşamalarda yakalamasını sağlar. Uyumlu bir sözdizimi kullanmak, kodun beklenmeyen davranışlardan veya hatalardan kaçınmasına yardımcı olur.
- ***Kod Konsistansı:*** Python, kodun nasıl yazılması gerektiği konusunda belirgin kurallar sunar. Bu, farklı geliştiricilerin benzer projelerde bile tutarlı ve benzer şekilde kod yazmalarına yardımcı olur.
- ***Dilin Anlaşılabilirliği:*** Python, dilin kendine özgü bir şekilde ifade edilmesine olanak tanır. Bu, geliştiricilerin daha az karmaşıklıkla uğraşmasını ve daha az kod yazarak daha fazla iş yapmasını sağlar.

Sözdizimi, programlama dillerinin temel taşıdır ve kodun anlaşılır, düzenli ve hatasız bir

şekilde

yazılmasını sağlar. Python'un sözdizimi, bu hedeflere ulaşmak için özellikle odaklanmış ve geliştiricilere verimli bir kodlama deneyimi sunmayı amaçlamıştır.

a. Girintiler:

Python'da kod blokları oluşturmak için girinti (indentation) kullanılır. Diğer programlama dilleri genellikle süslü parantezler veya başka sembollerle kod bloklarını belirtirken, Python'da bloklar girinti ile belirlenir. Örneğin, bir "if" koşulu altında çalışacak kodu belirtirken, kodu içe gömerek yani girinti yaparak belirtiriz.

Örnek:

```
1. İf x > 5:
2. print(“x 5’ten büyük”)
```

Burada print fonksiyonuna dikkat edin gördüğünüz üzere aynı sütun da değiller. Bunun sebebi print fonksiyonun if koşulunun bir parçası olduğunu göstermek içindir.

b. Değişkenler ve Atama:

Python'da değişkenler, değerleri saklamak için kullanılan isimlendirilmiş bellek bölgeleridir. Değişkenlere değer ataması yaparken = işaretini kullanırız.

Örnek:

```
1. x = 10
2. y = "Merhaba, Dünya!"
3. #Burada x ve y değişkenlerine sırasıyla bir sayı ve bir metin atandı.
```

c. Yorumlar:

Yorumlar, kodun anlaşılmasını ve açıklamasını kolaylaştırmak için kullanılır. Yorum satırları, Python tarafından kodun çalıştırılmasında dikkate alınmaz.

Örnek:

```
1. # Bu bir yorum satırıdır
```


d. Fonksiyonlar ve Metodlar:

Fonksiyonlar, belirli bir işlevi yerine getiren kod bloklarıdır. Python'da fonksiyonlar def anahtar kelimesi ile tanımlanır.

```
1. def toplama(a, b):
2. return a + b
```

Burada 'toplama' adında bir fonksiyon tanımlandı. Bu fonksiyon, iki parametre alır ve bu parametrelerin toplamını döndürür.

e. Koşullu Ifadeler:

Python'da koşullu ifadeler, belirli şartlara bağlı olarak farklı kod bloklarının çalıştırılmasını sağlar. if, elif (opsiyonel) ve else ifadeleri kullanılır.
Örnek:

```
1. if x > 0:
2.  print("x  pozitif")  3.
elif x == 0:
4. print("x sıfır")
5. else:
6. print("x negatif")
```

Bu örnekte, "if" koşulu, x değerinin pozitif olup olmadığını kontrol eder. Eğer pozitifse, ilgili mesaj yazdırılır. Eğer değilse, başka bir koşul kontrol edilir.

.

Python'un sözdizimi, kodun okunabilir ve anlaşılabilir olmasını sağlamak amacıyla tasarlanmıştır. Bu da geliştiricilere daha kolay ve verimli bir şekilde kod yazma imkanı sunar.

3. MicroPython

SPIKE Prime Hub, nispeten sınırlı belleğe ve işlem gücüne sahip, mikrodenetleyici adı verilen küçük bir bilgisayardır. Python programlama dili çok fazla bellek kullanacağından Hub, Python dilinin mikrodenetleyicilerde çalışabilen yüksek oranda optimize edilmiş bir sürümü olan MicroPython'ı kullanmaktadır. SPIKE Prime Hub'ı, sensörleri ve motorları kontrol eden modüller de optimize edilmiş veri tipleri kullanılarak yüksek oranda optimize edilmiştir.

Kod Düzenleyici'nin metin ve sayıları farklı renklerde gösterdiğini gördünüz çünkü bunlar farklı veri tipleridir. Python ayrıca tam sayılar ve ondalık sayılar arasında ayrım yapar. Tam sayılar,
MicroPython da
optimize edilmiş tam sayılar (Integer) veri tipi olarak da bilinir. Ondalık sayılar, optimize edilmemiş

float
veri tipini kullanır, bu nedenle SPIKE Prime modülleri bu veri tipinden kaçınır. Bu nedenle, tam sayılar kullanmalı veya ondalık sayıları tanımlamak için farklı birimler kullanmalısınız. Yarım saniye için 0,5 saniye yerine 500 milisaniye kullanabilirsiniz.

Veri tipleri

Az önce de bahsettiğimiz gibi tam sayılar MicroPython'da Integer veri tipiyle belirtilir.
Bunun yanında diziler (string) ve metinler şeklinde tanımlanır. Stringler 2 tırnak (") arasında veya iki kesme işareti (') arasında olması ile anlaşılabilir.
Ondalıklı veri tipleri (floatlar) bir sayının ondalıklı haliyle gösterimidir.


```
1. "Hello World" #Bu bir String veri tipli metindir.
2. 0.895 #Bu bir float tipi veridir.
3. 234 #Bu bir Integer tipi veridir.
```

Bu veri tiplerini kullanabileceğiniz bir örneği ise şöyle verebiliriz:

```
1. import time #python'nun zaman kütüphanesi
2. time.sleep(2) #print fonksiyonumuzun çalışmadan önce 2 saniye beklemesini sağlıyoruz bunun yerine dilediğiniz her hangi bir float veya integer'ı kullanabilirsiniz ama bizim örneğimiz hatrına biz "2" integer'ını kullandık
3. print("2 saniye duruldu")
```

4. SPIKE Prime Modülleri

SPIKE Prime Hub'ı, sensörleri ve motorları kontrol etmek için SPIKE Prime modüllerine ihtiyaç duyarsınız. Modüller, ilgili kodu düzenlemek için kullanılır. Her SPIKE Prime bileşeni için bir modül; örneğin motor modülü, motorları kontrol etmek için gereken kodu içerir. Bir modülün işlevini kullanmak için öncelikle modülü şu import deyimle içe aktarın:

```
1. import motor
```

.

İhtiyacınız olan modülleri, Python programınızın başında bir kez içe aktarın. Modüller ve işlevleri hakkında daha fazla bilgi için bu Kullanım Kılavuzu'nun SPIKE Prime Modülleri bölümüne bakınız. Ayrıca bu modüller dokümanımızın devam eden kısımlarında örneklerle anlatılacaktır.

5. Parametre Ekleme

Aşağıdaki örnek merhaba() fonksiyonumuzun parametresi bulunmamaktadır; bu nedenle her çağırdığınızda konsolda "Merhaba!" yazar. Daha dinamik hale getirmek için fonksiyon tanımının parantezlerinin içine bir parametre adı ekleyin. Fonksiyon gövdesindeki kod bloğu, parametrenin değerine göre farklı bir şey yapmak için bu parametreyi kullanabilir. Örnek 2'ye bakarsanız **isim** parametresini print'in içine bunu eklememizi sağlayan **+** operatörüne dikkat edin. Bu **+** operatörü stringleri uçlarından birbirine bağlamaya yarar.

Örnek 1:

```
1. def merhaba(): #dinamik olmayan
2. print("Merhaba!")
```

Örnek 2:


```
1. def merhaba(isim): #dinamik olan
2.  print("Merhaba" + isim + " !")#gördüğünüz üzere bu print fonksiyonunda kişiselleştirilmiş bir merhaba mesajı vardır. Parametrenin değişkenliği yüzünden bu fonksiyon örnek 1'deki fonksiyona göre daha dinamiktir.
```

6. Pseudo Code

Pseudo code, bir programın mantığını ve adımlarını anlamak ve ifade etmek için kullanılan, gerçek programlama diline ait olmayan bir yazım şeklidir. Pseudo code, genellikle insana ve diğer programcılara programın genel yapısını ve işleyişini anlatmak için kullanılır. Gerçek bir programlama diliyle yazılmış kod gibi çalışmayacaktır, ancak temel işlem adımlarını ve mantığı anlamak için kullanışlıdır.

Pseudo code yazmak, programın tasarım aşamasında kullanışlıdır. Karmaşık problemleri daha basit adımlara bölmek, mantık hatalarını erkenden yakalamak ve programın genel akışını görselleştirmek için kullanılır. Ayrıca, birden fazla kişiyle çalışırken fikirleri ve planları paylaşmak için de kullanılır.

Gerçek Python kodu:

```
1. sayi1 = 5
2. sayi2 = 10
3. toplam = sayi1 + sayi2
4. print("Toplam:", toplam)
```

Pseudo Code:

.

```
- sayi1 değişkenini 5 olarak tanımla
- sayi2 değişkenini 10 olarak tanımla
- toplam değişkenini sayi1 + sayi2 olarak hesapla
- "Toplam:" yazısını ve toplam değişkenini ekrana yazdır
```

7. Asenkron Programlama nedir?

Asenkron programlama, bir programın işlemlerini sırayla ve beklemeksizin gerçekleştirmek amacıyla kullanılan bir programlama yaklaşımıdır. Bu yaklaşım, özellikle uzun süren işlemlerin (örneğin ağ istekleri, dosya okuma/yazma gibi) paralel olarak çalışmasını sağlamak için kullanılır. Böylece bir işlem tamamlanırken diğer işlemler engellenmez ve programın daha hızlı ve verimli çalışmasına olanak tanır.

SPIKE Prime, MicroPython aracılığıyla asenkron programlama özelliklerini destekler. Bu, sensör verilerini izlerken aynı anda motorları kontrol etmek gibi durumlarda oldukça yararlı olabilir.

Asenkron Programlamanın Temel Unsurları

Koşullu Bekleme: SPIKE Prime'da asenkron programlama, genellikle "await" anahtar kelimesiyle gerçekleştirilir. Bu anahtar kelime, bir işlemin tamamlanmasını beklemek için kullanılırken, programın diğer işlemleri engellemeden çalışmasını sağlar.

Etkin Kullanım: Asenkron programlama, uzun süreli işlemleri beklemeksizin diğer işlemleri gerçekleştirmenize olanak tanır. Örneğin, bir sensör verisi gelene kadar beklemek yerine, bu süre içinde başka işlemleri de yürütebilirsiniz.


```
1. import runloop
2. import hub
3. async def main(): #asenkron bir fonksiyon oluşturur
4.   await  hub.light_matrix.write("birinci  print  fonksiyonu")  #hub'ın  ışık
matrisinde "birinci print fonksiyonu" yazdırıcaktır
5. await hub.light_matrix.write("ikinci print fonksiyonu") #bu 2. satır üstteki
satırdaki kod çalışıp bitmeden önce çalışmayacaktır
6.
7. runloop.run(main()) #fonksiyonu asenkron şekilde çalıştırır
8.
```

8. HUB MATRIS KULLANIMI

SPIKE Prime Hub ekranı 5 x 5 matris ekran şeklinde tasarlanmıştır. Ekranda 25 tane ışık verebilen ve kontrol edilebilen küp şeklinde LED'miz vardır. Bu bölümde de Hub'ın matris ekranında istediğimiz yazıyı nasıl yazdıracağımızı ve bazı şekilleri nasıl göstereceğimizi öğreneceğiz.

a) Matris Ekranda Bir Yazıyı Göstermek

```
1. from hub import light_matrix
```

.

```
2. import runloop
3. async def main(): #fonksiyonu oluşturuyoruz
4.    light_matrix.write("HELLO    WORLD!")    #light_matrix.write    fonksiyonunu
çağırıyoruz ve ona HELLO WORLD! stringini veriyoruz
6. runloop.run(main()) #fonksiyonumuzu çağırıyoruz
```

Siz deneyin:

Pekâlâ şimdi dünyaya merhaba dediğimize göre ekranda kendi isminizi veya bir arkadaşınızın ismini yazacak şekilde değiştirin.

b) Matris Ekranda Bir Şekil Göstermek

Kodumuzu kişiselleştirdiğimize göre şimdi de matris ekranda bir inek resminin görünmesini sağlayalım.

```
1. from hub import light_matrix
2. import runloop
3. async def main():
4.    light_matrix.show_image(light_matrix.IMAGE_COW)    #light_matrix.show_image
fonksiyonunu  çağırıp  hazır  var  olan  light_matrix  kütüphanesinden  IMAGE_COW
resmini alıyoruz.
5. runloop.run(main())
```


9. Tekli Motor Kontrolü:

Bölüm |

Bir motoru beş farklı yöntemle çalıştırabilirsiniz. Şimdi en kolay yöntemle başlayacağız. Bu yöntemde motora sadece bir motorun portunu ve motora hız vermemiz yeterlidir. Ama bununla beraber gelen kötü yan etki ise motor durdurulmazsa sonsuza kadar ilerler. Hadi hemen aşağıdaki kodu sizde deneyin.

```
1. import motor
2. from hub import port
3.
4.  motor.run(port.A,  5000)  #A  portundaki  motoru  5000  derece/saniye  hızda
sonsuza kadar ilerler
5.
```

Yukarıdaki kodu çalıştırdığınıza göre şimdi biraz daha işleri biraz daha zorlaştıralım. Az önce yazdığımız koda ivme ekleyeceğiz. (ivme = derece/sn^2) bunun için az önce çağırdığımız motor.run fonksiyonun içine şunu ekleyeceğiz: acceleration=10 --accelleration bir opsiyonel parametredir-- öncelikle aşağıdaki kodu çalıştırın ve bu eklediğimiz yeni parametrenin ne işe yaradığını tahmin edin.

```
1. import motor
2. from hub import port
```

.

```
3. motor.run(port.A, 1110, acceleration=10)
4. motor.STOP(port.A) #A motorunu durdurur
5.
```

Tahmin ettiğinizi varsayarak cevabı veriyorum. Cevap 1110 derece/sn hıza ulaşana kadar 10 derece/sn² ivmeyle hızlanacak. Doğru tahmin edebildiniz mi?

Bölüm ||

Beş yöntemden ikincisine gelmiş bulunmaktayız. Dereceyle motorları kontrol etmek... Motorunuzun A portuna bağlı olduğunu kontrol edin ve aşağıdaki programı deneyin.

```
1. import motor
2. from hub import port
3.
4. #A portundaki motoru çalıştırır.
5.  motor.run_for_degrees(port.A, 360, 1000)
6.
```

Bu programda A portuna takılı olan motorumuzu saniyede 1000 derece hızla 360 derece çalıştırır.

“motor.run_for_degrees” Parametrelere ayrıştırılmış hali:

1. Parametre:
o Bu parametrede çalışan motorumuzun hangi porta takılı olduğunu belirtiyoruz.
2. Parametre:
o Bu parametrede ise motorumuzun kaç derece gideceğini belirtiyoruz.
3. Parametre:
o Bu parametrede ise motorumuzun ne kadar hızlı gideceğini derece/saniye türünden belirtiyoruz.

Pekâlâ şimdi de opsiyonel olan parametrelerimize bakalım:

1. Acceleration (ivme):
o Y hıza ulaşana kadar x hızda hızlanacak (ivme = derece/sn²)
2. Deceleration (ivmeli yavaşlama):
o İvme de olduğu gibi bu parametremizin de ölçüm birimi derece/sn²
Örnek olarak eğer bir motor 500 derece/sn hızda dönüyor ise deceleration=10 dersek yavaş yavaş 500 derc/sn’den duruşa geçecektir.
3. Stop (birçok seçenek içerir):
a. motor.COAST:

Motor boşa alınır motora elektrik gitmez.
b. motor.BREAK:

Motor fren yapar durduktan sonra da fren yapmaya devam eder.
c. motor.HOLD:

Motor pozisyonunu korur.
d. motor.CONTINUE:

.

Başka bir komut gelene kadar aynı hızda devam eder.

e. motor.SMART_COAST

Durana kadar fren yapar durduktan sonra motoru boşa alır ve sonraki komutta yapılan hataları düzeltir.

f. motor.SMART_BRAKE:

Durana kadar fren yapar durduktan sonra fren yapmaya devam eder sonraki komutta yapılan hataları düzeltir.

Örnek Deceleration parametresi kullanımı:

```
1. import motor
2. from hub import port
3. motor.run_for_degrees(port.A, 3600, 500, deceleration=10)
4.
```

Bölüm |||

Beş yöntemden üçüncüsüne gelmiş bulunmaktayız. Bu bölümde motorumuzu zamana göre çalıştırmayı öğreneceğiz. Bu dokümantasyonun başında konuştuğumuz gibi MicroPython Float veri tipli optimize edilmemiş verilerden kaçınır bu sebeple bu motor kontrolü fonksiyonu için milisaniye kullanacağız (Yani 5 saniye = 500 milisaniye)

Basit bir örnek kod yazalım:


```
1. from hub import port
2. import motor
3. motor.run_for_time(port.A, 1000, 500) #A portunda 1 saniye boyunca 500
hızda çalışır.
4.
```

“motor.run_for_time” Parametrelere ayrıştırılmış hali:

1. Parametre:

o Bu parametrede çalışan motorumuzun hangi porta takılı olduğunu belirtiyoruz.

2. Parametre:

o Bu parametrede ise motorumuzun kaç milisaniye gideceğini belirtiyoruz.

3. Parametre:

o Bu parametrede ise motorumuzun ne kadar hızlı gideceğini derece/saniye türünden belirtiyoruz.

Opsiyonel olan parametreler:

1. Acceleration (ivme):

o Y hıza ulaşana kadar x hızda hızlancak (ivme = derece/sn²)

2. Deceleration (ivmeli yavaşlama):

o İvme de olduğu gibi bu parametremizin de ölçüm birimi derece/sn² 0 hıza ulaşana kadar -x hızda yavaşlayacak

3. Stop (birçok seçenek içerir):

a. motor.COAST:

Motor boşa alınır motora elektrik gitmez.

b. motor.BREAK:

Motor fren yapar durduktan sonra da fren yapmaya devam eder.

c. **motor.HOLD:**
Motor pozisyonunu korur.

d **motor.CONTINUE:**
Başka bir komut gelene kadar aynı hızda devam eder.

. **motor.SMART_COAST**
Durana kadar fren yapar durduktan sonra motoru boşa alır ve sonraki komutta

e. yapılan hataları düzeltir.

f. **motor.SMART_BRAKE:**

Durana kadar fren yapar durduktan sonra fren yapmaya devam eder sonraki komutta yapılan hataları düzeltir.

Bölüm |V

Beş yöntemden dördüncüsüne gelmiş bulunmaktayız. Bu bölümde motorumuzu mutlak konumuna döndürmeyi öğreneceğiz. Mutlak konum motorumuzun 360 derece içerisinde bulunduğu spesifik bir açıyı ifade eder. Örneğin eğer bu fonksiyonu çağırırsak ve parametreye 90 verirsek o zaman motorumuz 360 derece içindeki **doksanıncı derece konumuna** (ne kadar dönmesi gerekirse gereksin) gidecektir.

Örnek kod:

```
1. from hub import port
2. import motor
3. motor.run_to_absolute_position(port.A, 90, 500) #motoru 500 derece/sn hızında 90’nıncı dereceye götürür.
```


“motor.run_to_absolute_position” Parametrelere ayrıştırılmış hali:

1. Parametre:
o Bu parametrede çalışan motorumuzun hangi porta takılı olduğunu belirtiyoruz.
2. Parametre:
o Bu parametrede motorumuzun gideceği mutlak konumu belirtiyoruz.
3. Parametre:
o Bu parametrede ise motorumuzun ne kadar hızlı gideceğini derece/saniye türünden belirtiyoruz.

Opsiyonel olan parametreler:

1. Acceleration (ivme):

o Y hıza ulaşana kadar x hızda hızlanacak (ivme = derece/sn²)

2. Deceleration (ivmeli yavaşlama):

o İvme de olduğu gibi bu parametremizin de ölçüm birimi derece/sn² 0 hıza ulaşana kadar -x hızda yavaşlayacak

3. Direction:

a. Motorun hangi yönde döneceğini belirmenize yardımcı olur.
b. motor.CLOCKWISE #motoru saat yönünde döndürür
c. motor.COUNTERCLOCKWISE #motoru saat yönünün tersine döndürür
d. motor.SHORTEST_PATH #motoru istenen mutlak konumuna en kısa yoldan götürür
e. motor.LONGEST_PATH #motoru istenen mutlak konumuna en uzun yoldan götürür

4. Stop (birçok seçenek içerir):

.

a. motor.COAST:

Motor boşa alınır motora elektrik gitmez.

b. motor.BREAK:

Motor fren yapar durduktan sonra da fren yapmaya devam eder.

c. motor.HOLD:

Motor pozisyonunu korur.

d. motor.CONTINUE:

Başka bir komut gelene kadar aynı hızda devam eder.

e. motor.SMART_COAST

Durana kadar fren yapar durduktan sonra motoru boşa alır ve sonraki komutta yapılan hataları düzeltir.

f. motor.SMART_BRAKE:

Durana kadar fren yapar durduktan sonra fren yapmaya devam eder sonraki komutta yapılan hataları düzeltir.

Bölüm V

Beş yöntemden sonuncusuna gelmiş bulunmaktayız. Bu bölümde "motor.run_to_relative_position" fonksiyonunu işleyeceğiz. Bu fonksiyonu dolayı uygulamalı test etmenizi tavsiye ederiz.

Her şeyden önce motorunuzun kaçıncı derecede olduğuna bakın bunu görmek için kodlayıcımızın sol üst köşesinde bulunan motor logolu verilerden yararlanabiliriz. Motorun derecesini elinizle 90° pozisyonunda olacak şekilde ayarlayın. Aşağıdaki kodu çalıştırın.


```
1. from hub import port
2. import motor
3. def main():
4.  motor.run_to_relative_position(port.A,  720,500)  5.
main()
```

Bildiğiniz üzere motorun 1 rotasyonu 360°'dir. Programı çalıştırdığınız zaman 2 rotasyon dönüp başladığınız yere yani 90° pozisyonuna gelmesi gerekir ama gözlemleyeceğiniz üzere farklı bir sonuç aldınız. Bunun sebebi bu fonksiyon çalıştığı zaman başlangıç yerini daima 0° olarak alır yani 0'dan başlayarak 2 rotasyon dönerseniz yine 0°'ye geri gitmiş olursunuz. Eğer 720 yerine 740 demiş olsaydınız 0° yerine 20°'ye gitmiş olacaktır.

"motor.run_to_relative_position" Parametrelere ayrıştırılmış hali:

1. Parametre:
o Bu parametrede çalışan motorumuzun hangi porta takılı olduğunu belirtiyoruz.
2. Parametre:
o Bu parametrede ise 0'dan başlayarak döneceği derece sayısı.
3. Parametre:
o Bu parametrede ise motorumuzun ne kadar hızlı gideceğini derece/saniye türünden belirtiyoruz.

Opsiyonel olan parametreler:

1. Acceleration (ivme):
o Y hıza ulaşana kadar x hızda hızlanacak (ivme = derece/sn²)
2. Deceleration (ivmeli yavaşlama):
o İvme de olduğu gibi bu parametremizin de ölçüm birimi derece/sn² 0 hıza ulaşana kadar -x hızda yavaşlayacak

.

3. Stop (birçok seçenek içerir):

a. motor.COAST:

- Motor boşa alınır motora elektrik gitmez.

b. motor.BREAK:

- Motor fren yapar durduktan sonra da fren yapmaya devam eder.

c. motor.HOLD:

- Motor pozisyonunu korur.

d. motor.CONTINUE:

- Başka bir komut gelene kadar aynı hızda devam eder.

e. **motor.SMART_COAST**

- Durana kadar fren yapar durduktan sonra motoru boşa alır ve sonraki komutta yapılan hataları düzeltir.

f. motor.SMART_BRAKE:

- Durana kadar fren yapar durduktan sonra fren yapmaya devam eder sonraki komutta yapılan hataları düzeltir.

10. Döngüler

Döngülerin syntax'ı da bir if koşulu gibidir. Döngü fonksiyonunun çağırımı ardından bir sütun boşlukla yazılır.

Bölüm | While döngüleri:

While döngüleri verilen veri doğru olduğu sürece döngüyü devam ettirir.

Örnek olarak while True sonsuza kadar gider. Ama eğer şöyle bir şey yaparsak:

```
1. a = 0
2. while a < 5:
3. a= a+1
4.
```

Bu kod a'ya beşten yüksek olana dek 1 ekler ve 5ten fazla olduğu zaman da döngü biter.

Bölüm || For döngüsü:

For döngüleri yineleme yapmak için kullanılır. Örnek olarak bir listeniz var ve bu listenin içindeki her şeyi ekrana çıkartmak istiyorsunuz bunu aşağıdaki kod ile yapabilirsiniz:

```
1. liste = [1, 2, 3, 4, "abc"]
2. for i in liste:
3. print(i)
4.
```

Peki bu "i" nedir? Buradaki "i" bir değişkendir ve bu değişken için her hangi bir şey yazabilirsiniz. Bu değişken liste adlı listenin içeriklerinin verilerini alır her dönüşte ya da yinelemede.

Döngü Ekleri:

"Break" ile başlarsak ilk olarak bu fonksiyonu bir döngü içerisinde çağırırsak döngü break bu

döngüden
çıkmamızı sağlar. örnek kod:

```
1. for i in “ornek_string”:
2. if i == “k”:
3. Break #k’ye geldiği zaman donguden cikar
4. print(i)
5.
```

“Continue” ile devam edelim, continue fonksiyonumuz çağırdığımızda ise geri kalan kodu atlayarak döngüyü sonraki adımdan devam ettirir ve alttaki kodu çalıştırmaz.

Örnek kullanım:

```
1. for i in “ornek_string”:
2. if i == “k”:
3. continue #k’ye geldiği zaman k yi atlar
4.  print(i) #sonuc olarak butun harfleri ve alt tireyi yazar k harfi haric
5.
```

Son olarak Range’e gelelim. **“Range”**, 0’dan başlayarak x-1 sayısına (x herhangi bir sayısal değer alabilir.) kadar döngüyü döndürür bu olurken de aynı şekilde yine i değişkenine o rotasyondaki sayı atanır mesela 2. rotasyondaysa i değişkeni 1 olur. Başka bir kullanımı ise x,y türünde de yazabiliriz x sayısından başlayarak y-1 sayısına kadar devam eder. Örnek olarak range(6) yazarsak döngü altı kere döner ve I'nin değerleri 0,1,2,3,4,5 olur. Örnek kod:


```
1. for i in range(6):
2. Print(i)#0,1,2,3,4,5 sayilarini yazar. 3.
```

11. Sensörler

Bu bölümde sensörlerden bahsedeceğiz. Sensörler sadece 1 parametre alır ve o da sensörün bulunduğu porttur. Ilk olarak mesafe sensörümüzü inceleyelim. Mesafe sensörü adı üzerinde mesafe ölçer ve mesafeyi milimetre cinsinden geri verir. Bu sensörle 200 cm’e ye kadar ölçüm yapabilirsiniz. Haydi deneyelim.

```
 1. import distance_sensor
2. from hub import port
3. import motor
4. async def main():
5. motor.run(port.A, 500)
6. motor.run(port.B,500)
    7. if Distance_sensor.distance(port.C) <= 1500:
8. motor.stop(port.A)
9. motor.stop(port.B)
10.
```

.

Eger mesafe sensörü 15 CMlik bir veri geri verir ise motorları durdurur ama eğer distance sensör 15cm veya daha az bir yakınlık ölçmezse o zaman sonsuza kadar durmadan gider.

Renk sensörü

Şimdi ise renk sensörüne geçiyoruz renk sensörünün birden fazla işlevi vardır bu işlevler arasında:

1. Işığın yansıma seviyesini ölçebilir.
2. Renkleri tanımlayabilir.

Birinci işlev:

Renk sensörü önüne koyulan her renkli objenin rengini tanımlayamaz. Tanımlaya bildiği renkler arasında: Kırmızı, Yeşil, Mavi, Fuşya, Sarı, Turuncu, Gök mavisi, Siyah ve Beyaz yer alır.

```
1. from hub import port
2. import color_sensor
3. color_sensor.reflection(port.D)#isigin yansima seviyesini soyler
4.
```

#Örnek olarak bunu cizgi takip icin kullanabilirsiniz

İkinci işlev, Renk Tanımlama:


```
1. from hub import port
2. import color_sensor
3. color_sensor.color(port.D)#o anda gördüğü rengi geri verir.
4.
```

Kuvvet Sensörü

Bu sensöür üstüne bir basınç uygulanıyor mu ve uygulanıyorsa da kaç decinewton uygulanıyor onu bildirir. Fonksiyonları:

Force:

Kaç decinewtonlık güç uygulanıyor onu söyler.

Pressed:

Kuvvet sensörüne basılıyor mu onu söyler.

Kullanımları:

1. Force

```
1. from hub import port
2. import force_sensor
3.
4. print(force_sensor.force(port.D)
```

.

```
5.
```

2. Pressed

```
1. from hub import port
2. import force_sensor
3.  print(force_sensor.pressed(port.D))
4.
```

4TH DIMENSION 6429