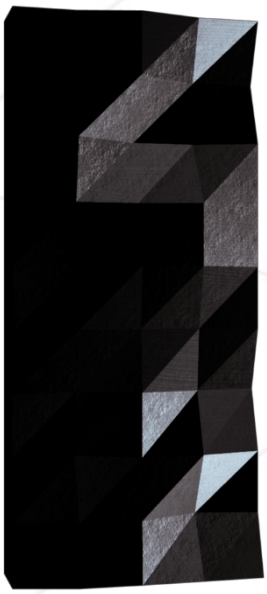

Veri Bilimi için Piscine Python Eğitimi - 0

Başlangıç

Özet: *Bugün, Python Programlama Dilinin temelleri hakkında bilgi edineceksiniz.*

Sürüm: 1.1

İçindekiler

Bölüm I Genel kurallar

- Modüllerinizi kümedeki bir bilgisayardan ya da bir sanal makine kullanarak oluşturmanız gerekir:
    - Sanal makineniz için kullanılacak işletim sistemini seçebilirsiniz
    - Sanal projenizi gerçekleştirmek için gerekli tüm yazılımlara sahip olmalıdır. Bu yazılımlar yapılandırılmalı ve kurulmalıdır.
- Ya da araçların mevcut olması durumunda doğrudan bilgisayarı kullanabilirsiniz.
    - Oturumunuzda tüm modüller için ihtiyacınız olanları yükleyecek alana sahip olduğunuzdan emin olun (kampüsünüzde varsa goinfre kullanın)
    - Değerlendirmelerden önce her şeyi kurmuş olmanız gerekir
- Fonksiyonlarınız tanımlanmamış davranışlar dışında beklenmedik bir şekilde çıkmamalıdır (segmentasyon hatası, veri yolu hatası, çift serbest bırakma, vb. Böyle bir durumda projeniz işlevsel değil olarak kabul edilecek ve değerlendirme sırasında 0 alacaktır.
- Bu çalışma **teslim edilmek zorunda olmasa ve notlandırılmasa** bile projeniz için test programları oluşturmanızı teşvik ediyoruz. Bu size kendi çalışmanızı ve meslektaşlarınızın çalışmalarını kolayca test etme şansı verecektir. Bu testleri özellikle savunmanız sırasında faydalı bulacaksınız. Nitekim, savunma sırasında kendi testlerinizi ve/veya değerlendirdiğiniz akranınızın testlerini kullanmakta özgürsünüz.
- Çalışmanızı size atanmış git deposuna gönderin. Yalnızca git deposundaki çalışmalara not verilecektir. Deepthought çalışmanıza not vermekle görevlendirilirse, bu notlandırma akran değerlendirmelerinizden sonra yapılacaktır. Deepthought'un notlandırması sırasında çalışmanızın herhangi bir bölümünde bir hata meydana gelirse, değerlendirme duracaktır.
- Python 3.10 sürümünü kullanmalısınız
- Alıştırmada yasaklanmamışsa herhangi bir yerleşik işlevi kullanabilirsiniz.
- Lib içe aktarımlarınız açık olmalıdır, örneğin "import numpy as np" yapmalısınız. "from pandas import *" içe aktarımına izin verilmez ve alıştırmada 0 alırsınız.
- Küresel bir değişken yoktur.
- Odin tarafından, Thor tarafından! Beyninizi kullanın!!!

Bölüm II

Egzersiz 00

	Egzersiz 00
Alıştırma 00: İlk python betiği	
Teslim dizini : ex00/ Teslim edilecek dosyalar : `Hello.py` İzin verilen işlevler : `Hiçbiri`	

Aşağıdaki selamlamaları görüntülemek için her bir veri nesnesinin dizesini değiştirmeniz gerekir: "Merhaba Dünya", "Merhaba "kampüsünüzün ülkesi"", "Merhaba "kampüsünüzün şehri"", "Merhaba "kampüsünüzün adı""

```
ft_list  = ["Merhaba", "tata!"]
ft_tuple= ("Merhaba", "toto!")
ft_set       = {"Merhaba", "tutu!"}
ft_dict  = {"Merhaba" : "titi!"}

#kodunuz burada

print(ft_list)
print(ft_tuple) print(ft_set)
print(ft_dict)
```

Beklenen çıktı:

```
$>python Merhaba.py| cat -e
['Merhaba', 'Dünya!']$
('Merhaba', 'Fransa!')$
{'Merhaba', 'Paris!'}$
{'Merhaba': '42Paris!'}$
$>
```

Bölüm III

Alıştırma 01

	Alıştırma 01
Alıştırma 01: Paketin ilk kullanımı	
Giriş dizini : *ex01/*	
Teslim edilecek dosyalar : `format_ft_time.py`	
İzin verilen işlevler : `time, datetime veya tarih almayı başka bir kütüphane`	

Tarihleri bu şekilde biçimlendiren bir kod yazın, elbette sizin tarihiniz örnekteki gibi benimki olmayacak ama aynı şekilde biçimlendirilmelidir.

Beklenen çıktı:

```
$>python format_ft_time.py| cat -e
Ocak 1970'ten bu yana geçen saniye: 1,666,355,857.3622 veya bilimsel gösterimle 1.67e+09$ 21
Ekim 2022$
$>
```

Bölüm IV

Alıştırma 02

	Alıştırma 02
Alıştırma 02: İlk python işlevi	
Dönüş dizini : *ex02/*	
Teslim edilecek dosyalar : `find_ft_type.py`	
İzin verilen fonksiyonlar : `Hiçbiri`	

Nesne türlerini yazdıran ve 42 döndüren bir işlev yazın. Prototipi şu şekilde olmalıdır:

```
def all_thing_is_obj(object: any) -> int:
        #kodunuz burada
```

Senin tester.py:

```
from find_ft_type import all_thing_is_obj

ft_list  = ["Merhaba", "tata!"]
ft_tuple= ("Merhaba", "toto!")
ft_set      = {"Merhaba", "tutu!"}
ft_dict  = {"Merhaba" : "titi!"}

all_thing_is_obj(ft_list)
all_thing_is_obj(ft_tuple)
all_thing_is_obj(ft_set)
all_thing_is_obj(ft_dict)
all_thing_is_obj("Brian")
all_thing_is_obj("Toto")
print(all_thing_is_obj(10))
```

Beklenen çıktı:

```
$>python tester.py| cat -e
Liste : <class 'list'>$
Tuple : <class 'tuple'>$
Set : <class 'set'>$
Dict : <class 'dict'>$
Brian mutfakta : <class 'str'>$  Toto
mutfakta : <class 'str'>$  Tip bulunamadı$
42$
$>
```

İşlevinizi tek başına çalıştırmak hiçbir işe yaramaz.

Beklenen çıktı:

```
$>python find_ft_type.py| cat -e
$>
```

Bölüm V

Alıştırma

03

	Alıştırma 03
Alıştırma 03: NULL bulunamadı	
Dönüş dizini : *ex03/*	
Teslim edilecek dosyalar : `NULL_not_found.py`	
İzin verilen fonksiyonlar : `Hiçbiri`	

Tüm "Null" türlerinin nesne türünü yazdıran bir işlev yazın. İşler yolunda giderse 0, hata durumunda 1 döndürün. İşlevinizin tüm NULL türlerini yazdırması gerekir.

Prototipi şu şekilde oluşturulmalıdır:

```
def NULL_not_found(object: any) -> int:
  #kodunuz burada
```

Senin tester.py:

```
from NULL_not_found import NULL_not_found

Hiçbir şey= Yok
Sarımsak=  float("NaN")
Sıfır      = 0
Boş        = ''
Sahte      = Yanlış

NULL_not_found(Nothing)
NULL_not_found(Garlic)
NULL_not_found(Zero)
NULL_not_found(Empty)
NULL_not_found(Fake)
print(NULL_not_found("Brian"))
```

`Boş= ''`

Beklenen çıktı:

```
$>python tester.py  | cat -e Nothing:
None <class 'NoneType'>$  Cheese: nan
<class 'float'>$  Zero: 0 <class
'int'>$
Boş: <class 'str'>$
Sahte: False <class 'bool'>$
Type not Found$
1$
$>
```

İşlevinizi tek başına çalıştırmak hiçbir işe yaramaz.

Beklenen çıktı:

```
$>python NULL_not_found.py| cat -e
$>
```

Bölüm VI

Alıştırma 04

	Alıştırma 04
Alıştırma 04: Çift ve Tek	
Dönüş dizini : *ex04/*	
Teslim edilecek dosyalar : `whatis.py`	
İzin verilen işlevler : `sys veya argümanları almaya izin veren başka herhangi bir kütüphane`	

Argüman olarak bir sayı alan, sayının tek mi çift mi olduğunu kontrol eden ve sonucu yazdıran bir kod oluşturun.

Birden fazla bağımsız değişken sağlanmışsa veya bağımsız değişken bir tamsayı değilse, bir **AssertionError**.

Beklenen çıktı:

```
$> python whatis.py 14
Ben bile.
$>
$> python whatis.py -5
Tuhafım.
$>
$> python whatis.py
$>
$> python whatis.py 0 Ben
bile.
$>
$> python whatis.py Merhaba!
AssertionError: bağımsız değişken bir tamsayı değil
$>
$> python whatis.py 13 5
AssertionError: birden fazla bağımsız değişken sağlandı
$>
```

Bölüm VII

Şu andan itibaren şu ek kurallara uymanız gerekmektedir

- Global kapsamda kod yok. Fonksiyonları kullanın!
- Her programın bir ana programı olmalı ve basit bir komut dosyası olmamalıdır:

```
def main():
  # testleriniz ve hata işlemleriniz

if name== " main ": main()
```

- Yakalanmayan herhangi bir istisna, test etmeniz istenen bir hata durumunda bile alıştırmaları geçersiz kılacaktır.
- Tüm fonksiyonlarınızın bir dokümantasyonu olmalıdır (_____doküman____)
- Kodunuz normda olmalıdır
  - pip install flake8
  - alias norminette=flake8

Bölüm VIII

Alıştırma 05

	Alıştırma 05
Alıştırma 05: İlk bağımsız python programı	
Dönüş dizini : *ex05/*	
Teslim edilecek dosyalar : `building.py`	
İzin verilen işlevler : `sys veya argümanları almaya` izin `veren başka herhangi bir kütüphane`	

Bu kez, tek bir dize argümanı alan ve büyük harf karakterlerinin, küçük harf karakterlerinin, noktalama karakterlerinin, rakamların ve boşlukların toplamlarını görüntüleyen bir main ile gerçek bir otonom program yapmalısınız.

- Hiçbiri veya hiçbir şey , kullanıcıdan bir dize girmesi istenir.
- Programa birden fazla argüman sağlanmışsa, bir **AssertionError** yazdırın.

Beklenen çıktılar:

```
$>python building.py "2008 yılında yayınlanan Python 3.0, önceki sürümlerle tamamen geriye dönük uyumlu olmayan
    büyük bir revizyondu. Python 2, 2020'de 2.7.18 sürümü ile sonlandırıldı."
Metin 171 karakter içermektedir:
2 büyük harf
121 küçük harf
8 noktalama işareti
25 alan
15 rakam
$>
```

Beklenen çıktılar: (satır başı boşluk olarak sayılır, eğer bir tane döndürmek istemiyorsanız ctrl + D kullanın)

```
$>python building.py
Sayılacak metin nedir? Merhaba
Dünya!
Metin 13 karakter içermektedir:
2 büyük harf
8 küçük harf
1 noktalama işaretleri
2 boşluk
0 rakam
$>
```

Odin tarafından, Thor tarafından! Beyninizi kullanın!!! Tekerleği yeniden icat etmeyin, dil özelliklerini kullanın.

Bölüm IX

Alıştırma 06

	Alıştırma 06
Alıştırma 06:	
Dönüş dizini : *ex06/*	
Teslim edilecek dosyalar : `ft_filter.py, filterstring.py`	
İzin verilen işlevler : `sys veya argümanları almaya izin veren başka herhangi bir kütüphane`	

Bölüm 1: Filtre işlevini yeniden kodlayın

Kendi ft_filter'ınızı yeniden kodlayın, orijinal yerleşik işlev gibi davranmalıdır (print(filter.)" ile aynı şeyi döndürmelidir._____doküman_____)"), **liste kavramalarını** kullanmalısınız
ft_filter'ınızı yeniden kodlamak için.

`Elbette orijinal filtreyi dahili olarak kullanmak yasaktır`

`Modülü buradan doğrulayabilirsiniz, ancak aşağıdaki projeler için bilmeniz gereken şeyler olduğundan devam etmenizi öneririz`

Bölüm 2: Program

İki argüman kabul eden bir program oluşturun: bir dize(S) ve bir tamsayı(N). Program, **S'**den uzunluğu **N'**den büyük olan kelimelerin bir listesini çıkarmalıdır.

- Kelimeler birbirinden boşluk karakterleriyle ayrılır.
- Dizeler herhangi bir özel karakter içermez. (Noktalama işaretleri veya görünmez)
- Program en az bir **liste kavrama** ifadesi ve bir **lambda**.
- Bağımsız değişken sayısı 2'den farklıysa veya herhangi bir bağımsız değişkenin türü yanlışsa, program **bir AssertionError** yazdırır.

Beklenen çıktılar:

```
$> python filterstring.py 'Merhaba Dünya' 4
['Merhaba', 'Dünya']
$>
```

```
$> python filterstring.py 'Merhaba Dünya' 99 []
$>
```

```
$> python filterstring.py 3 'Merhaba Dünya'
AssertionError: argümanlar kötü
$>
```

```
$> python filterstring.py AssertionError:
argümanlar kötü
$>
```

Bölüm X

Alıştırma

07

	Alıştırma 07
Alıştırma 07: Sözlükler SoS	
Dönüş dizini : *ex07/*	
Teslim edilecek dosyalar : `sos.py`	
İzin verilen işlevler : `sys veya argümanları almaya` izin `veren başka herhangi bir kütüphane`	

Bir dizeyi argüman olarak alan ve onu Mors Koduna kodlayan bir program yapın.

- Program boşluk ve alfanümerik karakterleri destekler
- Alfanümerik bir karakter noktalar . ve tire işaretleriyle temsil edilir -
- Tam mors karakterleri tek bir boşlukla ayrılır
- Boşluk karakteri eğik çizgi ile gösterilir /

Mors kodunuzu saklamak için bir **sözlük** kullanmalısınız.

```
NESTED_MORSE={          " ": "/ ",
                        "A": ".- ",
                        ...
```

Bağımsız değişken sayısı 1'den farklıysa veya herhangi bir bağımsız değişkenin türü yanlışsa, program **bir AssertionError** yazdırır.

```
$> python sos.py "sos"| cat -e
... --- ...$
$> python sos.py 'h$llo' AssertionError:
argümanlar kötü
$>
```

Bölüm XI

Alıştırma 08

Alıştırma 08

Alıştırma 08: Yükleme ...

Teslim dizini : *ex08/* Teslim edilecek dosyalar : `Loading.py` İzin verilen fonksiyonlar : `Hiçbiri`

Öyleyse ft_tqdm adında bir fonksiyon oluşturalım.
Fonksiyon, tqdm fonksiyonunu yield operatörü ile kopyalamalıdır.

Prototipi şu şekilde oluşturulmalıdır:

```
def ft_tqdm(lst: range) -> None:
  #kodunuz burada
```

Tester.py dosyanız: (kendi sürümünüzü orijinali ile karşılaştırırsınız)

```
from time import sleep from
tqdm import tqdm
from Loading import ft_tqdm

for  eleman in ft_tqdm(range(333)):
            sleep(0.005)
print()
for  eleman in tqdm(range(333)):
            sleep(0.005)
print()
```

Beklenen çıktı: (orijinal versiyona mümkün olduğunca yakın bir fonksiyona sahip olmalısınız)

```
    $> python tester.py
100%|[=============================================================>]| 333/333
100%|                                                                | 333/333 [00:01<00:00, 191,61it/s]
```

Bölüm XII

Alıştırma 09

	Egzersiz 09
Alıştırma 09: İlk paket oluşturma çalışmam	
Dönüş dizini : *ex09/*	
Teslim edilecek dosyalar : `*.py, *.txt, *.toml, README.md, LICENSE`	
İzin verilen fonksiyonlar : `PyPI veya paket oluşturmak için herhangi bir kütüphane`	

İlk paketinizi python'da istediğiniz şekilde oluşturun, "pip list" komutunu yazdığınızda kurulu paketler listesinde görünecek ve "pip show -v ft_package" yazdığınızda özelliklerini görüntüleyecektir.

```
$>pip show -v ft_package
Adı: ft_package Sürüm:
0.0.1
Özet: Örnek bir test paketi
Ana sayfa: https://github.com/eagle/ft_package Yazar: eagle
Yazar-email: eagle@42.fr Lisans:
MIT
Konum: /home/eagle/... Gerekli:
Gerekli:
Metadata-Version: 2.1
Yükleyici: pip
Sınıflandırıcılar:
Giriş noktaları:
$>
```

Paket, aşağıdaki komutlardan biri kullanılarak pip aracılığıyla kurulacaktır (her ikisi de çalışmalıdır):

- pip install ./dist/ft_package-0.0.1.tar.gz
- pip install ./dist/ft_package-0.0.1-py3-none-any.whl

Paketiniz bunun gibi bir komut dosyasından çağrılabilmelidir:

```
from ft_package import count_in_list

print(count_in_list(["toto", "tata", "toto"], "toto")) # çıktı: 2
print(count_in_list(["toto", "tata", "toto"], tutu")) # çıktı: 0
```

Bölüm XIII

Sunum ve akran değerlendirmesi

Ödevinizi her zamanki gibi Git deponuzda teslim edin. Savunma sırasında yalnızca deponuzdaki çalışmalar değerlendirilecektir. Doğru olduklarından emin olmak için klasörlerinizin ve dosyalarınızın adlarını iki kez kontrol etmekten çekinmeyin.

Değerlendirme süreci, değerlendirilen grubun bilgisayarında gerçekleşecektir.