**Session 1**

* Print(“Hello world”)
* #Comment için kullanılır

**Session 2**

* String data type’larında bir alt satıra geçecek şekilde yazıyorsak 3 tane tırnak açıp 3 tane kapatmalıyız. “”” bu bir örnektir “””

***Pep8 Kuralları***

* Yazılan kodların arasına boşluk koyulmalıdır ki okunabilirlik artsın. *Örn: 3 + 6 şeklide.*
* Bir satırda 79 karakterden fazla olmamalıdır.
* Gereksiz commentler girmekten kaçınılmalıdır, commentler açıklayıcı ve net olmalıdır.

***Variable***

* string “Hello world”
* integer 25
* float 3.14
* boolean True

Değişken tanımlanırken numeric karakterler başta olamaz.

* Örn: 4\_name = “Mikail”
* name\_4 = “Leyla”

Değerler yan yana atanabilir ve yazdırılabilirler.

* name\_7, name\_8 = "Selin", "Süleyman"
* print(name\_7, name\_8)

*type()* fonksiyonu bir değerin data type’ını gösterir.

* type(name)
* str

*del* bir değişkeni tamamen siler.

* del name\_14

**Session 3**

**Type Conversion**

**Not:** Dataları kalıcı olarak çevirmek istiyorsak atama işlemi yapmalıyız.

* age = str(age)
* **String’e çevirme**
* str(33) 🡺 “33”
* str(False) ”False”
* **Integer’a çevirme**
* int(“33”) *🡺 33*
* *int("33a”) -içerisinde harf bulunursa çevirme yapmaz, Python ile Java farklıdır.*
* float("5.4”) 🡺 5 *(Python Floatı int’e çevirirken noktadan sonrasını siler)*
* int(True) 🡺 1
* **Float’a çevirme**
* float(“3.14”) 🡺 3.14
* float(False) 🡺 0.0

**Arithmetic Operators**

* + **Toplama**: 3 **+** 5 🡺 8
    - **Çıkarma**: 9 **–** 4 🡺 5
      * **Çarpma**: 8 **\*** 5 🡺 40

Kare alma: 5 **\*\*** 2 🡺 25 *(5.5)*

Küp alma: 2 **\*\*** 3 🡺 8 *(2.2.2)*

* + - * + K. **Bölme**: 40 **/** 10 🡺 4.0  
          T. **Bölme**: 40 **//** 10 🡺 4  
          Kalan: 25 **%** 3 🡺 1

**Soru**: Yarıçapı 7 olan bir dairenin alanını bulan Python kodunu yazınız.

pi = 3.14

r = 7

x = pi \* r \*\* 2

print(x)

**Soru**: Bir kenarı 5, diğer kenarı 12 olan dik üçgenin hipotenüsünü bulan Python kodunu yazınız.

a = 5 \*\* 2

b = 12 \*\* 2

x = a + b

h = x \*\* 0.5

print(h)

**Session 4**

**print() function**

* **sep** = “ ” içine girilen karakterle cümleleri birbirinden ayırır

print("Mikail", "Serhan", "Mustafa", sep=", ")

Mikail, Serhan, Mustafa

* **end**= “ “ içine girilen karakteri her satırın sonunda kullanır.

print("a" , end=", ")

print("b", end=", ")

print("c")

a, b, c

**input()**

Verileri kullanıcıdan almak istediğimiz zaman kullanılır ve input’un data tipi her zaman stringtir.

* input(“Lütfen yasinizi giriniz”)
* int ( input(“Lütfen yasinizi giriniz) ) -*input’u integer olarak almak istedik*-

age = int(input("Lütfen yasinizi giriniz"))

yil = 2024

print("dogum tarihiniz" , yil-age)

dogum tarihiniz 1994

**Soru**: Kullanıcıdan kilo ve boy bilgisini alan ve girilen bilgilere göre vücut kitle indeksini(vki) hesaplayan kodu yazınız.

kilo = float(input("Lütfen kilonuzu giriniz"))

boy = float(input("Lütfen boyunuzu giriniz"))

sonuc = kilo / boy \*\* 2

print("vki:", sonuc)

**Escape Sequences** \

* Ters slash ( \ ) işareti ile kullanılır ve bir sonraki karakteri okur ancak işlevini yerine getirmesini engeller.

print('I\'m Python teacher') -Normalde hata vermesi gerekir-

I'm Python teacher

* + **\n** 🡺 Enter görevi görür ve alt satıra iner.
* **\t** 🡺 Tab görevi görür ve uzun bir boşluk bırakır.
* **\b** 🡺 Backspace görevi görür ve bir karakter siler.

**Session 5**

**Comparasion Operators** >, <, >=, <=, ==, !=

* 5 > 4
* 6 >= 6
* 8 == 8
* 8 != 9
* 7.5 > 4
* True > 0.6 *( True == 1 )*
* “Serhan” > “Sylvanas” *( ASCII table ile ilk harflerini karşılaştırır, ilki aynıysa devamına geçer )*
* **ord**( “a” ) *-içerisine string girilince int döndürür.-*
* **chr**( 105 ) *-içerisine int girince ascii değerindeki harfi döndürür.-*

**Boolean**

🡺 0, 0.0, “ ”, [ ], ( ), { }, None, False -> False

* bool (“ “) 🡺 False
* bool(“serhan“) 🡺 True
* bool(-20) 🡺 True

**Logic Expressions**

**İşlem önceliği sırası;**

**not:** Her zaman tersine çevirir.

**and:** Hepsi True ise en sondaki elemanı döndürür. Bir tane bile False varsa, ilk gördüğü False’u döndürür.  
  
5 **and** 8 🡺 result: **8 |** 5 **and** 0 **and** 8 🡺 result: **0**

5 **and** “ “ **and** 0 **and** 8 🡺 result: **“ ” |** 5 **and** 8 **and** True **and** 1: 🡺result: **1**

**or:** Hepsi False ise en sondaki elemanı döndürür. Bir tane bile True varsa ilk gördüğü True’yu döndürür.

0 **or** 0.0 **or** “ “ 🡺 result: **“ “** | 0 **or** 15 **or** 20 **or** False 🡺 result: **15**

Not: İşlem önceliğine dikkat etmek gerekir. Not -> And -> Or diye gruplayarak eleye eleye gider ÖNEMLİ!

F T T F T T T

False **or** 5 and **not** 0 🡺 result: True | False or not 0 and 6 and “kahoot”

*( not 0 ile başladı, elene elene kahoot kaldı, en son False ile kahoot karşılaştırıldı ve or olduğu için ilk gördüğü True’yu verdi. )*

**Örn 1:**

True **or** False **and** not 0 **and** 3 **or** False

*Çözüm: not 0 == T ile başladık. Ardından andlere geçtik ve kapsadığı ‘False and not 0 and 3’ işlemi başladı ve F T T sonucundan and ilk gördüğü False’u kabul etti. Ardından True or False or False kaldı. Böylece hepsi False olmadığı için or ilk gördüğü True’yu verdi.*

**Örn 2:**

False **or** 5 **and** 0 **and** not True **or** True

*Çözüm: not True == F ile başladık. Ardından andlere geçtik ve kapsadığı ‘5 and 0 and not True’ işlemi başladı ve T F F sonucundan and ilk gördüğü False olan 0’ı kabul etti. Ardından False or 0 or True kaldı. Böylece hepsi False olmadığı için or ilk gördüğü True’yu verdi.*

**Session 6**

**Stringlerin en temel 2 özelliği;**

1. **Immutable**: Değiştirilemez demektir. Yeniden atama yapılmadan kalıcı değiştirilememesidir.
2. **Iterable**: Yenilenebilir demektir. Her karaktere ayrı ayrı ulaşılabilinmesine denir. (örn: index)

**Indexing**

* Indexler 0’dan başlar.   
   **0 1 2 3 4 5**  
  “Serhan”
* "Serhan" [ **0** ] 🡺 S
* "Serhan" [ **5** ] 🡺 n
* "Serhan" [ **-1** ] 🡺 n *( -1 demek saymayı tersten başlatmaktır )*
* "Serhan" [ **-6** ] 🡺 S *( -0 olmadığı için negatifler -1’den başlar o yüzden 6 )*
* + ile concatenation yapılır.
* “zu” **+** “ha” **\*** 4 🡺 “zuhahahaha”

**Slicing** [ start : stop : step ]

* Slicingler’de son karakter hariçtir.  
    
   4 5 6 7
* “Techpro Educating” [ 4:7 ] 🡺 pro
* “Techpro Educating” **[ 4 : 100** ] 🡺 pro Educating *(Slicing’de stop noktasının önemi yoktur)*
* “Techpro” [ **7** ] 🡺 ERROR *( Ancak index’te hata alırız. )*
* Start noktasının default’u ilk karakterdir.
* Stop noktasının default’u son karakterdir ve dahildir.
* Step noktasının default’u 1 adımdır.

“Techpro Educating” [ : : ] 🡺 “Techpro Educating”

“Techpro Educating” [ : : -1 ] 🡺 “noitacudE orpceT” *( Tersten de gidebilir )*

“Techpro Educating” [ 5 : 1 : -1 ] 🡺 “rphc”  *( Tersten ise Start ve Stop ters yazılmalı )*

**String Methods**

* + **Upper Case Method**: Yazıları büyütür.

*“techpro".****upper****()* 🡺 TECHPRO

* **Lower Case Method:** Yazıları küçültür.

*“TECHPRO”.****lower****()* 🡺 techpro

* **Capitalize Method:** İlk harfi büyütür.

"techPRO education".**capitalize**() 🡺 Techpro education

* + - * + **Title Method:** Her kelimenin ilk harfini büyütür.

"techPRO educaTİON".**title**() 🡺 Techpro Education

* **Replace** **Method**: Bir karakteri başka bir karakterle değiştirir.

“Tech?pro”.**replace**(“?”,””) 🡺 Techpro

"?serhan?murat?".**replace**("?","\*",2) 🡺 \*serhan\*murat? *( ilk 2 tanesi ile limitledik )*

**Session 7**

**Is Methodları:** True veya False döndürürler. *“öyle mi?”* sorusu sorarlar.

1. **isalnum():** Alfabetik veya numerik karakterler içerip içermediğini sorgular.
2. **isalpha():** Alfabetik karakterler içerip içermediğini sorgular.
3. **isnumeric():** Numerik karakter içerip içermediğini sorgular.

**len:** Kaç karakter olduğunu söyleyen fonksiyondur.

* len(name) 🡺 6

**Count():** İçerisine girilen karakterin kaç tane olduğunu sayar.

* "techpro education".**count**("e") 🡺 2
* "techpro education".**count**("pr") *( illa tek karakter girmek zorunda değiliz. )*

**Split():** Girilen karakterden cümleyi böler, böldüğünü siler. Default: space

* str\_2.**split**() 🡺 ['Techpro', 'is', 'the', 'best', 'course', 'in', 'the', 'world']
* str\_2.**split**("o") 🡺 ['Techpr', ' is the best c', 'urse in the w', 'rld']
* str\_2.**split**("ğ") 🡺 ['Techpro is the best course in the world'] *( içinde olmayan bir karakterden bölmek istersen, string ifadeyi bütünüyle list haline çevirir. )*

**Strip():** Baş ve sondaki karakterleri temizler. Default: space

* " Techpro ".**strip**() 🡺 Techpro
* "??Techpro??".**strip**("?") 🡺 Techpro
* "??Techpro??".**lstrip**("?") 🡺 Techpro**??** *(****Left*** *Strip ile soldakileri sildik.)*
* "??Techpro??".**rstrip**("?") 🡺 **??**Techpro *(****Right*** *Strip ile sağdakileri sildik.)*

Not: Strip içerisine birden fazla karakter girilirse, onun her varyasyonu kontrol ederek siler. Böylece istenilenin dışında silmeler olabilir, dikkat edilmesi gerekir.

* "Techpro is the besssttss".strip("ts") 🡺 Techpro is the be 🡺 **t**,**s**,**ts**,**st** silindi.

**Session 8**

**If:** Eğer anlamına gelir.

* **if** 5 > 2:
* **if** 2 < 7:
* **if** 7 == 7:
* **if** 5 != 3:

**Else**: if koşulunun aksi durumunda gerçekleşecek durumdur.

* **if** 5 > 2:

print(“5,2’den büyüktür”)

**else**:

print(“5,2’den büyük değildir”)

**Elif**: if şartlarının devam

x = 8

* **if** x < 8:

print(“5,2’den büyüktür”)

**elif** x == 8:

print(“elif çalışır”)

**else**:

print(“5,2’den büyük değildir”)

kahoot = int(input("Lütfen bir sayı giriniz"))

if kahoot <0:

    print("Puanınız 0'dan küçük olamaz")

elif kahoot <= 5:

    print("Dahi misin")

elif kahoot <= 10:

    print("Şaka mısın sen?")

elif kahoot <= 15:

    print("Süpersin")

else:

    print("Girdiğiniz değer 15'in üzerinde olamaz")

**Lists [ ]**

Listeler içlerine her türden veriyi alabilirler. Elemanlarını virgüllerle ayırır. Mutable ve iterable’dır.

* + **Mutable**: Bir atama işlemi olmadan da değiştirilebilir demektir. *(Stringin aksine)*
* **Iterable**: Elemanlarına indexle, slicing yaparak ulaşılabilir demektir.
* 2 farklı yöntemle liste oluşturulabilir.

1. **[ ]** : Her bir tırnak içindeki stringi tek bir eleman sayar. *“Techpro”*
2. **list()** : Stringlerin her bir harfini ayrı ayrı liste elemanı olarak görür. *“T,e,c,h,p,r,o”*

* **len()** fonksiyonu iterable olan dataların eleman sayısını verir.
* **len(**liste\_1**)** *(listler iterable)*
* **len(**1453**)**  *(integerlar iterable değil)*
* Listeler nested, yani iç içe olarak da oluşturulabilir.

liste\_4 = [2, 3.5, False, [1,2,3] ] iç içe ikinci bir liste [ 1,2,3,]

* Listeler içerisinde aynı elemanı birden çok yazmaya müsaade ederler.

liste\_8 = ["Serhan", 5, "Serhan", 10, 5 ]

* İndex girerek liste içerisinden eleman seçebiliriz.

ortaya\_karışık[0] #3

ortaya\_karışık[4][2] #iç içe listler için  
ortaya\_karışık[1:4] # 4.4, ‘Orkun’, False -Slicing de yapılabilir.-

* **Listelerdeki elemanları index ile değiştirmek**
* liste\_1 = ['Betül', 3.14, True, 7, [1, 2, 3, 4, 5, 6]]

**liste\_1** = [ 0 ] = “Sirius” 🡺 0. indexteki eleman Sirius olarak değiştirilir.

* liste\_1 = [Sirius, 3.14, True, 7, [1, 2, 3, 4, 5, 6]]

**append()** **:** Listenin sonuna eleman ekler (TEKLİ)

* sayılar\_1.append(6) [1,2,3,4,5] 🡺 [1,2,3,4,5,6]

**extend() :** Listenin sonuna çoklu eleman ekler. (ÇOKLU)

* liste\_2.extend([4,5]) [1,2,3] 🡺 [1,2,3,4,5]

**insert() :**  İstediğin index’e eleman ekler. (TEKLİ)

* sayılar\_3.insert(1, 20) [11,12,13,14] 🡺 [11,20,12,13,14]

**remove()** : Liste **içerisindeki** elemanı siler. (ELEMAN YOKSA HATA VERİR)

* sayılar\_2.remove(20) #Birden çok varsa ilk gördüğü 20’yi siler.

**pop()** : Listenin indexindeki elemanı siler. (BOŞ BIRAKILIRSA SON ELEMAN)

* sayılar\_1.pop(2) ‘Bulut’

**count() :** Girilen elemandan listede kaç tane olduğunu sayar.

* sayılar\_5.count(3) #2 🡺 3’leri sayar.

**clear() :** Listenin içindeki tüm elemanları temizler.

* sayılar\_2.clear() [11, 12, 13, 14, 15] 🡺 []

**sort() :** Listenin elemanlarını sıralar. (Küçükten büyüğe ve ASCII)

* sayılar\_6.sort() [3, 5, 2, 1] 🡺 [1, 2, 3, 5]

**Session 10**

**TUPLE ( )**

Özellikleri tıpkı stringler gibi immutable*(değiştirilemez)* ve iterable*(index,slicing)* olmasıdır.

* ( )
* tuple( )
* Doğrudan virgül ile ayırarak yazarsak da tuple olarak algılanır.

liste\_1 = 3,2,5

* Parantez ile tek değişkenli bir tuple oluşturacaksak sonuna , eklemek gerekir.

("Techpro",)

* Tuple’lar tıpkı stringlerde olduğu gibi **+** ve **\*** işlemlerini gerçekleştirebilir.

[1,2,3] + [4,5] [1, 2, 3, 4, 5] 🡺 Concat etti.

[1,2,3] \* 3 [1, 2, 3, 1, 2, 3, 1, 2, 3] 🡺 Çoğalttı.

**TUPLE’LARDA INDEXING & SLICING**

Tıpkı stringler ve listlerde olduğu gibi ***(iterable)*** index ve slicing işlemleri yapılabilir.

**tuple\_2** = ("Aslan", 4, True, ["Tulay", 5, 6], ("Zeynep", 2, 3.14))

* tuple\_2[3][0] 🡺 ‘ Tulay ’
* tuple\_2[ 1:4 ] 🡺 (4, True, ['Tulay', 5, 6])

**Not**: Slicing’i hangi data tipinde yaparsan, sonuç o data tipinde olur.

tuple\_2[3].append("Hi") Tuple içerisindeki List’i değiştirebiliriz,

# Tuple içindeki listeler iç işlerinde bağımsız dış işlerinde tuple'a bağımlıdır.

('Aslan', 4, True, **['Tulay', 5, 6, 'Hi'],** ('Zeynep', 2, 3.14))

Tuple içerisindeki string methodlarda string fonksiyonları kullanabiliriz. ANCAK kalıcı olarak değişiklikler kaydedilemez. ***(immutable)***

tuple\_2[0].upper() 'ASLAN'

tuple\_2[0] 'Aslan'

Tuple içerisindeki değişkenler değiştirilemediği gibi silinemez de. Komple tuple’ın kendisini silebiliriz yalnızca.

* del tuple\_2[3] #HATA

**TUPLE METHODS**

**count** ve **index** methodlarıdır.

tuple\_3.count(5) # 5'den kaç tane olduğunu söyledi

tuple\_3.index(3) # 3'ün hangi indexte olduğunu söyledi

**Range() Fonksiyonu** *Start : Stop : Step*

Bir sayıdan diğer sayıya kadar olan sayıları üretir. **Ancak** yazdırabilmek için başka bir fonksiyona ihtiyaç duyulur. Start’ın default değeri 0’dır.

list(range(4,10))

tuple(range(4,10))

list(range(-5,10,2)) #2'şer olarak ilerler, step kısmı verdik.

list(range(10,2,-1)) # tersten de yazdırılır.

**Session 11**

**Dictionary { }** *Key – Value*

Key - value değerleriyle birlikte çalışan ikili sistemdir. Özellikleri **mutable** ve **iterable** olmasıdır. **Syntax**: “**key**” : “**value**”, **key2** : **value**

* {}
* dict()

**1 2 3**

dict\_2 = {"name": "Sirhan", "age": 30, "job": "Data Scientist"}

**Key** oluştururken liste hariç her data tipini kullanabiliriz.

dict\_4 = {0: "zero", 1: "one", 2: "two"} # int örneği

dict\_5 = {False: 0, True: 1} # boolean örneği

dict\_6 = {3.14 : "pi\_number"} # float örneği

dict\_8 = {(1,2,3) : "liste"} # Tuple örneği

dict\_7 = {[1,2,3] : "liste"} HATA Listeler keylerde kullanılamaz

**Value** oluştururken her data tipini kullanabiliriz.

1. **len** () ile dictonary’lerin eleman sayılarına bakabiliriz.

len(dict\_1) 2 # len ile bakabiliyorsak iterable demektir.

1. **keys**() ve **values**() ile dictonary’lerin sadece **key** veya **value** değerlerini çağırabiliriz.

dict\_2.keys() 🡺 dict\_keys(['name', 'age', 'job'])

dict\_2.values() 🡺 dict\_values(['Sirhan', 30, 'Data Scientist'])

* Not: Dictionary’ler **indexlenemez!** - **Key adıyla** çağrılabilir.

dict\_2[0] # HATA -  indexlenemez!

dict\_9["öğrenci"] 🡺 [Murti, 30, 'data\_science']

1. **items()** ile dictonaryleri tuple içerisinde eleman olarak alabiliriz. Böylece *index kullanılabilir.*

dict\_1.items() 🡺 dict\_items([('key1', 'value1'), ('key2', 'value2')])

liste\_2 = list(dict\_1.items())

1. **dict()** yöntemini kullanabilmek için assign etme modeli kullanılır. Ancak bu yöntem genelde ikili büyük verileri kolayca dönüştürmede kullanılır. *( ayrıca intleri key olarak kabul etmez )*

dict(one = 1, two = 2, three = 3) #Syntaxi assign şekline döndü.

**List** to **Dictionary**

liste\_1 = [("name", "Serhan"), ("name2","Fulya"), ("name3","Sedat")]

dict(liste\_1) # Bu şekilde büyük listeleri anında dictionary yapabiliriz.

**Dictionary** to **List** *( items() kullanılır )*

liste\_2 = list(dict\_9.items()) #items ile ikili formatına çevirdik ve hata almadık.

* **Hata örneği;**

list(dict\_9) HATA #items() olmadan kullanıldığında listelerde “key – value” olmadığı için hata verir.

**Hatırlatma**: Bir Dictionary’i index formatında kullanmak için de items() kullanıyorduk. Haricinde key isimleriyle çağırılırlardı.

liste\_2[0] 🡺 ('öğrenci', ['Orkun', 32, 'data\_science'])

1. Dictionarylere key girerek yeni bir value ekleyebiliriz. Ancak Keyler **Unique** olmak zorundadırlar. *(Aynı key isminden iki tane olamaz ama value olur)*

dict\_friend["friend4"] = "Badis" 🡺 {'friend1': 'Sylvanas', 'friend2': 'Buddy', 'friend3': 'Günes', 'friend4': 'Badis'}

**SORU**: Tek sayılar ve çift sayılar dictlerine data giriniz.

sayılar = {"tek\_sayılar": [], "çift\_sayılar": []}

1. *Yöntem:* **append**() kullanarak yapılabilir, ancak tek tek eklenir, uzun ve zahmetlidir.

sayılar["tek\_sayılar"].append(3)

1. *Yöntem:* **extend**() ile birden fazla ekleyebiliriz. *Bknz:* List methods.

sayılar["çift\_sayılar"].extend(range(2,11,2))

**Dictionary Methods**

* **update()**: Dictionary’e eleman ekler. *(Sıralama olmadığı için sonu başı yoktur )*

dict\_friend.update({"friend5" : "Robert"})

*Atama yaparak da eklenebilir*

bb = {"friend6" : "Bulut"}

dict\_friend.update(bb)

* **clear**(): Bir Dictionary’nin içindekileri tamamen boşaltır.

dict\_10.clear() 🡺 { }

* **pop**(): İçine girilen key adını siler.

dict\_11.pop(3.14) 🡺 3.14 – pi\_number silindi.

**Session 12**

**Nested Dictionaries**

myfamily = {"child\_1" : {"name" : "Emily", "year": 2004}, #1

            "child\_2" : {"name" : "Tobias", "year": 2007}, #2

            "child\_3" : {"name" : "Linus", "year": 2011}} #3

İç içe Dictionary olabilir. Bu örnekte “**child\_1**” keyinin, “**name**” keyi ve “**Emily**” value’su, “**year**” keyi ve **2004** value’su vardır.

myfamily["child\_1"] 🡺 {'name': 'Emily', 'year': 2004}

Emily’e ulaşmak istersek, “child\_1”, ve “name” keylerini çağırmalıyız.

myfamily["child\_1"]["name"] 🡺 ‘Emily’

Eğer value değerleri de list veya tuple olsalardı, bu defa index kullanarak ulaşırdık.

myfamily2 = {"child\_1" : {"name" : ("Emily","Rose"), "year": 2004},  #ÖRNEK

            "child\_2" : {"name" : "Tobias", "year": 2007},

            "child\_3" : {"name" : "Linus", "year": 2011}}

myfamily2["child\_1"]["name"] 🡺 ('Emily', 'Rose') #Value değeri Tuple

0 1

myfamily2["child\_1"]["name"][1] 🡺 'Rose'

**SETS – KÜMELER**

Kümeler hakkındaki en önemli bilgi, tüm elemanlarının ***unique*** olduğudur. Ayrıca tıpkı *lists* ve *dictionaries* gibi **mutable** ve **iterable** özelliklerine sahiptirler.

* { } *( Sadece key girince set, key – value girince dict olur. )*
* set()

print(set("hello")) 🡺 {'o', 'e', 'l', 'h'} #print ile elemanları random verir.

{"blue","green","yellow","blue","black","green"} 🡺 {'black', 'blue', 'green', 'yellow'}

**o** **o** **o** **x** **o** **x**

1. Kümeler ***List***, ***Dictionary*** *ve* ***iç içe küme*** barındıramazlar!

{"Günes", 5, True, 3.14, [1,2] (4,5,6) } # **HATA** list barındıramaz!

1. Boş küme yalnızca set() ile oluşturulabilir, çünkü { } ile dictionary oluşturulur.

empty\_set = set()

1. set( ) fonksiyonu yalnızca 1 elaman alabilir.

set("sedat","techpro") # **HATA** birden fazla eleman barındıramaz.

1. Kümeler, içerisinde aynı eleman varsa ilk gördüğünü kabul eder.

set\_1 = {True, False, 1, 5, 0.0, 3.5} 🡺 {False, True, 3.5, 5}

1. Kümeler ile looplar kullanılabilir.

for i in set\_2:

    print(i)

0.0

1

3.5

5

**Union – Intersection – Difference**  *( Birleşim – Kesişim – Fark )*

a = set("suzan")

b = set("yavuz")

* **Union** 🡪  **|**Birleşim demektir. Her iki kümedeki elemanların tamamını verir.

a | b 🡺 {'a', 'n', 's', 'u', 'v', 'y', 'z'} *suzan ve yavuz*

* **Intersection** 🡪 **&**   
  Kesişim demektir. Ortak elemanları verir.

a & b 🡺 {'a', 'u', 'z'} *suzan ve yavuz*

* **Difference** 🡪 **-**  
  Fark demektir. Bir kümede olup, diğer kümede olmayanı verir.

a – b 🡺 {'n', 's'} *suzan*

b – a 🡺 {'v', 'y'} yavuz

* + - * + **Simetric** 🡪 **^**

İki farkı aynı anda elde etmek istersek kullanırız.

set\_3 ^ set\_4 🡺 {'d', 'n', 's', 't'}

**Methodlar**

set\_3 = set("fıstık")

set\_4 = set("fındık")

* **clear():** Kümenin içini temizler.

set\_3.clear() 🡺 set()

* **remove():** Kümenin içinden girilen elemanı siler.

set\_3.remove("s") 🡺 {'f', 'k', 't', 'ı'}

* **pop():** Rastgele eleman siler. UZAK DURMAK GEREKİR.

b.pop()

* **add**(): Kümeler içerisine bir eleman ekler.
* set\_3.add("e") 🡺 {'a', 'e'}
* **is subset():** Bir kümenin diğer kümenin alt kümesi olup olmadığını sorgular.  
    
  **Alt küme:** Bir küme, diğer kümedeki elemanların hepsini içeriyorsa alt kümedir.

set\_5 = set("selin")

set\_6 = set("enes")

set\_6.issubset(set\_5) 🡺 True #set\_6, set\_5’in alt kümesidir.

**Session 13**

**LOOPS**

**While**

Genelde koşullarla çalışır, koşul gerçekleştiği sürece döngü de çalışır. Durması için False’a ihtiyacı vardır. Eğer olmazsa sonsuz döngüye girer.

x = 1

while x < 10: 🡺 1 2 3 4 5 6 7 8 9

    print(x)

    x += 1 #bu artırım olmasa veya döngü satırı dışında olsa sonsuza giderdi.

* Döngü dışında kalma örneği

x = 1

while x < 10:

    print(x)

x += 1 #Döngü dışında şu an! İnfinite loop.

while False:  #True olduğu sürece çalışır o yüzden çıktı vermeyecek.

    print("sanki çıktı vermeyecek gibi")

while True:  #Bu kod sonsuza kadar çalışır

    print("sonsuza kadar çalışır")

**Soru:** 1’den başlayarak 5 dahil rakamların karesini yazdıran while döngüsü kodu yazınız.

x = 1

while x <= 5:

    print(f" {x}'in karesi: {x \*\* 2}")

    x += 1

**Soru 2:** Kilomu tahmin et oyunu kodlarını yazdırınız.

kilo = int(input("Kilomu tahmin et. 10'luk değerler kullanabilirsin"))

while kilo < 90:

    print("Yanlış tahmin tekrar deneyin.")

    kilo = int(input("Kilomu tahmin et. 10'luk değerler kullanabilirsin"))

**Soru 3**: Aklımdaki sayıyı bul oyunu kodlarını yazdırınız.

y = True

aklimdaki\_sayi = 44

while y:

    tahmin = int(input(" 1 - 100 arasında bir sayı tahmin ediniz."))

    if aklimdaki\_sayi < tahmin:

        print(f"Girdiğiniz sayı: {tahmin}. Tahminini azalt")

    elif aklimdaki\_sayi > tahmin:

        print(f"Girdiğiniz sayı: {tahmin}. Tahminini artır")

    else:

        print("Hamilton'u buldunuz")

        y = False #else çalışınca y false olsun istedik ve else içine yazdık.

**Soru 4:** Aklımdaki sayıyı bul oyununa 5 deneme hakkı ekleyin.

y = True

aklimdaki\_sayi = 44

count = 0 🡸 Sayaç ekledik ki denemeleri sayabilelim.

while y:

    tahmin = int(input(" 1 - 100 arasında bir sayı tahmin ediniz."))

    count += 1 **🡸 Sayaç tahmin girildiği gibi artar.**

    if aklimdaki\_sayi < tahmin:

        print(f"Girdiğiniz sayı: {tahmin}. Tahminini azalt")

    elif aklimdaki\_sayi > tahmin:

        print(f"Girdiğiniz sayı: {tahmin}. Tahminini artır")

    else:

        print(f"Hamilton'u buldunuz {count} kerede bildiniz")

        y = False #else çalışınca y false olsun istedik ve else içine yazdık.

    if count == 5: 🡸 Sayaç 5 olduğunda deneme hakları bitti.

        y = False

**Soru 5:** Listenin elemanlarının karesini döndüren while döngüsünü yazınız.

x = 0 #ilk index

while x < len(sayilar\_listesi):

    print(sayilar\_listesi[x] \*\* 2) #sayilar listesinin 0.indexindeki elemana ulaş ve karesini yazdır.

    x += 1

**Soru 6:** Listenin elemanlarının karesini bir liste içinde döndüren while kodunu yazınız.

sayilar\_listesi = [1,3,4,6,5,7]

kareler\_listesi = [] 🡸 **YENİ**

x = 0

while x < len(sayilar\_listesi):

    kareler\_listesi.append(sayilar\_listesi[x] \*\* 2) 🡸 **append ile ekledik.**

    x += 1

print(kareler\_listesi)

**Soru 7:** Listenin elemanlarından çift olanların karesini bir liste içinde döndüren while döngüsünü yazınız.

sayilar\_listesi = [1,3,4,6,5,7]

çiftlerin\_kareler\_listesi = []

x = 0

while x < len(sayilar\_listesi):

    if sayilar\_listesi[x] % 2 == 0: 🡸 **Modulüs ile kontrol ediyoruz.**

        çiftlerin\_kareler\_listesi.append(sayilar\_listesi[x] \*\* 2)

    x += 1

print(çiftlerin\_kareler\_listesi)

**Soru 8:** [1,5,6,7,12,14,15,18,19,32,25] listesinin tek elemanlarının karesini, çift elemanlarının 2 katını liste halinde döndüren while döngüsünü yazın.

sayilar = [1,5,6,7,12,14,15,18,19,32,25]

çift\_kareler = []

tek\_kareler = [] 🡸 **Tek kareler de eklendi.**

x = 0

while x < len(sayilar):

    if sayilar[x] % 2 == 0:

        çift\_kareler.append(sayilar[x] \* 2) 🡸 **Çiftlerin iki katı**

    else:

        tek\_kareler.append(sayilar[x] \*\* 2) 🡸 **Teklerin karesi**

    x += 1

print("Çiftler:", çift\_kareler)

print("Tekler:", tek\_kareler)

**Session 14**

**Random Modülü**

Öncelikle random modülünü import random ile import etmemiz gerekmektedir.

* random**.random() 🡺** 0.14026029577793864

0 ile 1 arasında random sayı üretir.

* random.**seed(35) 🡺** 0.5486946056438222

İçerisine girilen belirli bir sabite göre aynı değerleri rassal üretir.

* random.**randint( 1 , 6 ) 🡺** 6

İçine girilen iki sayı arasında end point dahil random sayılar üretir.

* random.**sample(** range( 100 ), 5 **) 🡺** [2, 30, 41, 10, 20]

İçine girilen ilk sayıya kadar, ikinci sayı adedinde random üretim yapar.

Aşağıdaki kod bloğunda, randint kullanılarak random integer elde ederek her seferinde yeni ve rastgele sayıyla çalışması sağlandı.

aklımdaki\_sayı = random.randint(1,100)  #dün yazılan oyunu random hale getirdik.

while True:

    tahmin = int(input("1 ile 100 arasında bir sayı tahmin edin."))

    if aklımdaki\_sayı < tahmin:

        print(f"Girdiğiniz sayı : {tahmin} Tahminini azalt")

    elif aklımdaki\_sayı > tahmin:

        print(f"Girdiğiniz sayı : {tahmin} Tahminini arttır")

    else:

        print(f"Girdiğiniz sayı : {tahmin} Doğru bildiniz. Tebrikler. ")

        break

Girdiğiniz sayı : **14** Doğru bildiniz. Tebrikler.

**Break & Continue**

**break:** Bir kod bloğunun durdurulmasını sağlar, örneğin döngüleri kırabilir.

Sırasıyla int değerleri toplayan, str ifadeye denk gelince duran kod.

x = 0

toplam = 0

while x < len(liste):

    if type (liste[x]) == int:

        toplam += liste[x]

    elif type(liste[x]) == str:

        break

    x += 1

print(toplam)

**continue:** Kendisinden sonraki işlemi görmezden geldirerek çalışmayı devam ettirir.

Sırasıyla int değerleri toplayan, str ifadeye denk gelince atlayıp devam eden kod.

liste = [1,2,3,4,"Techpro",5,6,7]

x = 0

toplam = 0

while x < len(liste):

    if type (liste[x]) == int:

        toplam += liste[x]

    elif type(liste[x]) == str:

        x += 1   # x += 1'i buraya da yazmamızın sebebi continue kendisinden sonrasındakileri çalıştırmaz. Böylece x artmaz sonsuza gider.

        continue

    x += 1

print(toplam)

**Min(), max()**

liste = [2, 5, 1, 3, 9, 4, 6, 8, 15, 48, 256]

* **min( ):** İçerisine girilenlerin en düşük elemanını verir.

min(liste) 🡺 1

* **max( ):** İçerisine girilenlerin en büyük elemanını verir.

max(liste) 🡺 256

* Eğer liste tamamen string ise Ascii değerlerine göre verir.

min(["techpro" , "ayhan", "irem"]) 🡺 'ayhan'

* Eğer liste karışık bir liste ise, HATA verir.

min(["techpro" , "ayhan", "irem", 5 , 6 ]) 🡺 **HATA**