1) **Hangisi ile fonksiyon oluşturulur.**
   A. __init__ fonksiyon1():
   B. func fonksiyon1():
   C. def fonksiyon1():
   D. const fonksiyon1():
   E. class fonksiyon1():
2) a=.....(input("vize notunuzu girin = "))

   a*5

   a+5

   # tür dönüşümü sonrası a'ya atanan değer sonrasında matematiksel işlemlerde kullanılabilir ve tam sayı değer vermeli

   **Lütfen birini seçin:**

   A. int(.....)
   B. float(.....)
   C. Convert.To.Int32(.....)
   D. int.parse(..)
   E. IntValueof(.....)

3) Aşağıdakilerden hangisi karar yapılarında kullanılan deyimlerden birisi değildir?

   A. while
   B. continue
   C. if... : elif....: else:
   D. break
   E. def

4) Aşağıdaki kavramları birer cümle ile kısaca açıklayınız

a) Encapsulation (kılıflama-sarmalama) nedir?

Bir sınıfın verilerini (özelliklerini) dışarıdan doğrudan erişime kapatıp, kontrollü şekilde erişim sağlanmasına **kapsülleme** denir.

b) Inheritence (kalıtım) nedir?

Bir sınıfın (alt sınıf) başka bir sınıftan (üst sınıf) özellik ve metotlarını miras almasına **kalıtım** denir.

5) Insan sınıf ve öğrenci ve hoca sınıf oluşturunuz. Bu sınıflar arasında inheritence, metotlara aşırı yüklenme, ezme, çok biçimlilik (polymorphism) dahil tüm nesne tabanlı prensipleri gösteriniz

```
class Insan:
    def __init__(self, isim, yas, cinsiyet):
        self.isim = isim
        self.__yas = yas       # encapsulation (private)
        self.cinsiyet = cinsiyet
```

```
    def bilgi_ver(self):
        print(f"İsim: {self.isim}, Yaş: {self.__yas}, Cinsiyet: {self.cinsiyet}")
    # Getter-Setter
    def get_yas(self):
        return self.__yas
    def set_yas(self, yeni_yas):
        if yeni_yas > 0:
            self.__yas = yeni_yas
        else:
            print("Yaş negatif olamaz!")
    def konus(self):
        return f"{self.isim} konuşuyor."
class Hoca(Insan):
    def __init__(self, isim, yas, cinsiyet, brans):
        super().__init__(isim, yas, cinsiyet)
        self.brans = brans
    # Override
    def konus(self):
        return f"{self.isim} adlı hoca {self.brans} dersini anlatıyor."
class Ogrenci(Insan):
    def __init__(self, isim, yas, cinsiyet, sinif, okul_no):
        super().__init__(isim, yas, cinsiyet)
        self.sinif = sinif
        self.__okul_no = okul_no
    # Getter-Setter
    def get_okul_no(self):
        return self.__okul_no
    def set_okul_no(self, yeni_no):
        if yeni_no > 0:
            self.__okul_no = yeni_no
    # Override
    def konus(self):
        return f"{self.isim} adlı öğrenci derste soru soruyor."
    def katil(self):
        return f"{self.isim} adlı öğrenci {self.sinif} sınıfında derse katılıyor."
# Ana program
hoca1 = Hoca("Ayşe", 40, "Kadın", "Fizik")
ogr1 = Ogrenci("Mehmet", 18, "Erkek", "11A", 1234)
print(hoca1.konus())
```

```
print(ogr1.katil())
print(ogr1.konus())
```

6) Çalışan ata sınıfı, mavi ve beyazyaka çalışan alt sınıf yapıcı metot tanımlayalım. mavi yaka 3 vardiya beyaz yaka 2 vardıya .çalışsın. Çalış metotdu/fonlsiyonunda tezgahta/fabrika içinde çalış ve masabaşı çlaış gibi iki farklı çıktı veren fonksiyon tanımlayalım (metot overriding) nesne türetelim (construction ile) abstract özgeçmiş sınıfı olsun ve iki sınıfta bu sınıftaki boş (abstract metotları doldursun)

```
from abc import ABC, abstractmethod

# Soyut (abstract) sınıf
class Ozgecmis(ABC):
    @abstractmethod
    def calis(self):
        pass
    @abstractmethod
    def bilgileri_goster(self):
        pass
# Üst sınıf: Çalışan
class Calisan(Ozgecmis):
    def __init__(self, isim, yas):
        self.isim = isim
        self.yas = yas

    def bilgileri_goster(self):
        print(f"İsim: {self.isim}, Yaş: {self.yas}")
# Alt sınıf: Mavi Yaka
class MaviYaka(Calisan):
    def __init__(self, isim, yas):
        super().__init__(isim, yas)
        self.vardiya = 3
    def calis(self):
        print(f"{self.isim} fabrikada, tezgahta çalışıyor. (Vardiya: {self.vardiya})")
# Alt sınıf: Beyaz Yaka
class BeyazYaka(Calisan):
    def __init__(self, isim, yas):
        super().__init__(isim, yas)
        self.vardiya = 2
    def calis(self):
        print(f"{self.isim} ofiste, masa başında çalışıyor. (Vardiya: {self.vardiya})")
```

```
# Nesneler
mavi = MaviYaka("Ali", 30)
beyaz = BeyazYaka("Elif", 27)
# Test
mavi.bilgileri_goster()
mavi.calis()
beyaz.bilgileri_goster()
beyaz.calis()
```